

Your First Wideband Capture

How to Achieve the Perfect Capture

Introduction

Ellisys wideband sniffers are designed to be very easy to use. With zero configuration, captures can be initiated with a single click. The user can start a capture, connect the devices of interest, and immediately begin to understand a wide variety of performance and other behaviors, including conformance to design criteria, reliability aspects, errors, coexistence issues, etc.

With the wideband approach, all traffic will immediately be captured and displayed live, which is good, but this also presents an obvious need to understand how to drill down to isolate your devices of interest. To get the ideal capture, there are a few helpful things that you should know. This expert note will guide you through a few of the steps required to ensure that you can maximize the effectiveness of your Ellisys analyzer.

Good Things to Remember

The Ellisys wideband sniffer is designed to learn and retain important device parameters from the captured information or from information entered by the user, such as a link key. Information such as the BD_ADDR, friendly name, SDP parameters, L2CAP channels, link key, audio codecs, etc. are all necessary in order to display the information successfully.

The link key can be captured (over HCI), entered manually, or even injected programattically and will be saved for use in future connections. If your device uses (and transmits) an Identity Resolving Key (IRK), the same thing applies – the analyzer will capture it and the software will remember it.

HELPFUL HINT: In very busy environments, you may wish to deselect the "eye" icon on the toolbar of the Instant Piconet **(Show/Hide Broadcast Devices)** to hide broadcast traffic, to help isolate established (or establishing) piconets visually. Once you install your device filter, turn this back on so you can see broadcast events created by devices included in your filter.

Capture Process

1 Position

Position the analyzer and the devices in reasonable proximity. (See Expert Note, EEN_BT04 "Optimal Placement of your Analyzer" for details on optimal placement.)



Configure

2)

Configure the recording settings as needed (Record | Recording Options menu). This tells the analyzer what over-the-air traffic types and/or wired traffic types you want to capture, and controls other things, like radio sensitivity and a long-term capture mode.



REC 🕨

3 Record

Now you're ready to start the capture by simply selecting the Record button, located on the main toolbar.

Connect

4)

Connect your Bluetooth devices under test.

5) Stop

Once you've captured enough of the device traffic, simply select the Stop button to halt the capture. You can now drill down into the data using the many Ellisys software views. You can do this as recording is ongoing as well.



((•))



Save the trace for further analysis in the future. All devices in the area will be saved, but there is a method that allows you to save just the devices you're interested in (discussed later).





Filtering

There are many approaches to filtering information in the analyzer software. In the case of a wideband sniffer, learning to use these various filter mechanisms is key to drilling down quickly and efficiently to isolate to devices of interest, protocols of interest, packets of interest, etc.

The "biggest" filter is the device-based filter, one that shows or hides user-specified devices.

On your initial recording, you may see dozens or even hundreds of devices present, and odds are, your interests lie with a few of these, so installing a device filter is often a first step by many users. There are quite a few approaches to installing a device-based filter, including a right-click in the Instant Piconet view on the Piconet

desired, or on a communicating pair in the Communication column of an Overview, using the Device Traffic Filter dialog, and other approaches as described in more detail in the User Guide.

Populating the Devices Database Automatically

As mentioned above, the Ellisys wideband software will unobtrusively learn various details about devices from the captured traffic. The first piece of information needed by the Ellisys software is the BD_ADDR of the devices (Bluetooth device address). The BD_ADDR of one of two communicating devices is determined when a connection is captured (either paging or advertising, depending upon the use of classic BR/EDR or Low Energy, respectively), but the BD_ADDR of a connecting device cannot be known from the connection. An easy way to have all devices send out their BD_ADDR is by doing a discovery from a

HELPFUL HINT: The user can change the name of a device using the **Edit** button in the Device Traffic Filters dialog, accessible from the **Filter**: drop-down > **Configure**, located atop the GUI.

🖹 Save Filtered	Сору					×
Save in:	My Traces		~	3 🦸 📂 🛄	-	Information to keep:
Quick access Desktop Libraries This PC	Name	← ces 1.btt	Date modified 9/29/2020 11:	Type Ellisys Blue	Size 144,212 KB	Low Energy WIF HCI Injection (Primary) HCI Injection (Secondary) Message Log Spectrum
	File name:	Lots of Devices_f	itered btt	~	<u>S</u> ave	
	Save as type:	Ellisys Bluetooth 1	Trace Files (*.btt)	~	Cancel	

Figure 1 Saving a Filtered Copy.

HELPFUL HINT: Once a device filter is installed, the user can elect to save the existing trace into a new trace that includes only those devices included by the device filter. This feature is executed using **Save Filtered Copy**, located in the **File** menu. **See Figure 1**. In addition, the user can opt to remove certain capture components, such as raw spectrum information, Wi-Fi captures, HCI captures, etc. <u>This can significantly reduce file size</u> and makes sharing of files much easier (see the built-in cloud-based trace sharing, located in the **File** menu). The original trace is NOT replaced – a new trace is created from the original.

Bluetooth device. When a Bluetooth inquiry (BR/EDR) is sent, for example, all nearby devices will send FHS packets containing their BD_ADDR and other useful information. See **Figure 2** for a typical FHS packet's contents.

🖃 🌉 Baseband Packet		
🗆 🔩 Header		
LT_ADDR.	0	0x0
Packet Type	FHS	0x2
Flow	Go	0x1
ARQN	Nak	0x0
SEQN	1	0x1
🖃 🕂 User Payload		
LAP	1E:10:E6	0x1E10E6
Extended Inquiry Respon	No following Extended Inq	0x0
Scan Repetition (SR)	R0: scan interval <= 1.28s	0x0
UAP	0x76	0x76
NAP	00:02	0x0002
🖃 🔩 Class Of Device / Service		
Minor Device Class	Desktop Workstation	0x01
Major Device Class	Computer	0x01
🖃 🔩 Major Service Class		
Networking	Yes	0x1
Rendering	Yes	0x1
Capturing	Yes	0x1
Object Transfer	Yes	0x1
🧼 Audio	Yes	0x1
LT_ADDR	1	0x1
CLK[27-2]	0x382391D, as CLK[27-0]:	0x382391D

Figure 2 Typical FHS Packet Content.



item 🗸 🗸	Communication V	Status 🗸	Time v	p -
Raging (Mobile Nokia* 00: 1A:DC:66:CB:F4 > *AudioSource* 00: 1A:7D: 21:38:CD, responded, 824 ms)	Master: "Mobile Nokia" 00:1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.000 000 000	
8 9 LMP Version Exchange (Haster: 2.0 + Slave: 2.1)	Master: "Mobile Nokia" 00:1A:DC:66:C8:F4 <-> Slave: "Aud	OK .	0.826 248 250	
(8 9 LMP Features Exchange (29 Features > 34 Features)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.934 999 125	
(8 9 LMP Host Connection (Accepted)	Master: "Mobile Nokia" 00: 1A:DC:66:CB:F4 <-> Slave: "Aud	OK .	0.946 248 125	
is 🗤 LHP Setup Complete	Master: "Hobie Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.960 624 125	
8 4 LHP Set APH (0-79, 0x0106746, as CUX[27-0]: 0x0380F4C, APH enabled)	Master: "Mobile Nokia" 00: 1A:DC:66:CB:F4 <-> Slave: "Aud		0.963 748 125	
a ma LMP Auto Rate	Master: "Mobile Nokia" 00: 1A:DC:66:CB:F4 <-> Slave: "Aud		0.966 248 000	
B P Features Exchange (29 Features + 34 Features)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.969 998 000	
8 🍁 L2CAP Connection (Src=0x0040, PSH=SDP + Dist=0x0089)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK .	0.976 248 125	
13 WE LIMP Auto Rate	Master: "Mobile Nokia" 00: 1A:DC:66:CB:F4 <-> Slave: "Aud		0.978 124 000	
IMP Page Scan Mode (Mandatory scheme, R1 + Accepted)	Master: "Hobie Noka" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.980 623 875	
Here LMP Timing Accuracy Transaction (150 ppm, 3/tter=30 us)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	0.981874000	
8 10 UMP Features Exchange (34 Features > 29 Features)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	ox	1.016 873 500	
8 💠 L2CAP Configure (Dst=0x0089, HTU=65'535 + Drc=0x0040)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	1.022 499 125	
8 🛖 L2CAP Configure (Dst=0x0040, HTU=48 + Src=0x0089)	Master: "Hobie Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK .	1.041 875 125	
B A SDP Service Search Transaction (Hands-Free: 0x00030019, 0x0003001A)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	1.047 499 125	
B 🔑 SDP Service Attribute Transaction (0x00010019: Hands-Free Generic Audo L2CAP RFCOMM Ch 2)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK .	1.086 247 875	
B 2 SDP Service Attribute Transaction (0x0001001A: Hands-Free Generic Audio L2CAP RFCOMM Ch 1)	Master: "Hobie Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	ox	1.114 998 000	
🗄 💠 L2CAP Disconnection (Src=0x0040, Dist=0x0089)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	ox	1.143 747 875	
L2CAP Connection (Inc-0x0040, PSM-RFCOMM + Dat-0x008A)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	ox	1.159 997 875	
	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	ox	1.249 374 500	
	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	OK	1.315 623 750	
IMP Encryption Key Size (Accepted)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud	Error	1.321 247 750	
■ == LMP Start Encryption Request (48 ED 30 22 C8 66 88 C2 8C 3F 5E AF 63 AD 77 88)	Master: "Mobile Nokia" 00: 1A:DC:66:C8:F4 <-> Slave: "Aud		1.382 497 625	
<				>

Figure 3 Capture and Display of LMP Name.

Even better, most Bluetooth stacks determine the LMP name as well, so this will also be "learned" by the sniffer and used in several places throughout the software. See the Communication column in **Figure 3**.

When the full BD_ADDR of a device is not known by the analyzer prior to capturing the device's traffic, the sniffer can still partially determine the BD_ADDR, most of the time. In this case, the upper bytes will be indicated as missing with "xx" in the BD_ADDR, as shown in **Figure 4**. The traffic can still be captured successfully, but it will not be possible to decrypt the traffic on-the-fly if the BD_ADDR is not fully known, since this is one of the inputs to the security algorithms.

Populating the Device Database Manually

An alternative method to informing the analyzer software of a device address is to populate the Device Database manually. This can be done in the **Device Traffic Filters** dialog. To get there, select **View** from the main menu, then **Device Traffic Filters** (or select **Configure** from the dropdown menu on the main toolbar labeled **Device Filter**).

The New Device button (Figure 5) enables creation of a new device from scratch, including its BD_ADDR, its friendly name and associated color. Before creating a new device, it is useful to check if the device is not already in the database. The **Search** field is quite useful for this purpose. This dialog also enables the user to update the information of an existing device, which is useful especially to update a partial BD_ADDR (as discussed in the section above). It is also possible to delete an existing device if no longer needed.

HELPFUL HINT: Capturing the pairing process is **key**.

tem 🗸 🗸	Status 🗸	Payload V	Time v	Time_ v	Communication	Applic V
Paging (Unknown BD_ACOR > xxx:0C:8E:39, to response,	ox		408-453 085 625		Master: Unknown BD_ADDR <-> Slave: :00:00:8E:39	Baseband
Paging (Unknown BD_ADDR > txx:0C:8E:39, no response,	OK .		466.349 474 750	57.896	Master: Unknown BD_ADDR <-> Slave: :xx:0C:BE:39	Baseband
Paging (Unknown BD_ADDR > tox:0C:8E:39, to response,	OK		477.894 464 000	11.544	Master: Unknown BD_ADDR <-> Slave: :xxx00Ct8E:39	Baseband
🖡 🍽 Paging (Unknown 80_ADDR > txxt0Ct8Et39, no response,	OK		487.764 507 750	9.870 0	Master: Unknown 80_A00R <>> Save: 10x100/8E139	Baseband
R Paging (Unknown SD_ADDR > xxxxx:A4:4F:A6:D6, no resp	OK		545.000 775 125	57.236	Master: Unknown 8D_ADDR <>> Save: xxxxxA4r4F1A6:D6	Baseband
R Paging (Unknown BD_ADDR xxx:0C:8E:39, no response,	OK		595.709 117 125	50.708	Master: Unknown 8D_ADDR <>> Slave: :xx:0C:8E:39	Baseband
Paging (00: 10:08:36:2C:AF > 00:17/FA:03:90:83, respon	OK .		932.533 947 000	336.824	Master: 00:10:08:36:20:AF <-> Save: 00:17:FA:03:90:83	Baseband
8 2 AOL C Transfer	ox	1 byte (66)	932.552 695 875	0.018 7	Master: 00:10:08:36:20:AF <-> Slave: 00:17/FA:03:90:83	UP
8 4 ACL-C Transfer	OK .	6 bytes (48 03 0F 00 44 43)	932.554 573 750	0.0018	Master: 00:10:08:36:20:AF <-> Slave: 00:17:FA:03:90:83	UP
8 1 2 ACL-C Transfer	OK	6 bytes (40 03 09 00 82 5C)	932.557 696 000	0.003 1	Master: 00:10:08:36:20:AF <-> Slave: 00:17:FA:03:90:83	UIP
B 🚰 ACL-C Transfer	ox	9 bytes (4F FF FF 80 FE 98	932.559 573 750	0.0018	Master: 00: 10:08:36:2C:AF <-> Slave: 00: 17:FA:D3:90:83	UMP
R 2 ACL-C Transfer	OK	9 bytes (\$18C 02 04 38 00	932.562 695 875	0.003 1	Master: 00:10:08:36:20:AF <-> Save: 00:17:FA:03:90:83	UMP
B 4 ACL-C Transfer	OK	9 bytes (69 3F 03 83 90 D3	932.589 573 750	0.025 8	Master: 00:10:08:36:20:AF <-> Save: 00:17/FA:03:90:83	UP
B 4 AQ. C Transfer	ox	5 bytes (27 E4 98 PC 04)	932.590 823 750	0.0012	Masteri 00:10:08:36:20:AF <-> Slave: 00:17/FA:03:90:83	UP
B 🛫 🕈 ACL-C Transfer	OK	2 bytes (07 13)	932.595 196 000	0.004 3	Master: 00:10:08:36:20:AF <-> Save: 00:17/FA:03:90:83	UMP
PHS (00:17/FA:03:90:83) packet (ACL)	ox	20 bytes (58 43 3F 06 0C 76	932.740 198 500	0.145 0	Master: 00:10:08:36:20:AF <-> Save: 00:17/FA:03:90:83	Baseband
(ACL)	OK .		932.740 823 375	0.000 6	Master: 00:10:08:36:20:AF <-> Slave: 00:17/FA:03:90:83	Baseband
8 1 2 ACL-C Transfer	OK .	2 bytes (06 33)	932.744 779 500	0.003 9	Master: 00:17/FA:D3:90:83 <-> Slave: 00:10:08:36:20:AF	UP
8 : 2 ACL-C Transfer	OK	1 byte (62)	932.746 029 500	0.0012	Master: 00:17/FA:D3:90:83 <-> Slave: 00:10:08:36:20:AF	UIP
B : ACL-C Transfer	OK	1 byte (63)	932.747 904 250	0.0018	Master: 00:17/FA:03:90:83 <-> Slave: 00:10:08:36:20:AF	UP
B 2 ACL-C Transfer	ox	1 byte (SE)	932.749 779 500	0.0018	Master: 00:17/FA:03:90:83 <-> Slave: 00:10:08:36:20:AF	UMP
B ; 4 ACL-C Transfer	OK	1 byte (SP)	932.751 655 375	0.0018	Master: 00:17/FA:03:90:83 <-> Slave: 00:10:08:36:20:AF	UPP
B 2 ACL-C Transfer	OK	3 bytes (60 FA GA)	932.752 904 250	0.0012	Master: 00:17/FA:03:90:83 <-> Slave: 00:10:08:36:20:AF	UP
B 🚽 ACL-C Transfer	ox	3 bytes (61 FA GA)	932.753 529 500	0.000 6	Master: 00:17/FAID3/90:83 <-> Slave: 00:10:08:36:20:4F	UMP
B 2 ACL-C Transfer	OK .	1 byte (0A)	932.754 779 625	0.0012	Master: 00:17/FAID3:90:83 <-> Slave: 00:10:08:36:20:4F	UMP
ACL C Transfer	ox	3 hutes (NE-40 1E)	932,256,029,625	0.001 2	Marter: 00:17/54/03/90/83 cm Save: 00:10/08/36/20/85	LMP

Figure 4 When the BD_ADDR is Not Fully Known.

Learning the Next Pieces of Information

Once we have full BD_ADDRs of the devices, capturing the pairing procedure will then enable the Ellisys software to learn the missing pieces. During pairing and link establishment exchanges, the devices discover each other's capabilities and exchange the information that is useful in order for the sniffer to correctly decode the protocols, profiles, and services.

The pairing is also useful for determining the Link Key. In the case of a PIN-code based pairing, or with an SSP pairing in Debug Mode, the sniffer will automatically deduce the Link Key. In other cases, the Link Key needs to be entered into the Security pane (or if HCI is being captured, the software will automatically extract a link key should it be exchanged over that interface).

After these steps, all further connections involving these two devices will be decoded perfectly by the sniffer. The sniffer will remember all data necessary to display useful information, including the Link Key for decrypting the data.



Figure 5 Populating the Device Database Manually.



Different Approaches

The steps above are obviously just suggestions and various other approaches can be used. The most important thing to understand is that the device information mentioned above is required only in order to decrypt the data and decode it into various protocols. It is not required however for the capture itself, since a wideband sniffer is capable of capturing any Bluetooth packet without this information, even encrypted traffic.

Another important concept is that the Ellisys software learns information and then stores it in its local database, as well as in the capture file itself. If some information is missing at capture time, the trace might not be usable right away. However, the missing information may be updated at a later point, and older traces can be reopened successfully as soon as this information is learned by the software.

Let's take a simple example. We are capturing two completely new devices with the analyzer. These two devices are already paired and we don't want to re-pair. We also don't want to do an inquiry, so we start capturing the connection right away.

HELPFUL HINT: Once you have the BD_ADDR of one device, by performing a second capture, the device database will learn the BD_ADDR of the other device, adding this to your device database. In this case, the Ellisys sniffer will just know the BD_ADDR of the master device and nothing else, so it is not possible to decrypt the data. We save this capture.

We then do a second capture where the device that was the slave is now the master. At this point we know the BD_ADDRs of both devices and we can decrypt data when the link key is provided. Now that all information is known, we can reopen the first capture, which will be successfully decrypted and decoded as the required information has been learned by the software. The new information will be saved in this trace that now contains all of what is needed. It can thus be exchanged with a remote colleague who never had access to the actual devices.

Conclusion

In this Ellisys Expert Note we learned that a wideband sniffer captures all traffic sniffed as part of a typical capture. And, to achieve a more perfect capture, we explored and learned new methods to populate the devices database both automatically and manually, to save files more efficiently by using device-based filters, and how the analyzer captures and stores critical elements linke link keys, IRKs, codecs, etc.

Visit ellisys.com or email support@ellisys.com for more information.

Other Interesting Reading

- EEN_BT06 Bluetooth Security Truths and Fictions
- EEN_BT07 Secure Simple Pairing Explained

More Ellisys Expert Notes available at: www.ellisys.com/technology/expert_notes.php

Feedback

Feedback on our Expert Notes is always appreciated. To provide comments or critiques of any kind on this paper, please feel free to contact us at <u>expert@ellisys.com</u>

	ן וו
イー	4

Sales Contact:



USA: +1.866.724.9185 Asia: +852 2272 2626

Europe: +41 22 777 77 89









Copyright© 2021 Ellisys. All rights reserved. Ellisys, the Ellisys logo, Better Analysis, Bluetooth Explorer, Bluetooth Tracker, Bluetooth Vanguard, Ellisys Grid, and Bluetooth Qualifier are trademarks of Ellisys, and may be registered in some jurisdictions. The Bluetooth® word mark and logos are registered trademarks owned by the Bluetooth SIG, Inc. and any use of such marks by Ellisys is under license. Wi-Fi® and the Wi-Fi Alliance logo are trademarks of Wi-Fi Alliance. Other trademarks and trade names are those of their respective owners. Information contained herein is for illustrative purposes and is not intended in any way to be used as a design reference. Readers should refer to the latest technical specifications for specific design guidance.