RFcreations
...a different perspective

blueSPY Protocol Analyzer v23.07.03
USER GUIDE

A 2.4 GHz sniffer and protocol analyzer, with support for:

Bluetooth LE, BR/EDR, QBHSL, and mHDT;

HCI; 802.15.4; logic analysis; packet detection for WiFi;

Audio timing and analysis

Peter Ford

3 July 2023

CONTENTS

1 Overview 5
2 Getting started 5
3 Capturing, Saving and Loading files 5
3.1.1 Connect 6
3.1.2 Capture 6
3.1.3 New Capture Segment 6
3.1.4 Capture Settings 6
3.2 Saving and Loading files 7
4 Decrypting 8
4.1 Link Keys 9
4.2 ECDSA Pairing Keys 9
4.3 Broadcast Codes 9
4.4 Encrypted Advertising 9
5 Dashboard 9
6 Audiopod 10
6.1 Capturing and playing audio 10
6.2 Analysing Latency 10
6.3 Using audiopod to measure LE Audio latency 12
6.4 Using audiopod to measure A2DP, HFP, and ASHA latency 14
7 Importing GATT and HCI 15
7.1 Btsnoop import/merge 15
7.2 GATT Import/export 15
7.2.1 Manual correction 16
7.2.2 Android GATT database import 16
7.2.3 blueSPY GATT import and export 16
8 Summary tab 16
8.1 Finding and Filtering packets 17
8.1.1 Protocol and Profile filters 18
8.1.2 Filter Devices 19
8.1.3 Search interface 21
8.2 Choosing and Creating columns 23
8.3 GoTo 24
8.4 Multiple Summary tabs 24
8.5 Linked packets 24

8.6 Specific Views for different Protocol layers 25
9 Audio Export 26
9.1 Playback 26
9.2 Export 27
10 Spectrum and Timeline tabs 28
10.1 Common Interface 29
10.1.1 Filtering 29
10.1.2 Mouse control 29
10.1.3 Keyboard control 30
10.1.4 Throughput plot 30
10.1.5 Settings 30
10.2 Spectrum 31
10.2.1 Statistics display 31
10.3 Timeline 32
11 Other tabs 34
11.1 Details & Raw 34
11.2 Bookmarks 35
11.3 Devices/Connections Details 35
11.4 Audio Streams 36
11.5 Security Key Manager 36
11.6 LE Access Addresses 36
11.7 Topology 36
11.8 Packet errors/issues 36
11.9 Exporting Data 37
12 Queries 38
13 C API 40
13.1 bluespy.h File Reference 40
13.1.1 Classes 40
13.1.2 Typedefs 40
13.1.3 Enumerations 40
13.1.4 Functions 40
13.1.5 Detailed Description 41
13.1.6 Class Documentation 41
13.1.7 Function Documentation 42
14 Python API 47

14.1 bluespy.py File Reference 47
14.1.1 Classes 47
14.1.2 bluespy.BluespyError Class Reference 48
14.1.3 bluespy.error Class Reference 49
14.1.4 bluespy.Packets Class Reference 50
14.1.5 bluespy.event_id Class Reference 51
14.1.6 Functions 54
14.1.7 Variables 54
14.1.8 def bluespy.add_link_key (*key*, *addr0* = 0, *addr1* = 0) 54
14.1.9 def bluespy.capture (*filename*, *CL* = False, *LE* = False, *QHS* = False, *_15_4* = False, *wifi* = False, *MHDT_CL* = False, *MHDT_LE* = False, *spectrum* = 0) 54
14.1.10 def bluespy.close_file () 55
14.1.11 def bluespy.connect (*serial* = −1) 55
14.1.12 def bluespy.disconnect () 56
14.1.13 def bluespy.load_file (*filename*) 56
14.1.14 def bluespy.stop_capture () 56

1 Overview

The RF Creations Protocol Analyzer allows capture and analysis of LE, BR/EDR, QBHSL, and mHDT Bluetooth traffic, 802.15.4 and logic analyser signals; detection of WiFi packets; and power measurements of other signals in the 2.4 – 2.5 GHz band ("Spectrum" data).

2 Getting started

If you are using blueSPY on Windows or macOS, all you need to is download the correct build (with on Windows a choice of an installer or a portable zip file containing the software), plug in your Moreph, and start capturing.

On Linux, if your system has no GUI libraries (e.g. a server) you should download the 'headless' build; otherwise download the full build including the GUI. To use blueSPY you will need to create a udev rule to communicate with the Moreph as an ordinary user. Create

`/etc/udev/rules.d/50-minimoreph.rules`

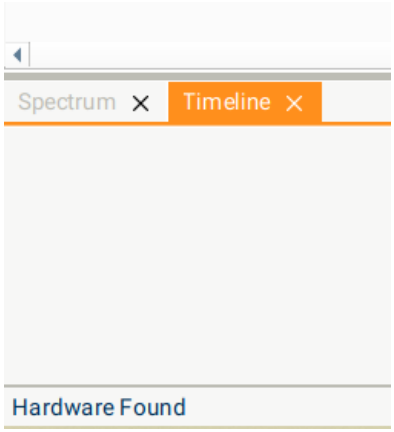
with the following contents:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="2bbd", ATTRS{idProduct}=="00f3",
MODE="0666"
```

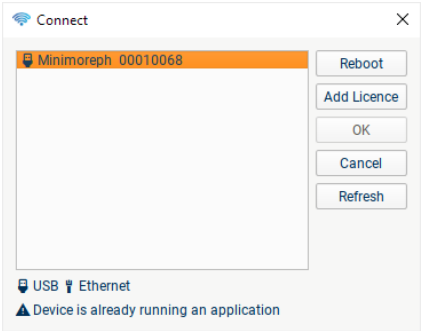
Reboot, or run "`udevadm control --reload-rules && udevadm trigger`" as root. Then you can use it as an ordinary user.

3 Capturing, Saving and Loading files

To capture traffic, you must have a Moreph 30 or a miniMoreph connected to your computer. When a Moreph is plugged in, or when the application is started, the status bar will display "Hardware Found"; or "No Hardware Found" if nothing is detected. The status of the hardware (idle, or capturing) will then be shown in the bottom-right of the status bar.

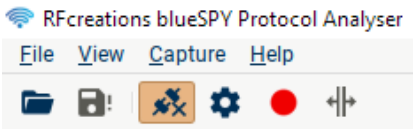


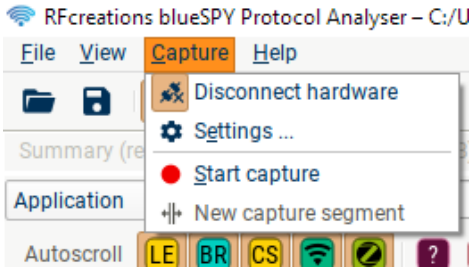
If the Moreph is not automatically detected, or you want to choose between multiple devices, use the "Connect" window to connect to a Moreph. If the Moreph is already running an application, you will need to Reboot it.



This window also allows you to add a licence to the Moreph, if you have been sent a new licence file for the device.

Once a Moreph is connected, the Capture Settings and Capture buttons will be enabled:





3.1.1 Connect

Open the Connect window, or disconnect the Moreph if it's already connected.

3.1.2 Capture

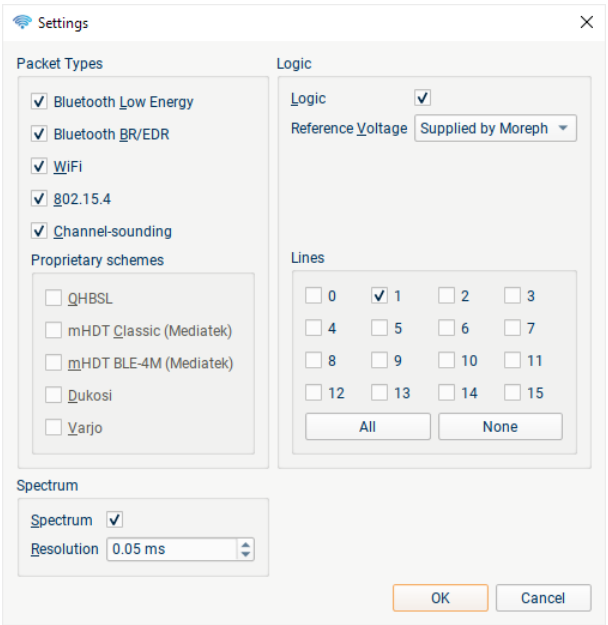
Starts and stops capture.

3.1.3 New Capture Segment

During a capture, seamlessly stops capturing and starts a new capture file; no packets are missed, so all traffic will be present in either the first or second capture. After the new capture has started a Save dialog will appear to allow you to choose a filename at your leisure. NB: Currently, connection timing is not preserved across the "new segment" operation, so only use this button at a time when none of your DUTs are connected.

3.1.4 Capture Settings

Use the Capture Settings window before starting a capture to select what data the Moreph should record. Each type of traffic can be enabled/disabled, and the time resolution of the Spectrum data can be selected; it is best to only capture the traffic you need, for the sake of capture file size if nothing else!



Not all combinations of packet types can be captured simultaneously due to hardware limitations; in addition, your licence may restrict which packets you can enable. If you have a licence which allows it, the following combinations of packets are possible:

1. QHS, Dukosi, Varjo, 802.15.4
2. Channel-sounding, 802.15.4
3. mHDT Classic, mHDT BLE-4M

Logic, Spectrum, WiFi, LE, and BR/EDR are supported in any of the combinations.

If that seems complicated, an alternative strategy: if a checkbox for a proprietary/unusual packet type is disabled, trying unchecking all other checkboxes and enabling it, and then see which other checkboxes are still enabled.

When using the logic analyser, it is important to specify the source of the reference voltage. If you are connecting an external reference voltage to the VREF pin (voltages between 0.8V and 3.6V are supported), select “Supplied Externally”. Otherwise, select “Supplied by Moreph”; the Moreph will supply 3.3V on VREF.

Although very fine resolution (5 µs) Spectrum capture is possible, using it leads to very large file sizes (many GB) and so it is best to disable Spectrum capture or set a longer time resolution if fine timing is not needed.

3.2 Saving and Loading files

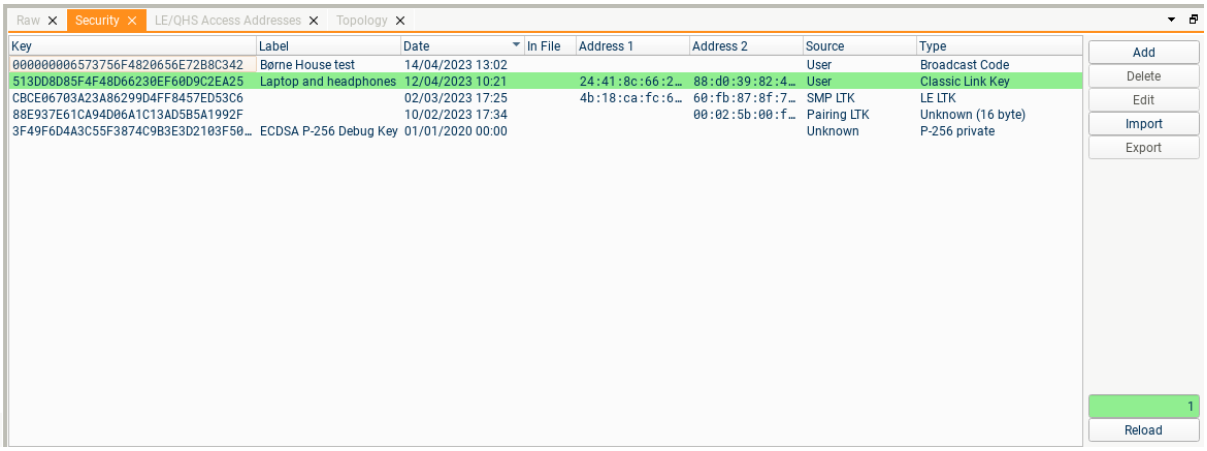
“Open” and “Save As” operate in the usual way, and are found in the File menu. A new capture always writes to a temporary file; afterwards the file can either be saved or discarded.

“Save Advanced” can be used to save a portion of a large file, for instance by omitting Spectrum data, selecting a shortened time-period, or saving only some of the packets. A Moreph must be connected to “license” the file. NB: Saving a subset of the packets may result in a file in which the packets can no longer be parsed, e.g. if the beginning of a Connection is missed. To minimise filesize: disable all protocol filters in the Summary; enable the device filter to show only devices you need; select the timerange you need in Save Advanced, and use “Save Packets: Shown in Summary” (with “Save Spectrum” not checked).

If you have added information to a file (e.g. adding decryption keys, adding Bookmarks, modifying device filtering) then the “Save” icon in the toolbar will change to show that you may want to save your changes:

4 DECRYPTING

Most Bluetooth communication is encrypted, and a key is required to decrypt it. In blueSPY this is handled through the Security tab, which stores a persistent database of keys.



The supported key types are BR/EDR link keys, LE LTKs, ECDSA pairing keys, Broadcast Codes and Encryted Advertising keys. You can add or import keys, and delete, edit or export the currently selected keys using the buttons on the right. Adding or editing a key will show the following dialog:

You only need to enter the key, the key type can be automatically detected and in most cases the MAC addresses can too. The label is also optional, but helps keep track of the keys. The 'Convert' menu allows you to easily perform some byte reordering operations on keys. When editing a key, you can press Duplicate to add a new similar key instead of replacing the old one.

After adding a key, press "Reload" to re-parse the current file using the new key. If a key is successfully used to decrypt some traffic, it will be highlighted in green in the Security table, and the number above the reload button will show the number of successful keys. Any successful keys will be stored in the file when the capture is saved.

Since there is often confusion about the endianness of keys, the software will try both the key as entered and the byte reversed version, and correct the key if it was entered reversed.

If a key is seen in an HCI, ATT or SMP message it will automatically be added and used.

Devices using LE Legacy Pairing are insecure, and the packets are decrypted without the need to add any keys as long as the pairing has been captured.

4.1 Link Keys

Link keys for BR/EDR or LE are entered as 16 hexadecimal bytes (32 characters), byte 0 first. The fields for adding MAC addresses do not need to be filled in usually, however if either of the full addresses for a BR/EDR device were not found during a capture you will need to add them manually. The addresses will be automatically filled in once a successful decryption has occurred.

4.2 ECDSA Pairing Keys

These are entered as a 24 or 32 byte (for P-192 and P-256 respectively) hexadecimal big endian integer. The corresponding public key is shown when the full key is entered and the appropriate type is selected so you can verify it has been entered correctly.

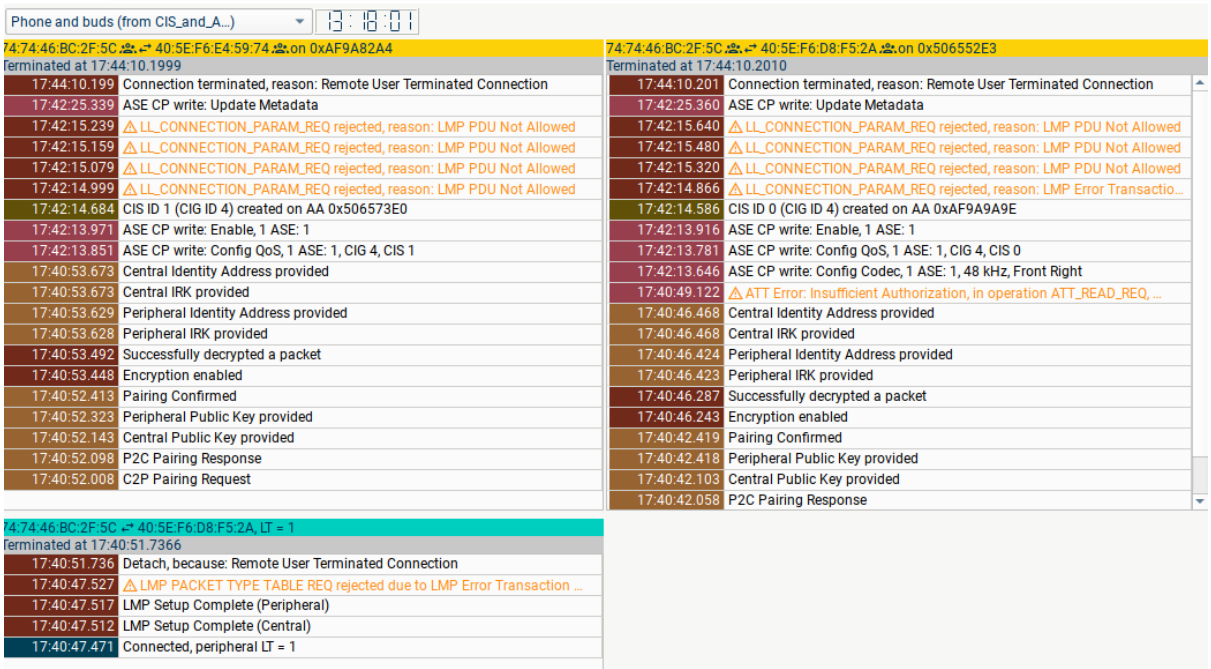
4.3 Broadcast Codes

You can enter broadcast codes as a 16 hexadecimal bytes like the other keys, or you can enter the code string directly if you select the 'Broadcast Code' key type. The first character of the code is the last byte of the hexadecimal version, as defined in the Bluetooth specification. You can convert between the code string and hexadecimal form using the convert menu.

4.4 Encrypted Advertising

Encrypted Advertising keys are entered as 24 hexadecimal bytes, a 16 byte key followed by an 8 byte IV.

5 Dashboard



The Dashboard tab provides a brief summary of the connections selected in a device filter; a list of notable events (encryption start, audiostream creation, connection termination etc), and any warnings or errors seen. Some warnings aren't really a problem (e.g. the LMP Transaction Collision shown above), but if you are trying to answer the question "what went wrong with the Bluetooth

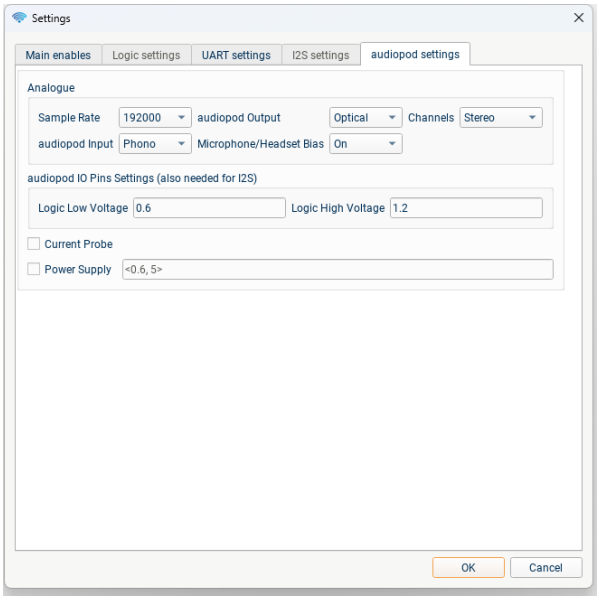
connection/stream" these messages are likely to be a good place to start; you can double-click any of them to jump to that point in the Summary/Timeline/etc to investigate further.

We look out for known bugs and highlight them in the Dashboard, so if you would like to see some particular bug or type of message displayed, let us know.

6 AUDIOPOD

Using the audiopod accessory, you can record and play back analogue or digital audio (or use the provided microphones), timestamped using the same clock as the sniffer uses to timestamp the Bluetooth packets on-air. This is particularly useful for measuring conformance with the Presentation Delay requirements of LE Audio, and can also be used to measuring timing, quality, and the effects of missed or retransmitted packets on all types of Bluetooth audio streams.

6.1 CAPTURING AND PLAYING AUDIO



To capture traffic using audiopod: connect the audiopod to the Moreph, use the USB-C port to provide power, and connect audio cables to whichever of the ports you want to use. Open the Settings dialog, enable the "audiopod" checkbox in the main tab, and switch to the "audiopod" tab to choose the sample rate, input and output audio ports, and other settings. Settings (other than volume controls, and enables for the AGC and DRC on input/output) must be selected before starting the capture.

For audio output, you can either provide a file for blueSPY to play (.wav or .mp3), or use the simple tone/chirp generating options in the audiopod tab. NB: A constant tone or other short-period periodic signal will not work for measuring audio latency/synchronisation! White noise is a good choice.

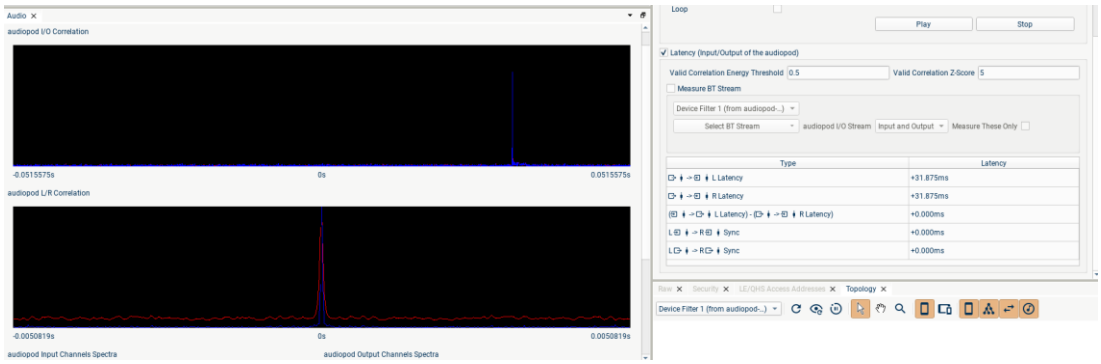
6.2 ANALYSING LATENCY

Audiopod introduces two new tabs to blueSPY, the "audiopod" control tab and the "Audio" tab displaying graphs of correlation and audio spectrum. In the top half of the "audiopod" tab are the volume/gain controls, and controls over the signal generation or file choice for the audio output. In the bottom half of the tab, you can choose which Bluetooth audio streams you want to correlate with the analogue/digital audio inputs and outputs. Below these controls, you will see a table containing either latency/synchronisation measurements, or "N/A" if no correlation was found between the signals you are comparing.

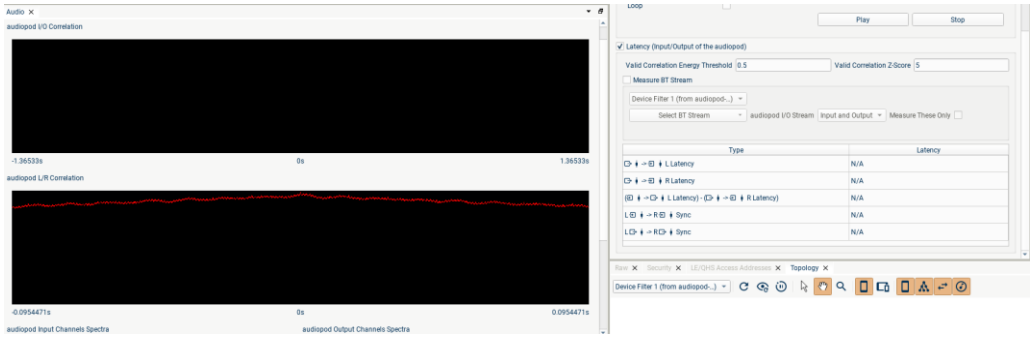
To trigger a measurement, choose a time to measure at (the calculation uses one second of audio centred at this point) by clicking on a packet, e.g. in the Timeline. While capturing you can also use "Autoscroll" in the Timeline; this will cause the calculations to continuously update using the most recent second of audio.

The “Audio” tab shows two types of graph; how many graphs are shown depends on what signals you have chosen to measure.

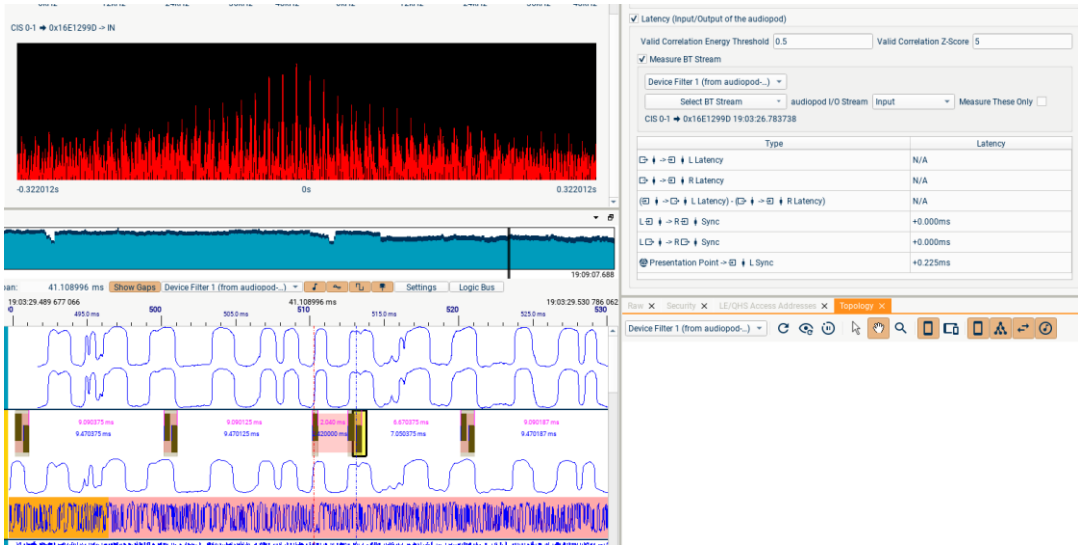
The first type of graph shows the output of a cross-correlation of two of the audio signals. If the two audio streams contain the same signal but one is delayed, you should see a sharp single peak whose x-axis position represents the time-delay:



If the signals are unrelated you will see no graph, or just noise:



If instead of one peak you see a number of peaks, this is likely to indicate that the audio signal you are using is too similar to a periodic signal, e.g. a single musical note:



Ideally you should change the audio input to something with a wider bandwidth. If this isn't possible, you may still be able to get a latency measurement, but you may need to reduce either/both of the thresholds in the audiopod tab to lower the criteria for what is considered a valid correlation.

NB: You don't need to try to read time values from the graph! The position of the highest peak (if it meets the specified thresholds) will be reported in the table of latencies.

The second type of graph displays the spectra of the audio signals at the audio input and output. Currently this is purely an indication of what audio is currently being heard; in a subsequent update we will measure the difference between the input and output spectra to measure the frequency response of the channel.

6.3 USING AUDIOPOD TO MEASURE LE AUDIO LATENCY

The example capture "audiopod_LE_Audio_CIG.pcapng" provides an example of using audiopod to measure all of the relevant subsections of latency of an LE Audio stream. The physical setup in this capture was:

1) Audio was played by audiopod into the input headphone jack of a development board.
2) The development board transmitted this audio over two CISes (left and right) to two other development boards.
3) The headphone jack outputs from these boards were combined and sent into the stereo jack input of audiopod.

Audiopod provides accurate timestamps for the audio at points (1) and (3), and the Moreph provides accurate timestamps for the on-air packets of (2). Using these timestamps and the detected CIS and audio parameters, we are able to calculate the "SDU Synchronization Reference" as detailed in this diagram from the TMAP test specification:

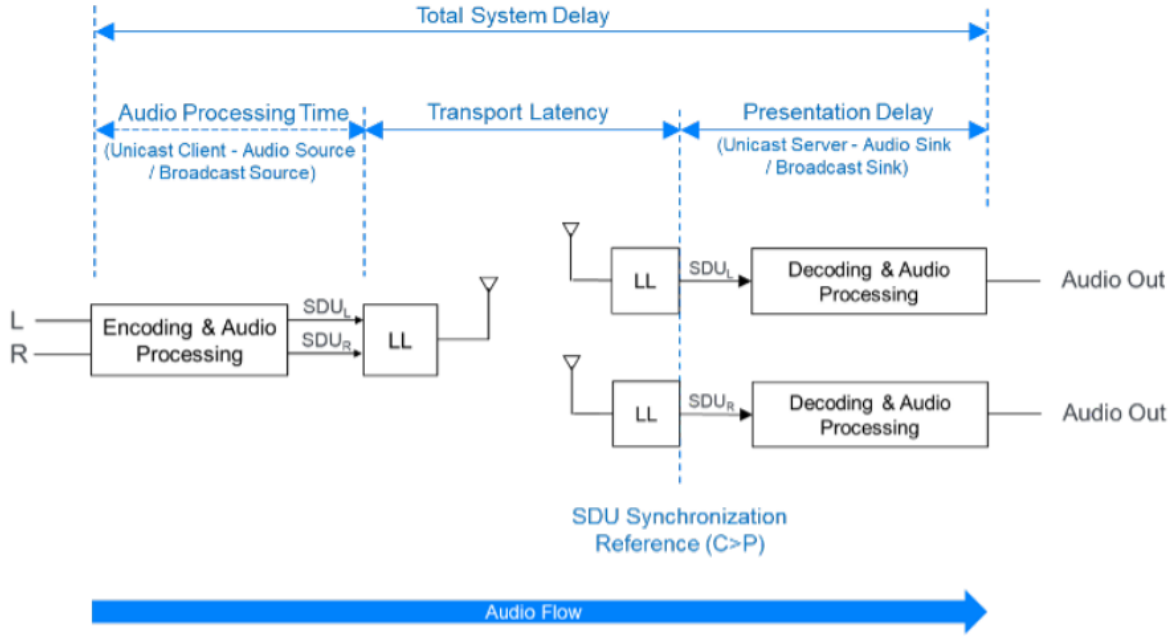
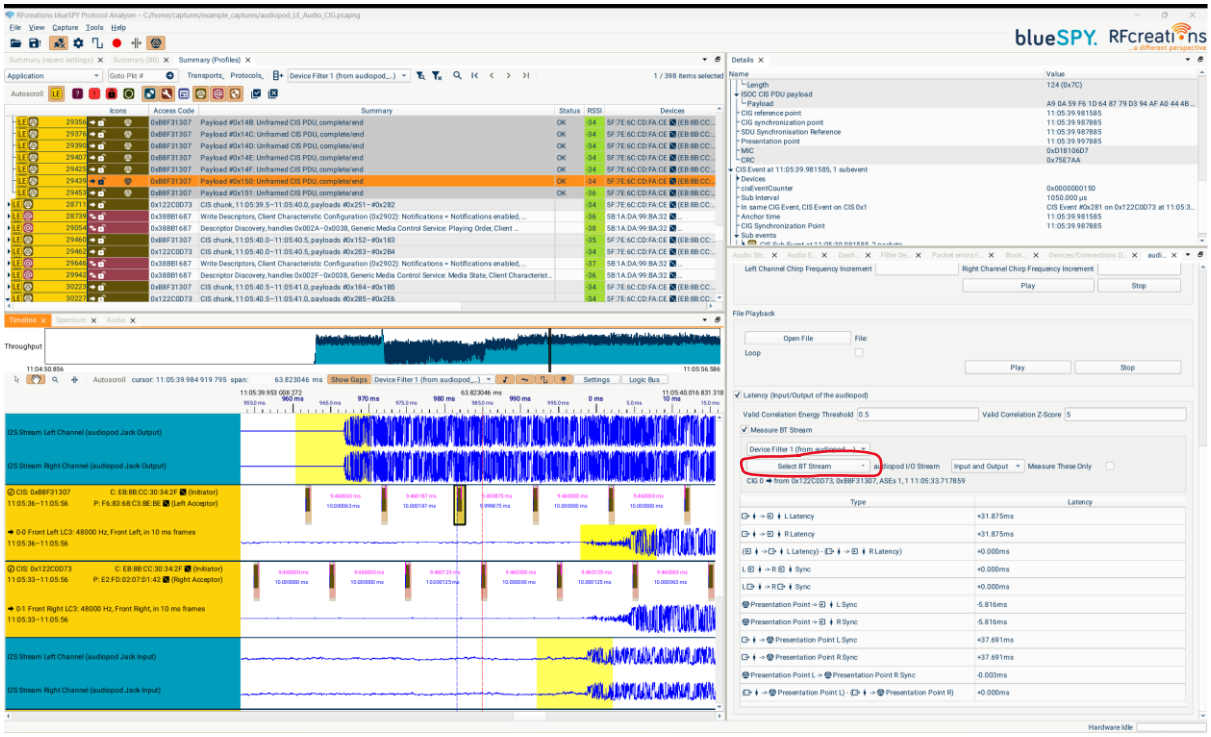


Figure 3.10: Unicast total system delay

If you select a baseband packet and go to the Details pane, you will find the SDU Synchronisation Reference for that packet, along with some of the other timing references related to the packet and the CIS Event.

To select a Bluetooth stream to compare with the audiopod input/output:

- First of all, ensure that the relevant devices are selected in the Device Filter
- Click on a suitable time in the Timeline, when the audio is playing.
- The audiostreams involving the filtered devices which are playing at that timepoint will now be available to select in the circled dropdown; here we have selected the CIG to measure both channels of the stereo audio.



The four rows of audio displayed here correspond to the three points in the audio chain mentioned above. The first row is the stereo output from audiopod, point (1). The next two rows are the two CISes (left and right), point (2). The final row is the output from the two development boards, returned to the stereo input of audiopod, point (3).

The yellow highlighted regions represent the same 10 ms frame of audio, at the different points:

1) The 10ms of audio played into the development board starting at 11:05:39.960194 is captured by the Central development board
2) The left channel of this audio is encoded using LC3 and transmitted in packet #29439, sent at 11:05:39.981585. Using the CIS Offset, CIG & CIS Sync Delays, Presentation Delay and other CIS parameters, it's possible to calculate the correct start time or "Presentation Point" for the frame of audio: 11:05:39.997885. This Presentation Point is the start of the yellow highlighted region on the second row.
3) The audio from packet #29439 is decoded by the development board and played out of its headphone jack starting at 11:05:39.992069, i.e. 5.816 ms early. This is displayed in the table of latencies as "Presentation Point -> audiopod L channel input Sync : -5.816 ms" instead of the ideal value of 0.000 ms

So although the total latency from audiopod output to audiopod input is a commendable 31.875 ms, this has unfortunately been achieved via a miscalculation of the correct presentation delay to apply.

On the most important measure, the Left-Right Sync, this implementation has done rather better. The audio played out of audiopod here is identical on the two channels, and so can be directly compared in the OTA packet and at the analogue output; the synchronisation here is almost perfect.

This example showed a setup where cabled audio input and output were possible; this allows measuring the end-to-end latency, and all of the sub-components of the latency defined in the TMAP and GMAP specifications (e.g. the audio input -> SDU sync reference timing is tested for Peripheral -> Central “uplink” audio).

When cabled audio is not available, the microphones can be used to record a DUT’s speaker’s output (e.g. by placing the earbuds you are testing into the earcanal microphones), and wired earbuds or other speakers plugged into audiopod’s outputs can be used to play into a DUT’s microphone.

When audio input to the system is not available (e.g. when testing using a smartphone or laptop as the Central), the second portion of the system (SDU Sync Reference -> audio output) can still be measured, provided a reasonable source of audio (music with a reasonably wide bandwidth) is available on the device.

6.4 USING AUDIOPOD TO MEASURE A2DP, HFP, AND ASHA LATENCY

In the Bluetooth audio specifications prior to LE Audio:

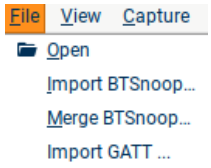
1) the correct time to render the audio is not defined;
2) the transport packets are not always sent at fixed intervals;

and so the calculations carried out by blueSPY are a little different. Instead of showing the discrepancy between a frame of audio’s “Presentation Point” and the time at which it was rendered, we show the time interval between a packet containing audio (e.g. an AVDTP packet) and the time it is rendered by the receiver. As in this case the transport isn’t synchronous, we show the mean, minimum, and maximum value of this time interval within the 1 second of audio used.

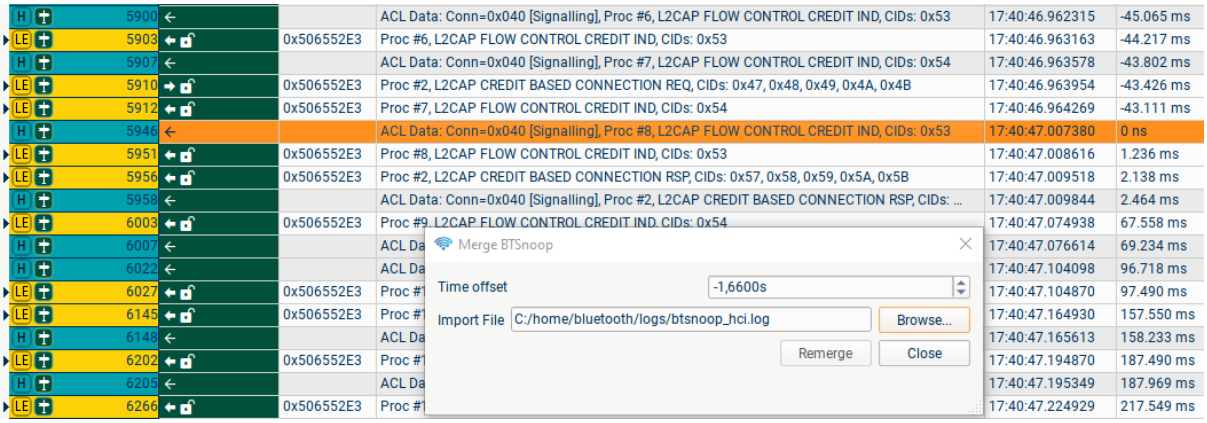
7 IMPORTING GATT AND HCI

If you have information to add to the air packets captured by the sniffer, they can be imported into the airtrace and displayed integrated with the Bluetooth traffic. Currently we only support import of GATT, and HCI from btsnoop logs, but various other formats/methods of import are planned. If there is a format you would particularly appreciate, let us know and we can prioritise it.

7.1 BTSNOOP IMPORT/MERGE



The btsnoop file format (with slight variations) is used by (at least) Android and blueZ for logging HCI traffic. If you have files in this format, they can either be imported (creating a new capture file) or merged into an existing file, including ongoing captures. NB: When you merge into a live capture, the files will be merged and the resulting files reloaded from scratch. No packets will be missed, but the resulting reload may take some time for large captures; you can watch the progress using the orange bar in the bottom right corner.



There is usually some timing offset between the clock used for the on-air capture (derived from your laptop's clock) and the clock of the Android phone. To get a rough time-alignment, find some packets which are present in both the airtrace and the HCI log (L2CAP Control packets are a good choice as they have a clear procedure number), eyeball the difference between the streams, and input the time difference before using "Remerge".

7.2 GATT IMPORT/EXPORT

In LE connections, missing GATT attribute definitions can make it very hard to understand the traffic you capture, and in particular can prevent decode of LE Audio in some cases. Attribute definitions may be missing because of a few corrupted packets, or may be entirely missing if the capture contains a reconnection of paired devices which are using GATT caching. We attempt to cache on your computer any GATT traffic that we see and use it in subsequent captures, but this will fail if:

(a) the initial pairing was not captured using blueSPY and the same computer
(b) the devices do not read the Database Hash attribute, or the read is in a CRC-fail packet or otherwise corrupted.

We currently support three methods of fixing this problem, detailed below. After adding GATT definitions using any of these methods, the Summary strings and any new Details trees will be updated immediately; but to reanalyse a CIG and decode audio using the new information, you will need to press reload in the Security tab.

7.2.1 Manual correction

Individual attributes with a missing definition can be edited by clicking on a relevant packet and right-clicking on the attribute in the Details tree, and then setting the UUID (16-bit, 32-bit or 128-bit). This is probably most useful for adding a missing ASE Control Point definition; fixing this gives most of the information required for decoding LE Audio.

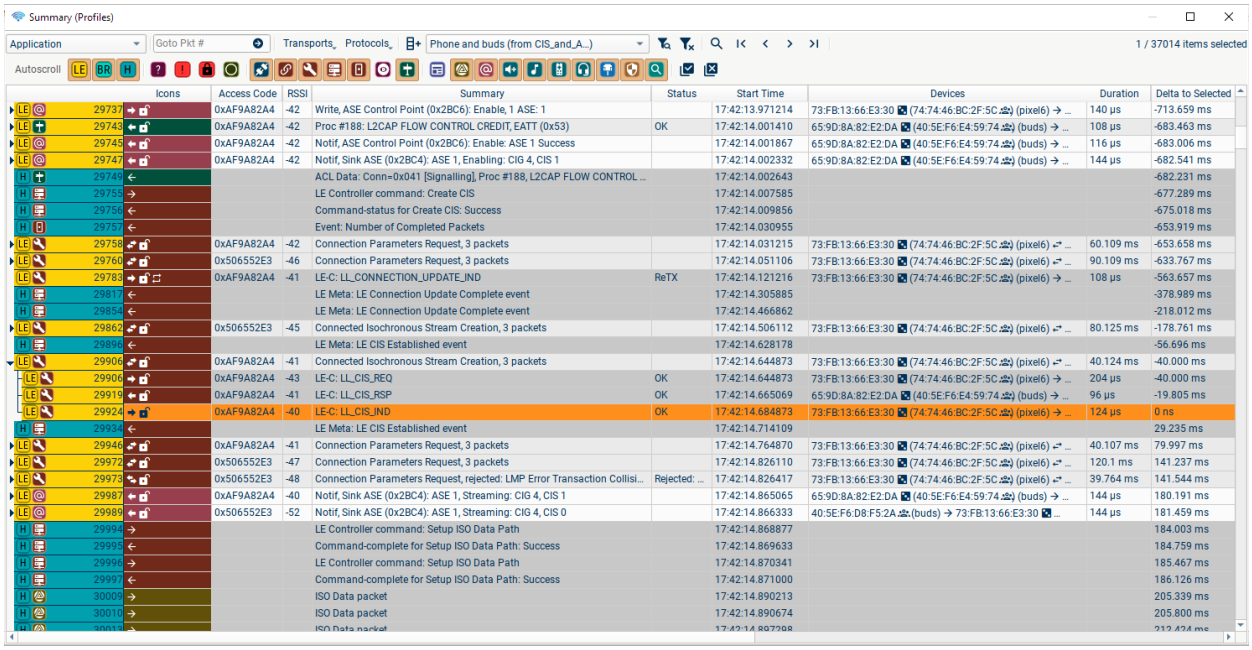
7.2.2 Android GATT database import

If one of your DUTs is an Android phone which you have root access to, you can copy the files from /data/misc/Bluetooth (`adb pull /data/misc/bluetooth`) and import the gatt_hash_* files you find there (File -> Import GATT…)

7.2.3 blueSPY GATT import and export

If the initial pairing was captured in blueSPY on another laptop, you can open the relevant capture, export the GATT database information (File -> Export -> Export GATT), and then import the file to add missing GATT to a subsequent capture. These files use a simple JSON format detailing the attribute handles and the matching UUIDs, so if you are able to generate these files from your build system or fix missing attributes in an exported file then blueSPY can import this information.

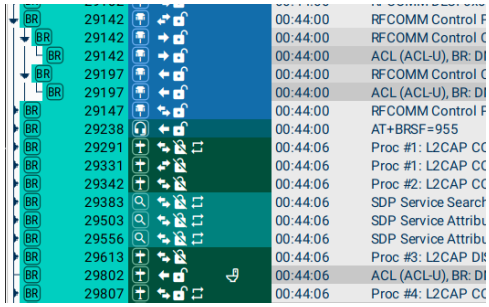
8 SUMMARY TAB



The Summary tabs are the main interface and control window for the application. By default they display all of the captured packets, aggregated into e.g. GATT Procedures, LE Advertising Events, etc.

Selecting a packet or higher-layer event in the Summary will:

- display more information about the packet in the Details tab;
- load the raw bytes into the Raw tab;
- scroll the Spectrum and Timeline tabs to the correct time, and select the event there unless it is hidden in that view.



Expanding a higher layer event (using the arrow at the left-hand edge) will display the constituent packets from the lower protocol layer.

During Capture, the “Autoscroll” button in the top left of the window causes the view to automatically scroll, displaying the most recently captured packets. “Autoscroll” will be disabled when a packet is selected. Equivalent “Autoscroll” buttons are in the top left of the Spectrum and Timeline tabs, to provide independent control of this feature.

1. View selector
2. GoTo box
3. Add/modify columns, and add coloured columns matching some filter.
4. Choose which Device Filter to use, or choose no device filtering.
5. Complex filter popup
6. Search
7. Transport filter buttons
8. Protocol and Profile filter buttons

8.1 Finding and Filtering Packets

To find and display packets of interest there are three main tools. If you are unsure which filters are currently enabled, hover over the “Clear All Filters” button to see a summary. If you can’t see packets you are expecting to receive, check whether they appear if you “Clear All Filters”!

8.1.1 Protocol and Profile filters

The simplest filtering tool consists of three sets of buttons. The buttons enable/disable the following types of packet:

1. Transports:

   BR BR/EDR “Classic” traffic

   LE LE traffic

   WiFi packets

   802.15.4 packets, e.g. Zigbee

   H HCI packets

   Q QBHSL traffic

   CS Channel-sounding tones and sync packets

… and various other proprietary transports.

2. Empty/degraded packets:

   Unintelligible packets: CRC failures, packets not dewhitened, incomplete L2CAP PDUs

   Encrypted packets

   Unknown packets

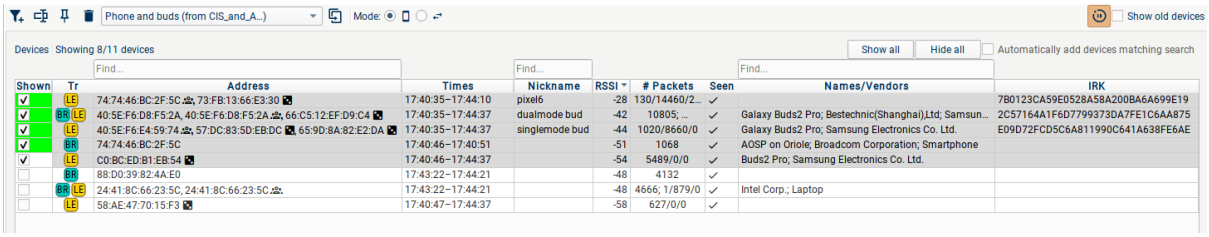
   Empty packets

3. Protocols/other categories of packet (LE advertising, FHS, etc). E.g.

The “All” and “None” buttons are shortcuts to enable/disable all toggle buttons in this row.

Descriptions of these icons, and all other icons and symbols used in the software, can be found in the “Glossary of Symbols” in the Help menu (shortcut F1).

8.1.2 Filter Devices



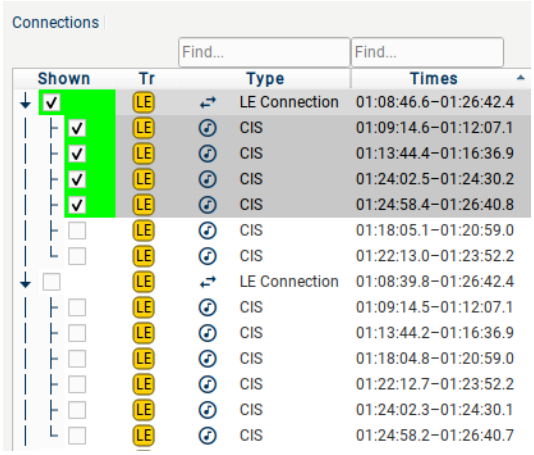
The second filtering tool allows LE and BR/EDR devices to be hidden or displayed. It is possible to create multiple device filters, which can be used independently in different tabs; so for instance you can create a “Broadcasters” filter to display in one Summary tab, and a “Phone and earbuds” filter to display in a second Summary tab.

Use the dropdown menu to select which filter you are editing, and the New Filter, Rename Filter and Delete Filter buttons to modify the list of filters. In each of the other relevant tabs (Summaries, Timeline, Spectrum, Topology, Dashboard) there is a dropdown menu to select one of the filters you have created, or to choose No Device Filtering. If you want to use one filter throughout blueSPY, press the “Use this filter in all tabs” button in the Filter Device tab.

Any device filter you interact with (create or modify) during capture will be stored in the capture file on Save; to make a device filter available for future captures, press the Pin button and it will be stored on your computer until you “unpin” it.

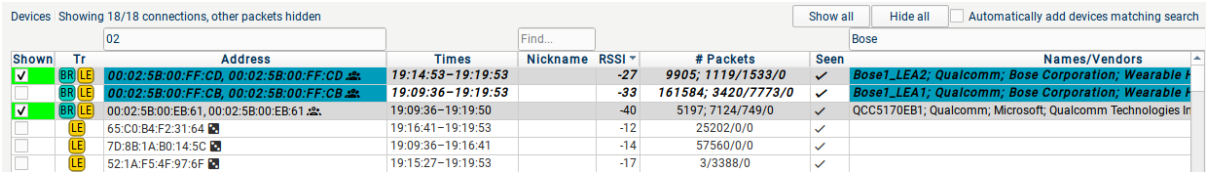
The filter has two modes, selected using the “Devices” and “Connections” radiobuttons:

In Devices mode, the filter displays all packets involving a selected device, and all packets on any connection involving the device. In Connections mode, you can filter these packets further; the connections and audio streams involving the selected devices are shown in the lower pane of the tab, and you can choose which connections, and which streams (CIS, A2DP etc) on a connection to show.



When you use Connections mode, the Shown column of the Devices pane changes colour to highlight which connections involving a device are enabled in the filter; solid green for showing all connections, pale green for showing some connections, and clear for showing no connections involving that device.

In either pane, you can search for devices using any of the “Find…” boxes at the top of a column. Devices matching your search will be shown in bold, with a blue background in the relevant columns, and brought to the top of the pane. NB: If you enter text in more than one “Find…” box, the search only finds devices matching both search strings; so if you type in the Names/Vendors field and nothing appears, check that you haven’t left a partial address in the Address box.



To find the devices you want to display, there are two tools to help:

1. You can sort the panes by any of the columns.
2. You can filter based on substring matching on either Address (ignoring colons), Nickname, or Names/Vendors (case-sensitive).

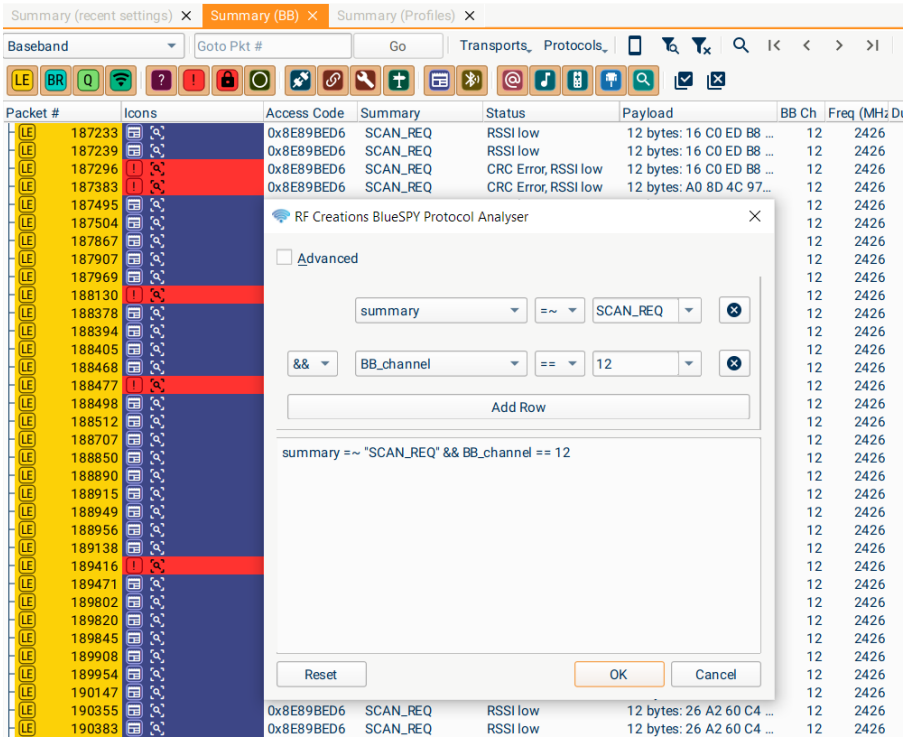
In the Devices pane of the filter, you can check “Automatically add devices matching search” to have devices added for you; e.g. if the device you are testing is frequently changing RPA (and no IRK has been seen), you can add its name or vendor to the search box and have all copies of the device added to the filter.

Any known IRKs are shown in the rightmost column of the tab (you may need to expand the tab or scroll right on lower resolution displays to see this column). If you know an IRK that has not been seen by the sniffer, you can add the IRK in this column. You can also set a nickname for any of the devices, and this nickname will then be used in place of/as well as the address in other parts of blueSPY.

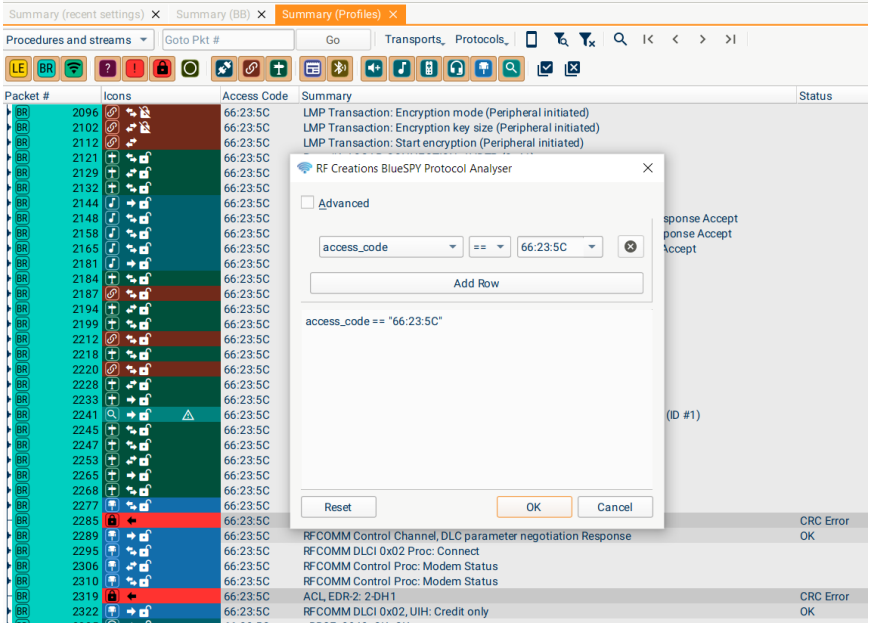
8.1.3 Search interface

The final, more complex search tool is the Search and Filter interface. This allows you to search on a variety of queries associated with the packets. The available queries are listed below. It supports matching strings by regular expressions using the =~ operation. The supported regular expression syntax is documented here. It also supports == and != for exact comparison, < > <= >= for comparing numerical values, and && and || for combining expressions with logical AND or OR respectively.

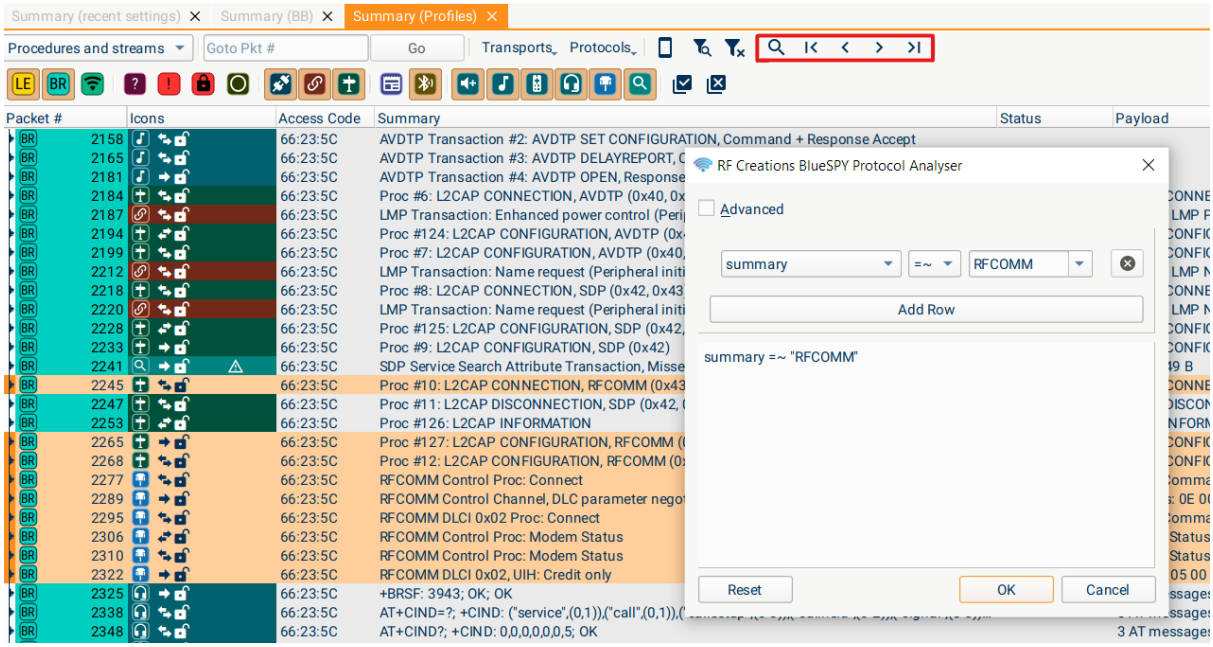
For example, the search below finds all SCAN_REQs sent on channel 12. We use the =~ operation to find any summary containing the string SCAN_REQ.



The following search illustrates filtering a particular LAP.

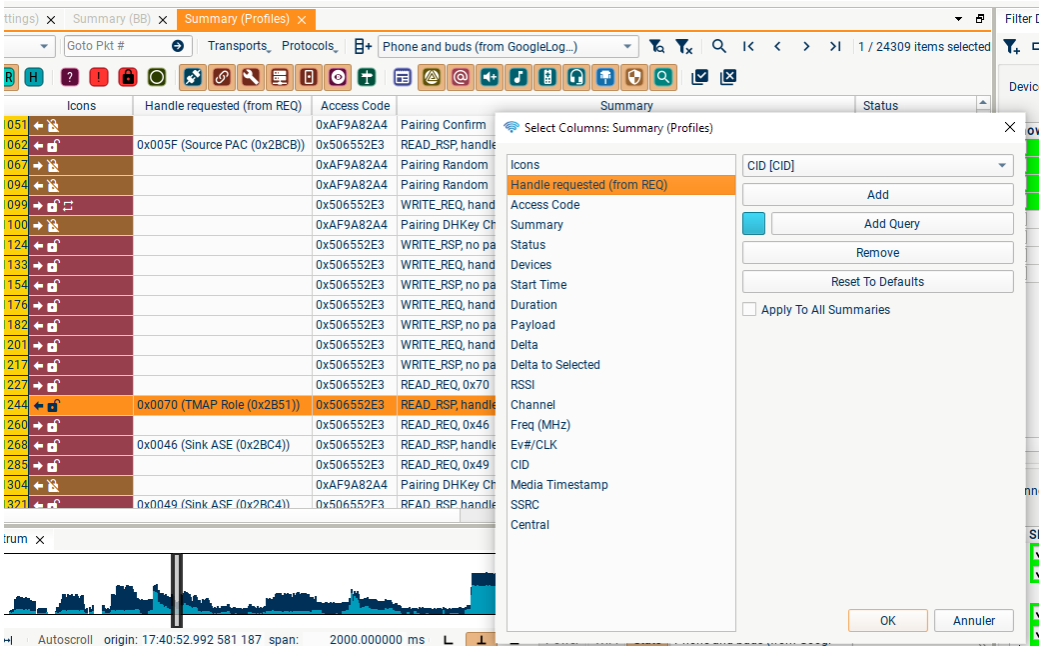


A similar interface can also be used to select packets matching a given search; the arrows to the right of the Search icon can then be used to step between the selected events.



8.2 CHOOSING AND CREATING COLUMNS

In each Summary tab, you can choose which columns to show, and add new ones, using the Select Columns dialog (launched using this button).



There are three types of columns:

1. Columns using one of the documented queries
2. Columns created by clicking on a non-expandable field in a Details pane and dragging it into a Summary tab. These columns will mostly only show data when an identical field is present in another packet; so in the screenshot a column was created by dragging the “Handle requested” field from a ATT_READ_RSP details pane, and it is only showing information from other READ_RSP packets.
3. Columns showing a chosen colour when a particular query is true. For instance, when you have two similar connections to two earbuds, you can add a column highlighting which packets/transactions are on which Access Address:

8.3 GoTo

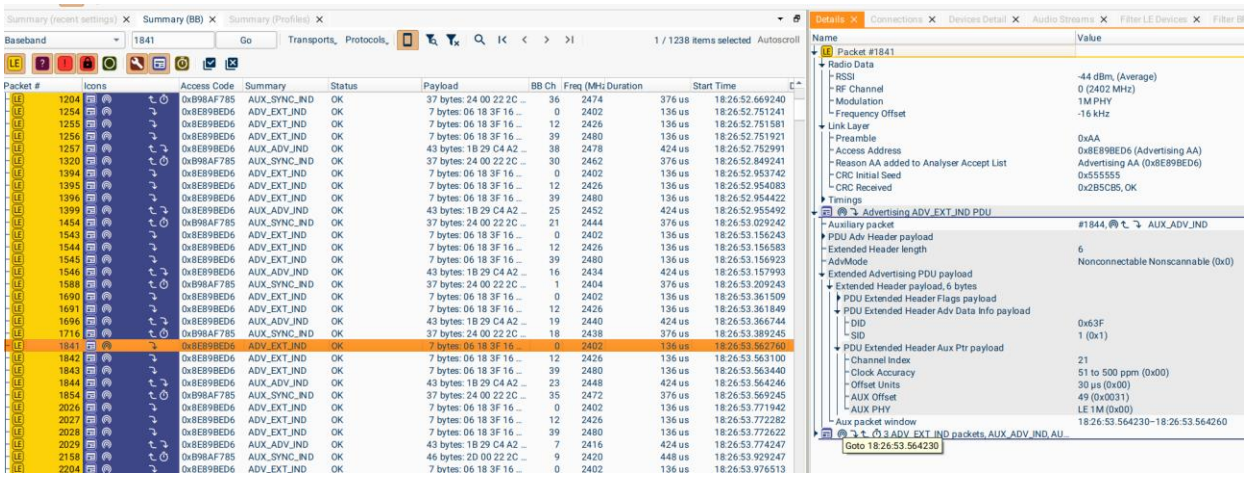
If you have previously found a packet of particular interest, you can use the GoTo box to navigate quickly to that packet using its packet number.

8.4 Multiple Summary tabs

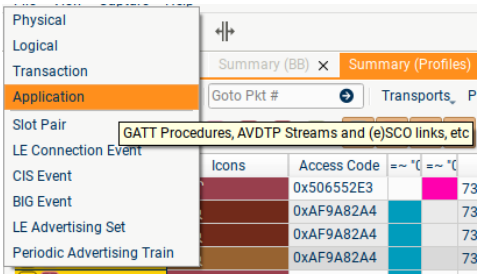
Using the View menu, you can open extra Summary tabs; the filtering and aggregation options can be selected independently in these tabs, so you can e.g. keep one tab open showing baseband packets with no filtering, and another showing just the protocols you are interested in. These tabs can also be renamed by right-clicking in the tab.

8.5 Linked packets

If a baseband packet or a timestamp is referenced in the Details, Connections, or Devices tabs, double-clicking on the text will navigate to that packet (or to the first of that set of packets) in the Summary tab. For example, double-clicking on the “Aux packet window” field in this extended advertising packet will Goto 18:26:53.564230, and select packet #1841 which falls in that window.

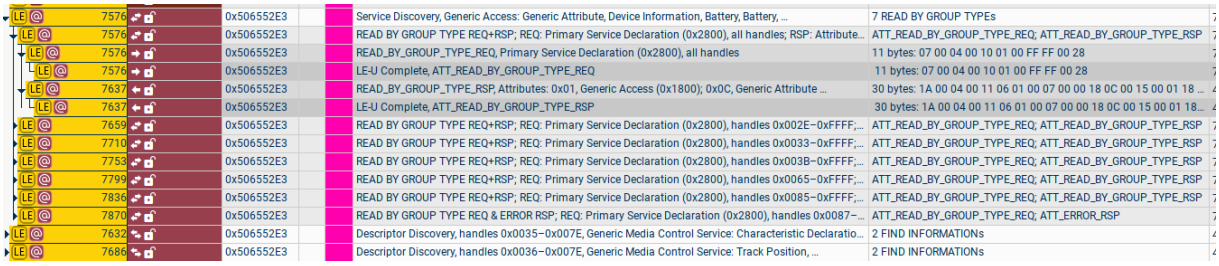


8.6 SPECIFIC VIEWS FOR DIFFERENT PROTOCOL LAYERS



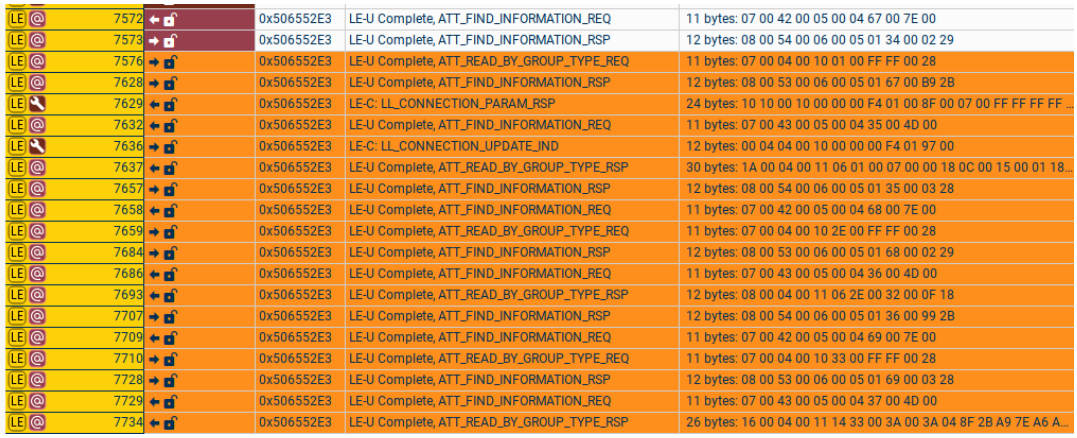
Several different Views are available in the Summary window, displaying different protocols and different types of event. Some of the Views are most useful for studying timing and NACKs/ReTXes in the lower layers, whereas others are better for following the flow of data in higher-level protocols.

The first four Views organise packets at increasing levels of abstraction/aggregation, collecting together packets that are part of e.g. a single L2CAP packet, a single Transaction involving multiple L2CAP packets, or a procedure consisting of a sequence of Transactions.



LE	7576	0x506552E3		Service Discovery, Generic Access: Generic Attribute, Device Information, Battery, Battery, ...	7 READ BY GROUP TYPEs
LE	7576	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), all handles; RSP: Attribute...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7576	0x506552E3		READ_BY_GROUP_TYPE_REQ, Primary Service Declaration (0x2800), all handles	11 bytes: 07 00 04 00 10 01 00 FF FF 00 28
LE	7576	0x506552E3		LE-U Complete, ATT_READ_BY_GROUP_TYPE_REQ	11 bytes: 07 00 04 00 10 01 00 FF FF 00 28
LE	7637	0x506552E3		READ_BY_GROUP_TYPE_RSP, Attributes: 0x01, Generic Access (0x1800); 0x0C, Generic Attribute ...	30 bytes: 1A 00 04 00 11 06 01 00 07 00 00 18 0C 00 15 00 01 18 ...
LE	7637	0x506552E3		LE-U Complete, ATT_READ_BY_GROUP_TYPE_RSP	30 bytes: 1A 00 04 00 11 06 01 00 07 00 00 18 0C 00 15 00 01 18...
LE	7659	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), handles 0x002E–0xFFFF;...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7710	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), handles 0x0033–0xFFFF;...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7753	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), handles 0x003B–0xFFFF;...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7799	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), handles 0x0065–0xFFFF;...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7836	0x506552E3		READ BY GROUP TYPE REQ+RSP; REQ: Primary Service Declaration (0x2800), handles 0x0085–0xFFFF;...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_READ_BY_GROUP_TYPE_RSP
LE	7870	0x506552E3		READ BY GROUP TYPE REQ & ERROR RSP; REQ: Primary Service Declaration (0x2800), handles 0x0087–...	ATT_READ_BY_GROUP_TYPE_REQ; ATT_ERROR_RSP
LE	7632	0x506552E3		Descriptor Discovery, handles 0x0035–0x007E, Generic Media Control Service: Characteristic Declaratio...	2 FIND INFORMATIONs
LE	7686	0x506552E3		Descriptor Discovery, handles 0x0036–0x007E, Generic Media Control Service: Track Position, ...	2 FIND INFORMATIONs

The GATT Procedure above is shown as displayed using the “Application” View, in which upto 3 levels of aggregation are used. The Procedure consists of seven READ_BY_GROUP_TYPE ATT Transactions, each of which contains two L2CAP packets, each of which is sent in a single LL packet.



LE	7572	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 42 00 05 00 04 67 00 7E 00
LE	7573	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 54 00 06 00 05 01 34 00 02 29
LE	7576	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_REQ	11 bytes: 07 00 04 00 10 01 00 FF FF 00 28
LE	7628	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 53 00 06 00 05 01 67 00 B9 2B
LE	7629	0x506552E3	LE-C: LL_CONNECTION_PARAM_RSP	24 bytes: 10 10 00 10 00 00 00 F4 01 00 8F 00 07 00 FF FF FF FF ...
LE	7632	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 43 00 05 00 04 35 00 4D 00
LE	7636	0x506552E3	LE-C: LL_CONNECTION_UPDATE_IND	12 bytes: 00 04 04 00 10 00 00 00 F4 01 97 00
LE	7637	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_RSP	30 bytes: 1A 00 04 00 11 06 01 00 07 00 00 18 0C 00 15 00 01 18...
LE	7657	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 54 00 06 00 05 01 35 00 03 28
LE	7658	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 42 00 05 00 04 68 00 7E 00
LE	7659	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_REQ	11 bytes: 07 00 04 00 10 2E 00 FF FF 00 28
LE	7684	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 53 00 06 00 05 01 68 00 02 29
LE	7686	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 43 00 05 00 04 36 00 4D 00
LE	7693	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_RSP	12 bytes: 08 00 04 00 11 06 2E 00 32 00 0F 18
LE	7707	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 54 00 06 00 05 01 36 00 99 2B
LE	7709	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 42 00 05 00 04 69 00 7E 00
LE	7710	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_REQ	11 bytes: 07 00 04 00 10 33 00 FF FF 00 28
LE	7728	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_RSP	12 bytes: 08 00 53 00 06 00 05 01 69 00 03 28
LE	7729	0x506552E3	LE-U Complete, ATT_FIND_INFORMATION_REQ	11 bytes: 07 00 43 00 05 00 04 37 00 4D 00
LE	7734	0x506552E3	LE-U Complete, ATT_READ_BY_GROUP_TYPE_RSP	26 bytes: 16 00 04 00 11 14 33 00 3A 00 3A 04 8F 2B A9 7E A6 A...

Viewing the same procedure in the “Physical” View shows the LE LL layer packets in sequence (Empty packets have been hidden here); here we can see that another GATT procedure (using the ATT_FIND_INFORMATION… packets) and some LL Control packets are mixed in with this procedure.

The “Logical” and “Transaction” Views use intermediate levels of aggregation; the first collects together segmented packets (e.g. a complete L2CAP PDU), and the second collects together sets of these packets.

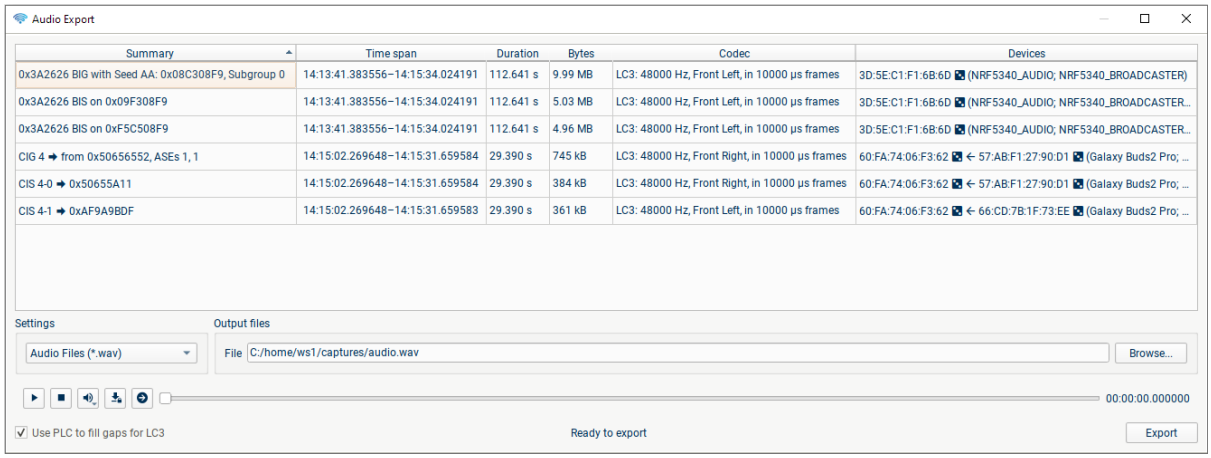
The four Views showing BR/EDR Slot Pairs, LE Connection Events, CIS and BIG Events display limited information about the content of the packets sent, and instead focus on the timings of events, and any missing packets/NACKs/ReTXes.

The LE Advertising Sets View collects together Advertising Events sent by one device at regular intervals. The lower events in this View are usually substantially out of order, as Advertising Sets will often overlap and can be very long.

The Periodic Advertising Train View shows only periodic advertising packets, and not the legacy/extended packets pointing to them. This is particularly useful for analysing Periodic Advertising with Responses, where the View shows the PAwR Events and Subevents.

If no packets are shown when a View is selected, check that there are no Filters/Searches hiding the relevant packets. For example, displaying only BR/EDR packets while on the LE Connection Events View will result in a blank screen!

9 Audio Export



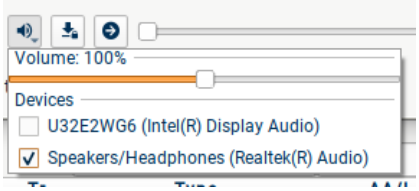
The Audio Export tab allows you to export captured audio to a file, and to play it back live or at a later point. All audio streams that are in the capture and we have some chance of decoding (i.e. we have decrypted them, or they were sent unencrypted) are present, including those for which we are missing some audio parameters (e.g. sample rate) or those using an unsupported or unknown codec; so not all of the rows can necessarily be decoded for playback.

The table can be sorted by any of the columns; the first number printed in the Summary column for BIGes and BISes is the Broadcast Code, to make it easier to find a broadcast with a known Code.

For playback or export of LC3 audio, you have the option to either use the PLC algorithm to fill gaps where packets were missed, or to have an appropriate number of zeroes inserted in the stream.

9.1 Playback

To play some decoded audio, select a row, select an output (headphones, laptop speakers etc) and press play. The audio will start playing from the start of the stream; to hear it (approximately) live, press play and then drag the play position slider all the way to the right.

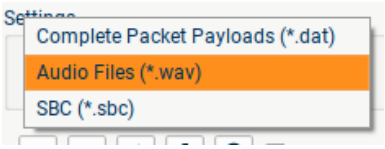


Multichannel streams sent over separate transports (e.g. multiple CISes carrying stereo, or multiple BISes) can be played either individually by selecting the CIS/BIS row, or with both/all channels by selecting the CIG or BIG row.

You can choose to have the Timeline and Summary scroll to remain synchronised with the audio you are hearing, using the Autoscroll button . This works smoothly when the selected Summary is in "Physical" mode and all of the audio packets are visible; otherwise, the autoscroll will be jerky as it jumps between larger chunks of audio in the Summary.

You can also click to go to the packet matching the current audio position, e.g. when the playback is paused.

9.2 EXPORT



Audio streams can be exported either to a WAV file as decoded audio, or to binary files containing the original (decrypted) bytes from the air packets for analysis in other tools (e.g. reference decoders). In addition, for A2DP streams the binary export can either include the AVDTP header ("Complete Packet Payloads"), or only the audio payload; in the case of SBC, this results in a *.sbc file which can be played by standard audio players.

10 SPECTRUM AND TIMELINE TABS

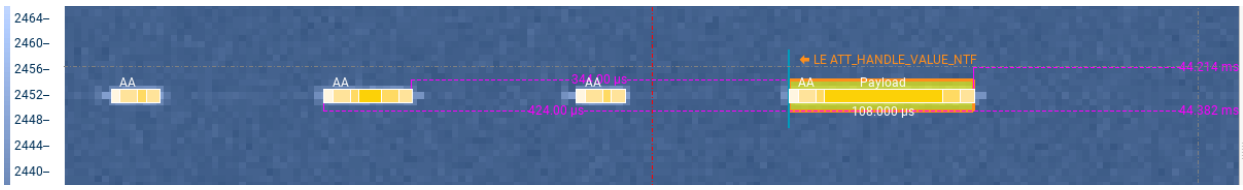
The Spectrum and Timeline tabs have much in common, and take two different approaches to a common goal: Displaying received packets sequentially in time.

In the Timeline tab the packets are grouped vertically by their sender device.

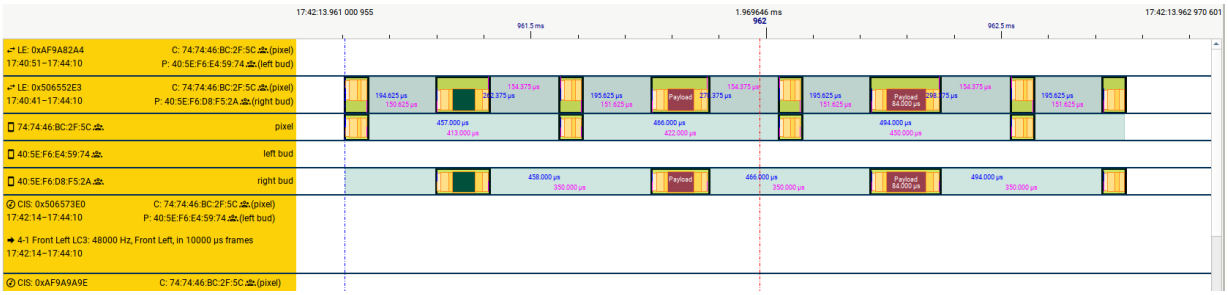
In the Spectrum tab, the packets are plotted according to frequency and time. In addition, the grey background colouring represents the power detected in the spectrum at that time and frequency.

In both tabs, time gaps to earlier/later packets and events are shown. More detail is shown in the Spectrum tab than in the Timeline, but the gaps displayed in the Timeline (if “Show Gaps” is enabled) will always represent gaps between events from the same Device/Connection.

Time deltas are drawn from packet start to packet start; when space allows, the gap between the end of a packet and the following start is also shown.

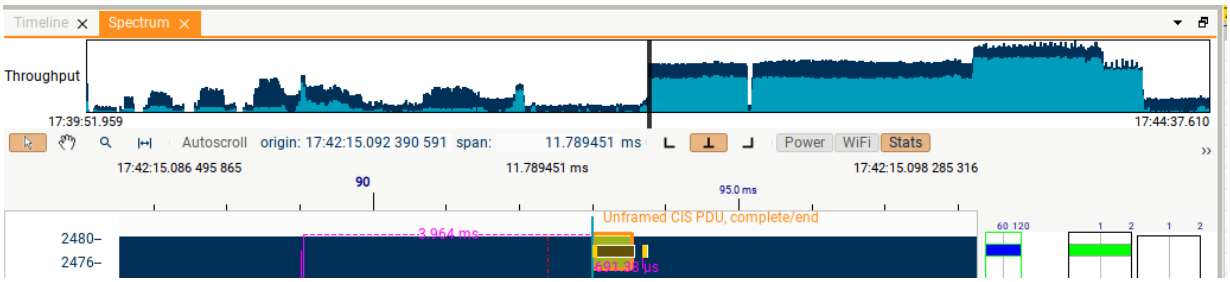


Spectrum

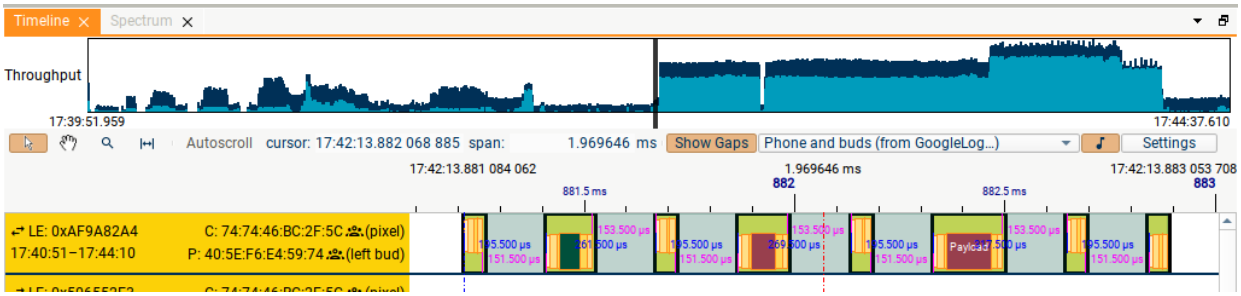


Timeline

10.1 COMMON INTERFACE



Spectrum



Timeline

10.1.1 Filtering

In both the Spectrum and Timeline tabs, the filtering applied corresponds to the filtering in the currently active (last clicked-on) Summary tab, combined with a device filter of your choice, or no device filtering. So to see all packets in the Spectrum or Timeline, clear all filtering in the active Summary tab, and select No Device Filtering in the dropdown in the Spectrum/Timeline. In addition, in the Timeline you have the option to display or hide decoded audio for any suitable audiostreams which are shown.

10.1.2 Mouse control

The mouse can be used in one of four modes: Pointer, Pan, Zoom and Measure. These can be selected using the buttons in the top left of the tab. The Pan, Zoom, and Measure modes can also be enabled temporarily by holding down Shift, Control or Alt respectively.

In all modes other than Zoom mode, the mouse scrollwheel can be used to:

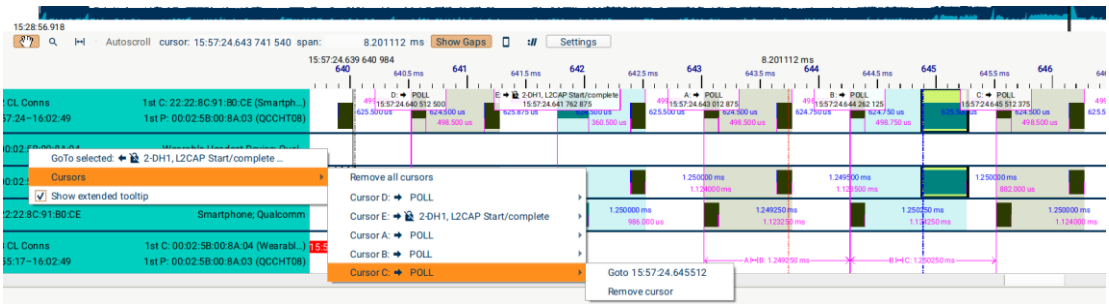
a) scroll vertically in the Timeline view, when the mouse is over the labels at the left-hand side.
b) scroll horizontally in the rest of the Timeline view and in the Spectrum view.

Details of the four modes:

- In Pan mode, the display can be dragged left and right.
- In Zoom mode:
  - The scroll wheel zooms in and out. There are two modes for zooming, to match conflicting customer requests; zooming in and out will either zoom to the mouse pointer, or zoom while leaving the centre of the time window fixed. Choose which mode you want using the context menu in the Timeline/Spectrum.

- A region can be selected by clicking and dragging, to zoom to that time region.

- In Pointer mode, clicking in the Spectrum/Timeline display will select the packet/event closest to the pointer (including selecting it in the Summary and Details tabs).
- In Measure mode, pairs of packets can have cursors added to them; the time difference between cursors is shown, and cursors can be removed or jumped to using the context menu.



In the Spectrum tab, when an event is selected, time gaps to the previous/next event of the same time will be displayed if the zoom level allows. If no time gaps appear, try zooming in! A selected event will also be selected in the Timeline and Summary tabs, unless you have used the filtering controls to hide it in those tabs.

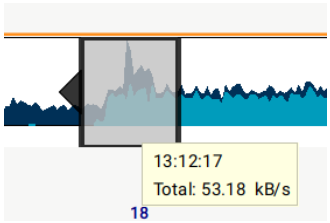
10.1.3 Keyboard control

The arrow keys can also be used to navigate; Up and Down zoom in and out, Left and Right jump earlier and later.

In the Spectrum tab, the A and Z keys increase and decrease the intensity of the Spectrum Power plot. The colourbar at the left-hand side of the tab shows how the grayscale corresponds to power in dBm.

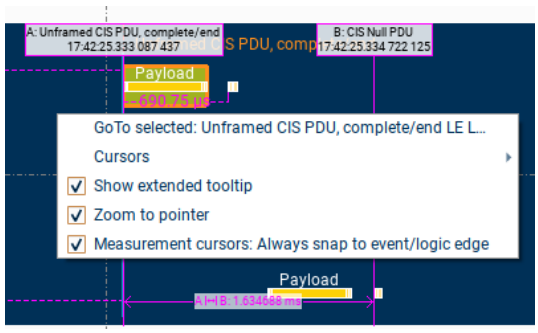
10.1.4 Throughput plot

The throughput plot at the top of each tab shows the whole capture, and illustrates two estimates of Bluetooth traffic rate; one (darker blue) including all packet types, and one excluding Advertising packet and empty packets (LE Empty, NULL/POLLs etc). The black window shows the current time-range shown in the main Spectrum/Timeline area, and this window can be dragged (select the middle of the window) or extended/shrunk (select the arrows which appear to the left and right of the window when you hover).



10.1.5 Settings

Right-click in the rows of the Timeline, or anywhere in the Spectrum to see the context menu. There are several options to control how the GUI behaves:



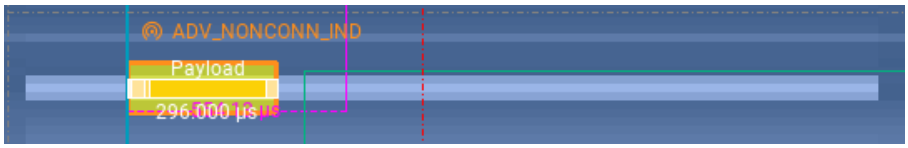
1. "Zoom to pointer" controls the behaviour of the scrollwheel when zooming, as described above.
2. Measurement cursors: as you move your mouse, these can either snap to a packet edge only when close to a packet, or always snap to a packet/other event (e.g. logic line edges).

10.2 Spectrum

In addition to the Power plotted in grayscale, the text in the top-right of the tab displays the power that was measured at the time and frequency corresponding to the pointer position. This text also records the attenuation used by the Moreph's amplifier during capture; an unexpectedly large attenuation value could indicate interference.

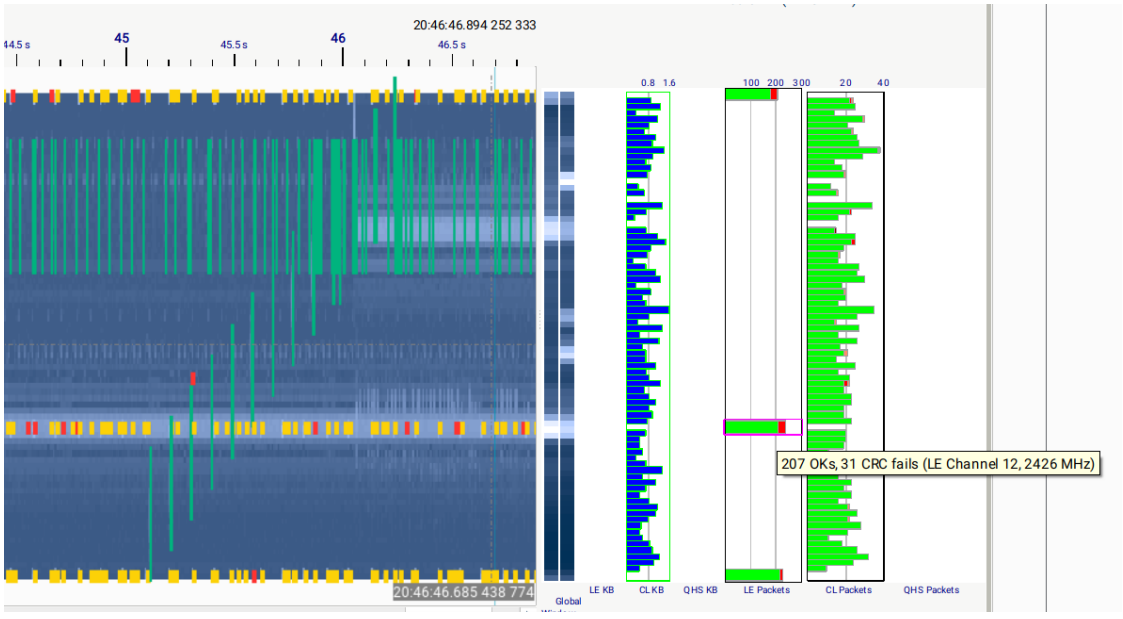
Two important notes:

1. The power displayed in a time interval represents the peak power measured during that interval. This can look slightly confusing when using low-resolution spectrum capture, as a packet can appear as a short blob in the middle of a long grey rectangle.
2. The lengths of the rectangles representing the packet header and other packet segments are:
    a. accurate at high zoom
    b. pictorial at low zoom; the rectangles are wider than the actual time interval in order to prevent them becoming invisibly narrow.



A 376 µs packet marooned in a 2 ms peak-hold window.

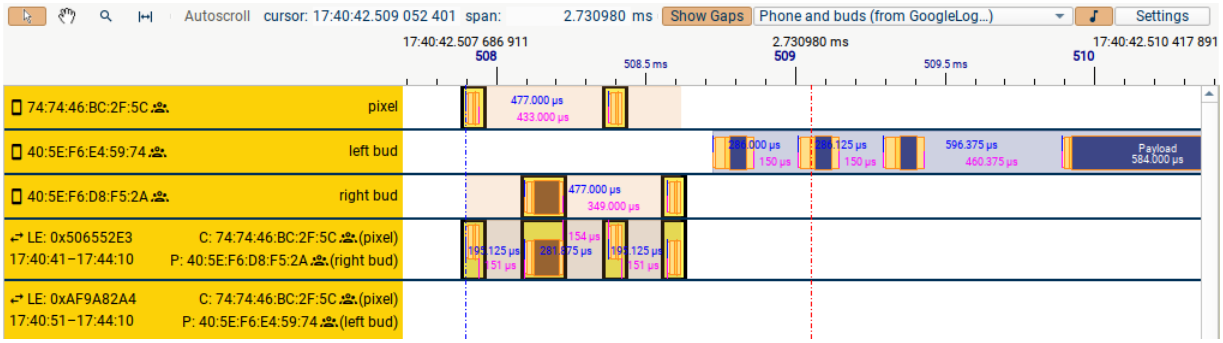
10.2.1 Statistics display



Per-channel statistics showing average power, throughput, and packet success (rates of ReTXes, CRC or dewhitening fails, and Rejections) can be enabled using the "Stats" button. To see details of the values represented by each bar in the bar chart, hover over the bar.

10.3 TIMELINE

The shorter rectangles in solid colours represent on-air packets. The larger translucent rectangles represent groupings of those packets (LE Advertising Events and Connection Events, BR/EDR Slot Pairs, CIS/BIS events). The selected packet or packets are highlighted with a yellow box; as in the Spectrum tab, clicking on a packet will cause it to be selected in the other tabs unless it has been hidden there.



The rows shown in the Timeline are determined by the devices, connections, and audio streams you have enabled in the chosen device filter. If you find that the packets from your DUTs are separated out into too many rows and it's hard to see everything you need, there are two ways to reduce the number of rows:

1. Use the "Connections" mode in the device filter and deselect some of the connections or audio streams. This is helpful when your DUT has various connections to uninteresting 3rd devices and you are only interested in e.g. the phone <-> earbud connections.
2. Use the "Connection Sessions" mode (selected in the Timeline "Settings" dialog). This collapses all connections between a particular pair of devices onto one row, so is particularly helpful if your DUTs are repeatedly disconnecting and reconnecting.

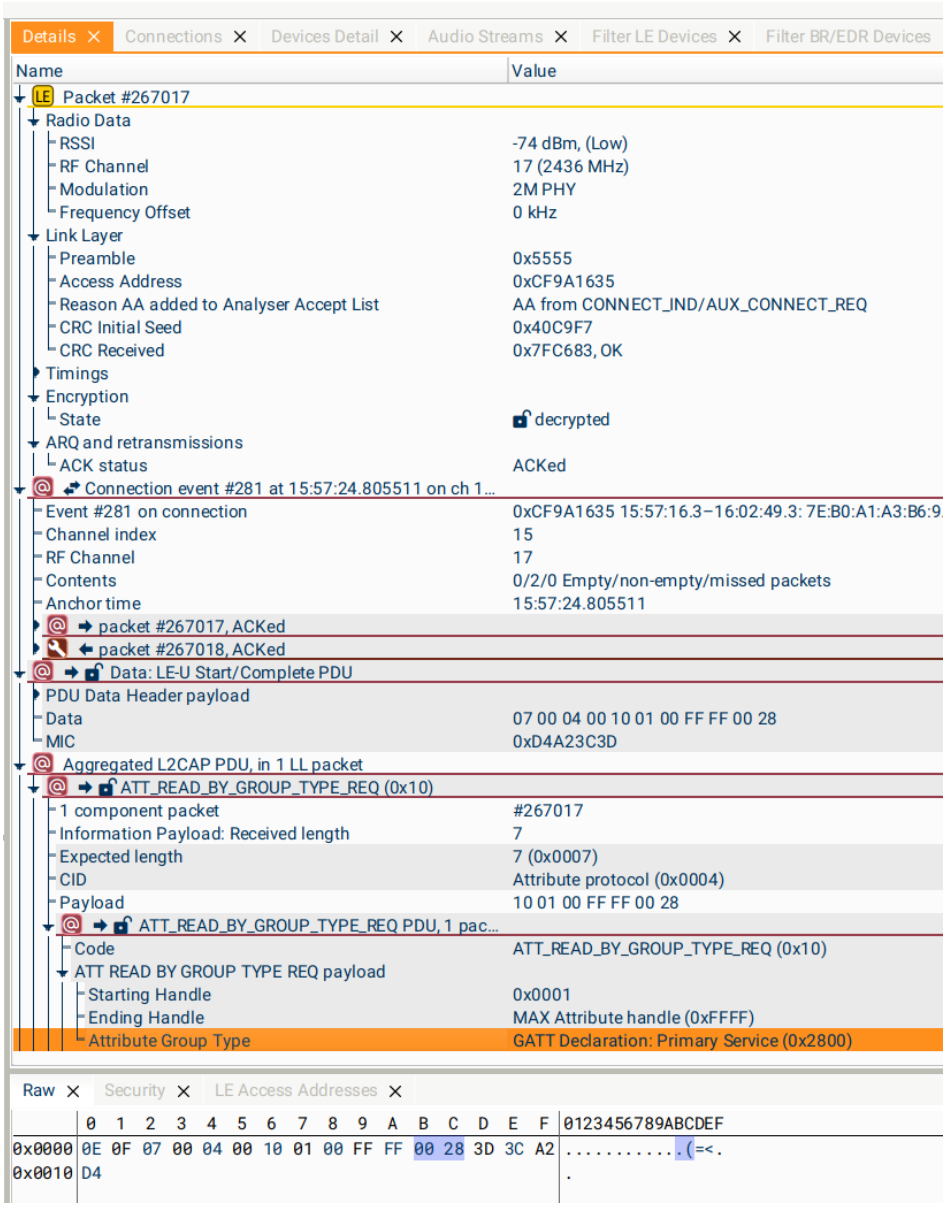
You can control the order the rows are displayed in using the "Settings" popup:

You can “pin” the rows you’re interested in by moving them into the left-hand column (double-click on an item to move it left or right), and choose the order of the pinned rows; these will be displayed at the top of the Timeline view. All other rows, including rows created when new devices/connections are seen, will appear below these, sorted according to your choice from “First Seen”, “Activity” (e.g. number of packets), “Last Seen”, or “Address”.

11 Other tabs

11.1 Details & Raw

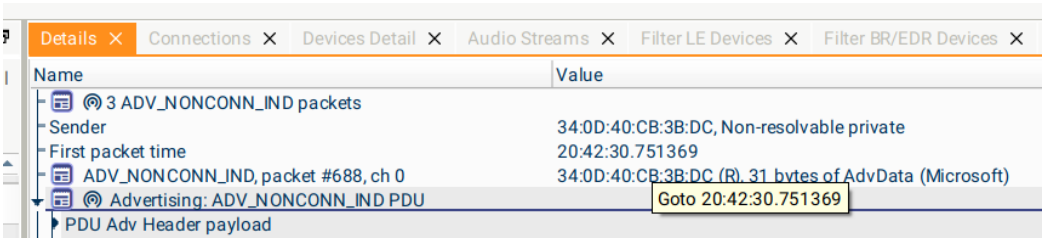
These tabs provide the most detailed look at a packet or higher-layer event. The Details tab is populated whenever an event is selected in the Summary/Timeline/Spectrum tabs, and the Raw tab is also populated when the event has “a payload”.



In the Details pane, baseband packets display some details about the layer(s) above (e.g. the Connection Event and Control Procedure trees displayed for the LE Link Layer packet selected in the screenshot above), and higher-layer packets display some information about the layer(s) below.

Any highlighted fields in a payload shown in the Details tab correspond to bits or bytes within the Raw data. In the screenshot above, the “Attribute Group Type” field has been selected, and so the corresponding bytes 0x00, 0x28 in the Raw data have been highlighted.

Double-clicking on any field containing a time, or referencing another packet, will jump to that location in the Summary. These fields will display a tooltip showing what time you can jump to. Similar jump-to-time fields are present in the Connections and Devices tabs.

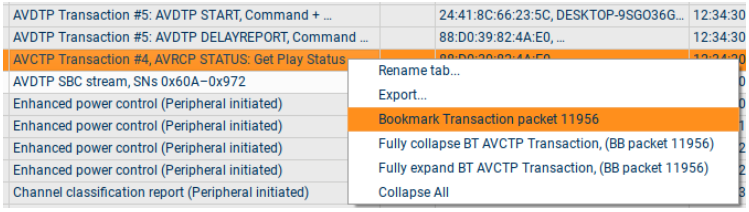


The other fields with coloured backgrounds are section headers, marking different types of information about the packet. The colours used are consistent throughout the application; lighter colours denote baseband packets/information, and darker colours show Connection/Advertising Events, Slot Pairs, and higher protocol PDUs.

11.2 Bookmarks

Any packet or larger event can be bookmarked for later study or to send to other users; when you have added bookmarks, make sure to Save to add them to the file!

Bookmarks can either be added using the context menu in the Summary tab, or by selecting a packet and pressing in the Bookmarks tab. Bookmarks can also be deleted and renamed in this tab; double-clicking on a bookmark, or pressing the button will jump to that packet in the Summary, if it is visible under the current filtering.



11.3 Devices/Connections Details

This tab collects together extra information about the Connections and Devices selected in the filter currently being edited in the Filter Devices tab.

The connections trees show a summary of the Control Procedures/LMP packets sent, and detail of any L2CAP Channels in use. Details of any GATT Characteristics/Services and ATT Transactions seen are also found in the L2CAP Channels subtree.

For an LE Connection, in addition to the above information we collect a list of the Connection Parameters valid at each time during the Connection, including both Parameters transmitted and Parameters we have inferred from an ongoing Connection.

The device trees, in addition to device addresses, collect any other transmitted information including device names, vendors, IRKs, Class of Device, and various types of supported feature.

Each Device tree also includes summaries of the Advertising/Data/Paging/Connected packets sent and received, with links to jump to the first of each of these packet types.

11.4 Audio Streams

This tab displays details of the transport configuration (e.g CIS parameters) of any audiostreams we have seen (in contrast to the audio parameters, which are shown in the Audio Export tab.

11.5 Security Key Manager

See 4 Decrypting.

11.6 LE Access Addresses

This displays the LE Access Addresses that we have seen during the capture; each row is a link to jump to the first packet seen on that AA. Only confirmed AAs are displayed here; packets with CRC failures will not cause an AA to be added to the list.

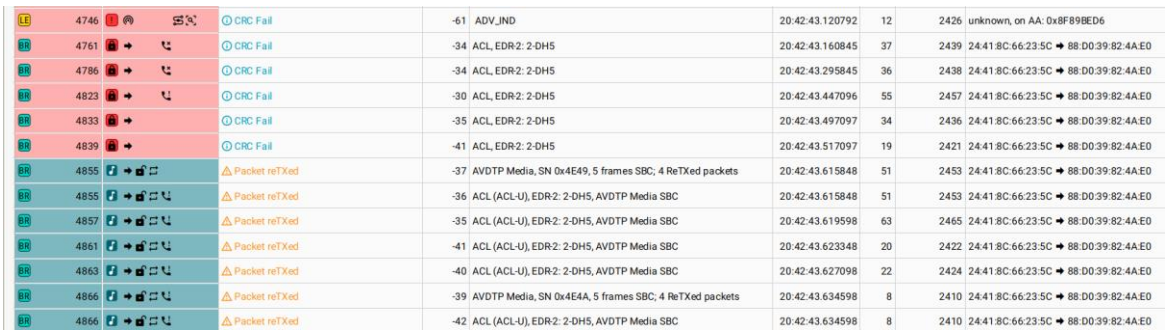
11.7 Topology

This shows the current devices, connections and audiostreams in the chosen device filter. You can enable/disable labels on the devices and the streams, and you can choose to show or hide “Inert” devices, i.e. those only advertising/inquiring and not participating in any displayed connections. The colours of the devices and links indicate whether they are communicating using LE, Classic, or both. The icons decorating the links indicate whether any audio is currently being sent, and in which direction.

11.8 Packet errors/issues

This collects together errors/warnings/informative messages seen during the capture, and allow filtering and sorting them to easily display all the instances of a particular type of issue. The issues displayed include both Bluetooth errors (invalid values, packets with invalid lengths etc), communication impairments (ReTXed packets, rejected packets), receive impairments (CRC fails, MIC fails, missing packets), and analyser difficulties (e.g. protocols not parsed, either due to use of proprietary protocols or features not yet supported).

Double-clicking on a row will navigate to that packet/event in the active Summary tab.

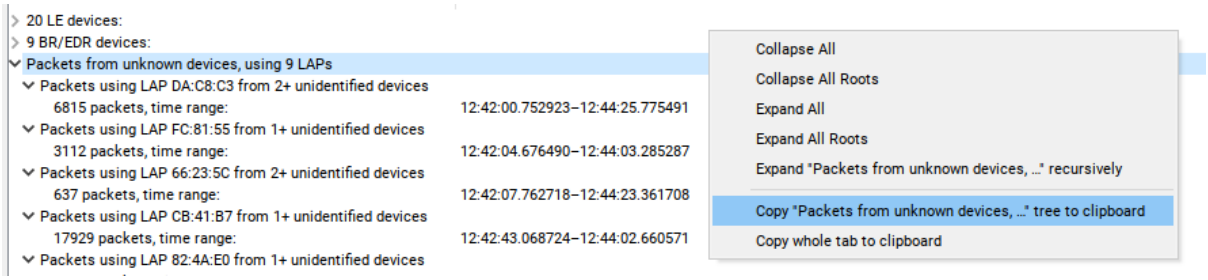


LE	4746		CRC Fail	-61	ADV_IND	20:42:43.120792	12	2426	unknown, on AA: 0x8F89BED6
BR	4761		CRC Fail	-34	ACL, EDR-2: 2-DH5	20:42:43.160845	37	2439	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4786		CRC Fail	-34	ACL, EDR-2: 2-DH5	20:42:43.295845	36	2438	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4823		CRC Fail	-30	ACL, EDR-2: 2-DH5	20:42:43.447096	55	2457	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4833		CRC Fail	-35	ACL, EDR-2: 2-DH5	20:42:43.497097	34	2436	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4839		CRC Fail	-41	ACL, EDR-2: 2-DH5	20:42:43.517097	19	2421	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4855		Packet reTXed	-37	AVDTP Media, SN 0x4E49, 5 frames SBC; 4 ReTXed packets	20:42:43.615848	51	2453	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4855		Packet reTXed	-36	ACL (ACL-U), EDR-2: 2-DH5, AVDTP Media SBC	20:42:43.615848	51	2453	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4857		Packet reTXed	-35	ACL (ACL-U), EDR-2: 2-DH5, AVDTP Media SBC	20:42:43.619598	63	2465	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4861		Packet reTXed	-41	ACL (ACL-U), EDR-2: 2-DH5, AVDTP Media SBC	20:42:43.623348	20	2422	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4863		Packet reTXed	-40	ACL (ACL-U), EDR-2: 2-DH5, AVDTP Media SBC	20:42:43.627098	22	2424	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4866		Packet reTXed	-39	AVDTP Media, SN 0x4E4A, 5 frames SBC; 4 ReTXed packets	20:42:43.634598	8	2410	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0
BR	4866		Packet reTXed	-42	ACL (ACL-U), EDR-2: 2-DH5, AVDTP Media SBC	20:42:43.634598	8	2410	24:41:8C:66:23:5C ➔ 88:D0:39:82:4A:E0

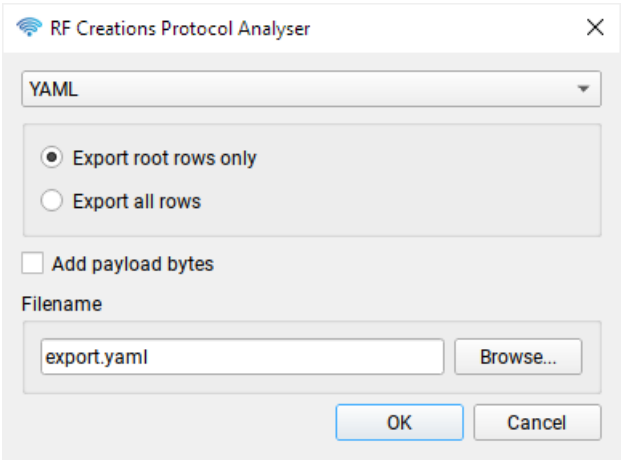
The “Filter Errors” popup can be used to select which errors to display. To select/deselect a whole error level, use the leftmost column; for error categories, the middle column, and for individual errors, the rightmost column.

11.9 Exporting Data

In the Details, Connections, and Devices Detail tabs, the contents of the tabs can be copied to the clipboard for pasting into a text editor or other application. The context menu allows copying either the whole tab, or individual subtrees.



Additionally, the complete contents of a Summary tab can be exported as either a CSV or YAML file using the “Export…” option in the context menu. When using an aggregation level other than “Baseband”, you have the option to export only the top-level, root rows. You can also choose to add the complete payload bytes (dewhitened, decrypted etc) to the export.



12 QUERIES

Bluespy supports various query strings, which can be applied to any event and are used to define the columns, for searching and in the API. The querys can all be represented as a string, but may also return an integer or a boolean. The following queries are supported:

CID	L2CAP Channel Identifier
LAP	Classic Lower Address Part
access_address	LE Access Address
access_code	Access Address or LAP
acked	Packet was acknowledged (bool)
central.addr	Bluetooth address of the central device
central.irk	Identity Resolving Key of the central device
central.localnames	Discovered names of the central device
central.vendor_names	Vendor(s) of the central device
central_packet_count	Packet number from central in this LE connection event
channel	RF channel of packet
connectable	Advertising Event/Set is connectable (bool)
crc	CRC of packet
crc_valid	CRC passed (bool)
cte	LE packet has constant tone extension (bool)
decrypted	Packet has been successfully decrypted (bool)
delta_ref	Time delta to last selected packet
device_summary	Information on sender and receiver
directed	LE Advertising is a directed type
duration	Packet time duration
emojis	A list of symbols concisely expressing essential packet stats
encrypted	The packet was encrypted
errors	Problems detected in this packet
event_counter	LE connection event number or Classic clock
frequency	RF frequency
handles	ATT handles
is_ID	Packet was a Classic ID packet (bool)
is_adv	Packet is LE advertising (bool)
is_data	Packet is LE data (bool)
is_empty	Packet has no payload (bool)
is_isoc	Packet is isochronous (bool)
is_test	LE Test packet (bool)
media_timestamp	AVDTP medai timestamp
modulation	PHY layer modulation scheme
nacked	Packet was negatively acknowledged
payload_hex	Packet payload in hexadecimal format
payload_length	Number of payload bytes (integer)
payload_raw	Payload as raw bytes
payload_summary	Beginning of payload

peripheral.addr	Bluetooth address of the peripheral device
peripheral.irk	Identity Resolving Key of the peripheral device
peripheral.localnames	Discovered names of the peripheral device
peripheral.vendor_names	Vendor(s) of the peripheral device
peripheral_packet_count	Packet number from peripheral in this LE connection event
pkt_no	Baseband packet index in the capture file
pkt_type	Baseband packet type
read_by_type_uuid	GATT read by type UUID
receiver.addr	Bluetooth address of the receiver device
receiver.irk	Identity Resolving Key of the receiver device
receiver.localnames	Discovered names of the receiver device
receiver.vendor_names	Vendor(s) of the receiver device
rejected	Packet was rejected by receiver (bool)
rssi	Received signal strength in dBm at Moreph
scannable	LE Advertising is scannable (bool)
sender.addr	Bluetooth address of the sender device
sender.irk	Identity Resolving Key of the sender device
sender.localnames	Discovered names of the sender device
sender.vendor_names	Vendor(s) of the sender device
service_uuid	GATT service UUID
ssrc	AVDTP SSRC
status	Packet status
summary	Packet summary
throughput	Byte throughput rate
time	Packet timestamp
type	Kind of packet

13 C API

The C API can be viewed in bluespy.h in the root of the installation. You will need to link against the libblueSPY shared library.

13.1 BLUESPY.H FILE REFERENCE

blueSPY C API

13.1.1 Classes

- struct bluespy_capture_options

13.1.2 Typedefs

- typedef enum bluespy_error **bluespy_error**
- typedef struct bluespy_capture_options **bluespy_capture_options**
- typedef uint64_t **bluespy_event_id**
  Indentifier for an packet or higher layer event. -1 means invalid.
  Only use bluespy_event_ids returned by the API.
  After bluespy_load_file or bluespy_capture all previous ids are invalid.

13.1.3 Enumerations

- enum **bluespy_error** : uint32_t { **BLUESPY_NO_ERROR** = 0, **BLUESPY_ERROR_NO_DEVICE**, **BLUESPY_ERROR_LICENCE**, **BLUESPY_ERROR_NO_FILE**, **BLUESPY_ERROR_CAPTURE_NOT_STARTED**, **BLUESPY_ERROR_INVALID_PACKET** }

13.1.4 Functions

- BLUESPY_API const char * bluespy_error_string (bluespy_error error)
  Get message for error.

- BLUESPY_API void **bluespy_init** ()
  Initialise bluespy, run once at start of program.

- BLUESPY_API void **bluespy_deinit** ()
  Clean up before program exits.

- BLUESPY_API void bluespy_start_gui ()
  Start a GUI instance.

- BLUESPY_API bluespy_error bluespy_connect (uint32_t serial=-1)
  Connect to Moreph.

- BLUESPY_API bluespy_error bluespy_disconnect ()
  Disconnect from Moreph.

- BLUESPY_API bluespy_error bluespy_capture (const char *filename, bluespy_capture_options opts)
  Start a capture.

- BLUESPY_API bluespy_error bluespy_stop_capture ()

Stop a capture.

- BLUESPY_API bluespy_error bluespy_load_file (const char *filename)
  Load a capture.

- BLUESPY_API bluespy_error bluespy_close_file ()
  Cloase current file.

- BLUESPY_API uint32_t bluespy_packet_count (void)
  Number of baseband packets loaded.

- BLUESPY_API bluespy_event_id bluespy_get_baseband (uint32_t index)
  Get a baseband packet.

- BLUESPY_API bluespy_event_id bluespy_get_parent (bluespy_event_id event)
  Get higher layer packets.

- BLUESPY_API const bluespy_event_id * bluespy_get_children (bluespy_event_id event, uint32_t *count)
  Get all lower layer packets.

- BLUESPY_API const char * bluespy_query (bluespy_event_id event, const char *query)
  Query a packet.

- BLUESPY_API int64_t bluespy_query_int (bluespy_event_id event, const char *query)
  Query a packet.

- BLUESPY_API bool bluespy_query_bool (bluespy_event_id event, const char *query)
  Query a packet.

- BLUESPY_API int bluespy_query_auto (bluespy_event_id event, const char *query, const char **s, int64_t *i, bool *b)
  Query a packet.

- BLUESPY_API void bluespy_add_link_key (const unsigned char *key, uint64_t addr0, uint64_t addr1)
  Add a link key for decryption.

13.1.5 Detailed Description

blueSPY C API

13.1.6 Class Documentation

13.1.6.1 struct bluespy_capture_options

13.1.6.1.1 Class Members:

bool	enable_15_4	
bool	enable_CL	
bool	enable_LE	
bool	enable_MHDT_CL	
bool	enable_MHDT_LE	
bool	enable_QHS	
bool	enable_wifi	
uint16_t	spectrum_period	Valid spectrum periods in microseconds are: 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 0 to disable spectrum_

13.1.7 Function Documentation

*13.1.7.1 BLUESPY_API void bluespy_add_link_key (const unsigned char * key, uint64_t addr0, uint64_t addr1)*

Add a link key for decryption.

13.1.7.1.1 Parameters

in	*key*	- 16 bytes of binary
in	*addr0*	- Bluetooth address of first device, set to 0 if unknown
in	*addr1*	- Bluetooth address of second device, set to 0 if unknown

The link key should be added before loading/capturing, or you should load again afterwards.

*13.1.7.2 BLUESPY_API bluespy_error bluespy_capture (const char * filename, bluespy_capture_options opts)*

Start a capture.

13.1.7.2.1 Parameters

in	*filename*	- UTF8 filename
in	*opts*	- Capture options

13.1.7.2.2 Returns

Error code

13.1.7.3 BLUESPY_API bluespy_error bluespy_close_file ()

Cloase current file.

13.1.7.3.1 Returns

Error code

13.1.7.4 BLUESPY_API bluespy_error bluespy_connect (uint32_t serial = −1)

Connect to Moreph.

13.1.7.4.1 Parameters

in	*serial*	- Serial number of the Moreph

13.1.7.4.2 Returns

Error code

Connect by serial number, or first found on USB if serial == -1

13.1.7.5 BLUESPY_API bluespy_error bluespy_disconnect ()

Disconnect from Moreph.

13.1.7.5.1 Returns

Error code

*13.1.7.6 BLUESPY_API const char * bluespy_error_string (bluespy_error error)*

Get message for error.

13.1.7.6.1 Parameters

in	*error*	

13.1.7.6.2 Returns

Internal pointer to null terminated string (do not free)

13.1.7.7 BLUESPY_API bluespy_event_id bluespy_get_baseband (uint32_t index)

Get a baseband packet.

13.1.7.7.1 Parameters

in	*index*	- 0 <= index < bluespy_packet_count()

13.1.7.7.2 Returns

Event ID

13.1.7.8 BLUESPY_API const bluespy_event_id * bluespy_get_children (bluespy_event_id event, uint32_t * count)

Get all lower layer packets.

13.1.7.8.1 Parameters

in	*event*	
out	*count*	- size of returned array

13.1.7.8.2 Returns

Event ID array

Returned child array is valid until next call, so take a copy

13.1.7.9 BLUESPY_API bluespy_event_id bluespy_get_parent (bluespy_event_id event)

Get higher layer packets.

13.1.7.9.1 Parameters

in	*event*	

13.1.7.9.2 Returns

Event ID

13.1.7.10 BLUESPY_API bluespy_error bluespy_load_file (const char * filename)

Load a capture.

13.1.7.10.1 Parameters

in	*filename*	- UTF8 filename

13.1.7.10.2 Returns

Error code

13.1.7.11 BLUESPY_API uint32_t bluespy_packet_count (void)

Number of baseband packets loaded.

13.1.7.11.1 Returns

N

13.1.7.12 BLUESPY_API const char * bluespy_query (bluespy_event_id event, const char * query)

Query a packet.

13.1.7.12.1 Parameters

in	*event*	
in	*query*	- query string to apply

13.1.7.12.2 Returns

String result of a query

Returned value is valid until next call, so take a copy

13.1.7.13 BLUESPY_API int bluespy_query_auto (bluespy_event_id event, const char * query, const char ** s, int64_t * i, bool * b)

Query a packet.

13.1.7.13.1 Parameters

in	*event*	
in	*query*	- query string to apply
out	*s*	- return if string
out	*i*	- return if int
out	*b*	- return if bool

13.1.7.13.2 Returns

0 = None, 1 = str, 2 = int, 3 = bool

Returned string value is valid until next call, so take a copy

13.1.7.14 BLUESPY_API bool bluespy_query_bool (bluespy_event_id event, const char * query)

Query a packet.

13.1.7.14.1 Parameters

in	*event*	
in	*query*	- query string to apply

13.1.7.14.2 Returns

Bool result of a query

13.1.7.15 BLUESPY_API int64_t bluespy_query_int (bluespy_event_id event, const char * query)

Query a packet.

13.1.7.15.1 Parameters

in	*event*	

in	*query*	- query string to apply

13.1.7.15.2 Returns

Integer result of a query

13.1.7.16 BLUESPY_API void bluespy_start_gui ()

Start a GUI instance.

The GUI exists in a background thread, this function returns immediately

13.1.7.17 BLUESPY_API bluespy_error bluespy_stop_capture ()

Stop a capture.

13.1.7.17.1 Returns

Error code

13.1.7.18

14 PYTHON API

The python API provides functions for connecting to a moreph and loading existing captures. It also provides a 'packets' object, which behaves like a list and can be used to access the packets in the current file. `len(packets)` shows the number of available packets, `packets[0]` accesses the first packet. The queries documented above can be accessed with attribute syntax, e.g. `packets[0].summary`

Examples:

To connect to a moreph with serial 00010100 and capture CL and LE:

```
import bluespy
from time import sleep

bluespy.connect(0x00010100)
bluespy.capture("example.pcapng", CL=True, LE=True)
sleep(20)
bluespy.stop_capture()
bluespy.disconnect()
print("Captured {} packets".format(len(bluespy.packets)))
bluespy.close_file()
```

To load an existing capture and print the summary strings of all packets:

```
import bluespy

bluespy.load_file("example.pcapng")
for p in bluespy.packets:
    print(p.summary)
bluespy.close_file()
```

14.1 BLUESPY.PY FILE REFERENCE

14.1.1 Classes

- class bluespy.error
  Return type showing why an operation failed.
- class bluespy.BluespyError

Exception showing why an operation failed.

- class bluespy.event_id
  An object referencing a loaded packet.
- class bluespy.Packets
  List-like object representing the currently loaded baseband packets.

14.1.2 bluespy.BluespyError Class Reference

Exception showing why an operation failed.

14.1.2.1 Public Member Functions

- def **get_error** (self)
  Return the underlying error object.

14.1.2.2 Detailed Description

Exception showing why an operation failed.

14.1.3 bluespy.error Class Reference

Return type showing why an operation failed.

14.1.3.1 Public Member Functions

- def **__str__** (self)
- def **__repr__** (self)
- def **__bool__** (self)

14.1.3.2 Public Attributes

- **value**

14.1.3.3 Detailed Description

Return type showing why an operation failed.

Evaluates to True if the operation succeeded, else False. str() and repr() give an error string, .value gives an error code.

14.1.4 bluespy.Packets Class Reference

List-like object representing the currently loaded baseband packets.

14.1.4.1 Public Member Functions

- def **__len__** (self)
  Current number of baseband packets captured.

- def __getitem__ (self, i)
  Get an event_id for a packet.

14.1.4.2 Detailed Description

List-like object representing the currently loaded baseband packets.

14.1.4.3 Member Function Documentation

14.1.4.3.1 def bluespy.Packets.__getitem__ (*self*, *i*)

Get an event_id for a packet.

14.1.4.3.1.1 Parameters

i	Index of packet, 0 <= i < **len** ()

14.1.4.3.1.2 Returns

: An event_id

14.1.5 bluespy.event_id Class Reference

An object referencing a loaded packet.

14.1.5.1 Public Member Functions

- def **__bool__** (self)
  Returns true if this is a valid packet.

- def parent (self)
  Get the a higher layer packet that contains this one.

- def children (self)
  Get all constituent packets of this packet.

- def query (self, name)
  Get a query from this packet, and return it in its preferred form.

- def **__getattr__** (self, name)
  Access queries as attributes.

- def query_str (self, name)
  Get a query from this packets, and return it as a string.

- def query_int (self, name)
  Get a query from this packets, and return it as an integer if possible.

- def query_bool (self, name)
  Get a query from this packets, and return it as a bool if possible.

14.1.5.2 Detailed Description

An object referencing a loaded packet.

Do not make your own, only get these from the 'packets' object. After running close_file(), do not call any methods on any existing event_id objects.

Any query (see the documentation or Help->Query List in the GUI) can be accessed as an attribute on this object

---

14.1.5.3 Member Function Documentation

14.1.5.3.1 def bluespy.event_id.children (*self*)

Get all constituent packets of this packet.

e.g. if this is an L2CAP packet

14.1.5.3.1.1 Returns

: List of event_ids

14.1.5.3.2 def bluespy.event_id.parent (*self*)

Get the a higher layer packet that contains this one.
e.g. if this is a baseband data packet get the L2CAP packet it is part of.

14.1.5.3.2.1 Returns

: event_id

14.1.5.3.3 def bluespy.event_id.query (*self*, *name*)

Get a query from this packet, and return it in its preferred form.

14.1.5.3.3.1 Parameters

name	A query string, e.g. "summary"

14.1.5.3.3.2 Returns

: A string, int or bool depending on the query

14.1.5.3.4 def bluespy.event_id.query_bool (*self*, *name*)

Get a query from this packets, and return it as a bool if possible.

14.1.5.3.4.1 Parameters

name	A query string, e.g. "acked"

14.1.5.3.4.2 Returns

: bool

14.1.5.3.5 def bluespy.event_id.query_int (*self*, *name*)

Get a query from this packets, and return it as an integer if possible.

14.1.5.3.5.1 Parameters

name	A query string, e.g. "summary"

14.1.5.3.5.2 Returns

: int

14.1.5.3.6 def bluespy.event_id.query_str (*self*, *name*)

Get a query from this packets, and return it as a string.

14.1.5.3.6.1 Parameters

name	A query string, e.g. "summary"

14.1.5.3.6.2 Returns

: string

14.1.6 Functions

- def bluespy.connect (serial=-1)
  Connect to Moreph hardware via USB or Ethernet.

- def bluespy.disconnect ()
  Disconnect from current Moreph.

- def bluespy.capture (filename, CL=False, LE=False, QHS=False, _15_4=False, wifi=False, MHDT_CL=False, MHDT_LE=False, spectrum=0)
  Start a new capture in filename.

- def bluespy.stop_capture ()
  Stop the current capture.

- def bluespy.load_file (filename)
  Load an existing capture.

- def bluespy.close_file ()
  Load an existing capture.

- def bluespy.add_link_key (key, addr0=0, addr1=0)
  Add a link key for decryption.

- def **bluespy.start_gui** ()
  Spawn an instance of the user interface.

14.1.7 Variables

- **bluespy.packets** = Packets()
- **bluespy.argtypes**
- **bluespy.restype**

14.1.8 def bluespy.add_link_key (*key*, *addr0* = 0, *addr1* = 0)

Add a link key for decryption.

14.1.8.1 Parameters

key	Link key as a 16-byte bytes object
addr0	(Optional) MAC address of central as a 64-bit integer
addr1	(Optional) MAC address of peripheral as a 64-bit integer

14.1.9 def bluespy.capture (*filename*, *CL* = `False`, *LE* = `False`, *QHS* = `False`, *_15_4* = `False`, *wifi* = `False`, *MHDT_CL* = `False`, *MHDT_LE* = `False`, *spectrum* = 0)

Start a new capture in filename.

14.1.9.1 Parameters

filename	Path to store capture in
CL	Enable Bluetooth classic capture
LE	Enable Bluetooth LE capture
QHS	Enable Qualcomm High Speed capture
_15_4	Enable 802.15.4 capture
wifi	Enable wifi capture
MHDT_CL	Enable MediaTek mHDT Classic capture
MHDT_LE	Enable MediaTek mHDT LE capture
spectrum	Spectrum capture interval in microseconds. 0 means disabled. Allowed values are: 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000.

14.1.9.2 Exceptions

BluespyError	Exception in the bluespy library

14.1.10 def bluespy.close_file ()

Load an existing capture.

14.1.10.1 Exceptions

BluespyError	Exception in the bluespy library

14.1.11 def bluespy.connect (*serial* = −1)

Connect to Moreph hardware via USB or Ethernet.

14.1.11.1 Parameters

serial	A serial number as an integer, or -1 to connect to the first USB device. The serial number shown in the software and on the MiniMoreph is hexadecimal, so should be entered as 0xNNNNNN. There is a serial number on the bottom of some Moreph30s is of the form AYYYY-XXXXX, the XXXXX is the required serial number in decimal.

14.1.11.2 Exceptions

BluespyError	Exception in the bluespy library

You should run disconnect() later if this is successful.

14.1.12 def bluespy.disconnect ()

Disconnect from current Moreph.

14.1.12.1 Exceptions

BluespyError	Exception in the bluespy library

14.1.13 def bluespy.load_file (*filename*)

Load an existing capture.

14.1.13.1 Exceptions

BluespyError	Exception in the bluespy library

14.1.14 def bluespy.stop_capture ()

Stop the current capture.

14.1.14.1 Exceptions

BluespyError	Exception in the bluespy library