DFIR Review

How to Use iOS Bluetooth Connections to Solve Crimes Faster

Heather Mahalik, Matt Goeckel

Published on: Nov 16, 2020

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)

Synopsis

Forensics Question:

We aim to educate those working iOS investigations on the primary files that are used to track Bluetooth connectivity and to provide a deeper understanding of the timestamp formats.

OS Version:

iOS 11-13 were used for the testing and validation in this blog

Tools:

Cellebrite Physical Analyzer
checkra1n
Cellebrite UFED checkm8
DCode
Sanderson Date Decoder

Here's how to unlock critical evidence from Apple devices using Bluetooth connection data.

Bluetooth connections are often a factor in many investigations and can cover a wide range of case types from accident investigations to cases involving proximity to locations. Proving whether a driver was distracted before a fatal accident occurred is a common request. Were they really connected to Bluetooth? How can you be sure? What about a “seen” Bluetooth device? Can you leverage that to put a suspect in an approximate location at a point in time? Yes, you can. But only if you really understand the data.

This blog is going to cover Bluetooth for iOS devices. Part II will cover Bluetooth for Android devices. This research does not include “spoofing” Bluetooth and involves real or detected Bluetooth devices that could have been paired to your smartphone.

Matt and I have spent a good amount of time independently researching Bluetooth connectivity on iOS devices over the last few years. Oddly enough, our research stemmed from a question asked by an IACIS colleague of ours. (We did not work for Cellebrite at this time and didn't even know one another, yet we each helped.) This is another reason why sharing your research is important! Matt and I could have worked on this together in 2018, but we weren't aware we were both researching the same thing, which is a common theme in DFIR.

In this scenario, the detective needed to know how to determine the correct time from an iOS plist. The timestamp was key to his investigation. Sounds simple, right? Not quite.

We assumed the timestamp would be in UTC – most database and plist timestamps are. That was not the case for this file, however. More details on this will be provided later in the blog. For now, let's see where most Bluetooth investigations begin. For these examples, we used Josh Hickman's checkm8 extraction¹.

Files of interest (from the test device)

DarArchive/root/private/var/containers/Shared/SystemGroup/<GUID>/Library/Database/com.apple.MobileBluetooth.ledevices.other.db

DarArchive/root/private/var/containers/Shared/SystemGroup/<GUID>/Library/Database/com.apple.MobileBluetooth.ledevices.paired.db

DarArchive/root/private/var/containers/Shared/SystemGroup/<GUID>/Library/Preferences/com.apple.MobileBluetooth.devices.plist

Note: WAL files should also be parsed by your forensic tool.

What is stored in each file

com.apple.MobileBluetooth.ledevices.other.db: This database tracks *low energy* devices the iOS device *detected* or came into range with. Low energy includes devices that stay in a "sleep mode" and wake up when connected. They operate on a different band from normal Bluetooth. These often include exercise devices (fitbits, etc.), headphones and more.

com.apple.MobileBluetooth.ledevices.paired.db: This database tracks *low energy* devices the iOS device paired with.

com.apple.MobileBluetooth.devices.plist: This plist tracks paired devices and last-detected times. This is one of the most important files that requires attention for pairing!

Diving into the data

Let's look at the file system of the iOS device and the files of interest for Bluetooth artifacts.

```
7.....-.....:4~...) .ce7M2v...  
.....:4~...) .....s...../  
...K.....d.....}.....I.....b.....  
{.....~.1....}...Q...8.j.....8....  
.....0.....6.T...0.....  
.....
```

```
.....U.5.U#.=.....E1DF27E2-DA99-  
A6D7-78F8-5CAEFB839F03[TV] Spanky  
Public 64:1C:B0:AE:6C:B3.\WJ...U.  
.=.....00ECDB81-308C-AEAD-FDCB-E  
1E26035DFE7Public FC:F1:36:20:22:  
76.[.a.*.U;. =.....7A272D6E-9BA9-  
057F-8105-74AA4CF9A137This Is...s  
AirPods ProPublic 38:EC:0D:E2:49  
:CF.ZZJ.'.U..=.....C9C3EBC4-632C  
-C716-BEC9-CEB370144E9DPublic F0:  
18:98:86:3F:52.I~J.:.U..=.....42  
004563-1678-4B78-98FD-EBCBED0E1DB  
5Public D0:03:4B:35:34:64.^..J.F.U  
..=.....1F1C7282-DEB9-E5D7-B209-  
5177E1B3B61EPublic AC:BC:32:75:4D  
:1A.0.J.E.U..=.....1E8741B3-D89B  
-DB06-B93E-9016BB751217Public D0:  
03:4B:04:A7:1D.0iJ.@.U..=.....E5  
D0E147~763B-E88E-D82B-8353B027F1C  
4Public 50:DE:06:6D:02:F0.^..J.A.U
```

Physical Analyzer File System View from Joshi's iPhone

You will notice tools parse Bluetooth data from iOS devices and show connected devices. Make sure you *do not* confuse “detected” with “connected” as these are two vastly different things.

In [Physical Analyzer](#) we put Bluetooth in one location and declare the connection type under the column labeled “**Connectivity nature.**” The Bluetooth results are in the Analyzed Data model under Devices & Networks > Device Connectivity > Bluetooth. Detected devices can be used for proximity to Bluetooth locations (think about investigations involving a suspect being at a specific location).

					Device Name	Device type	Timestamp	Connectivity method	Device Identifiers	Connectivity nature
		☒	1		This Is's Apple Watch	Unknown		Bluetooth	Address Random 6...	Paired
		☒	2		This Is's AirPods Pro	Unknown		Bluetooth	Address Public 38:E...	Detected
		☒	3		Reverie	Unknown		Bluetooth	Address Public 00:0...	Detected
		☒	4		PR BT FD84	Unknown		Bluetooth	Address Public 00:1...	Detected
		☒	5		Office	Unknown		Bluetooth	Address Public D4:...	Detected
		☒	6		LG webOS TV	Unknown		Bluetooth	Address Public A0:...	Detected
		☒	7		L307LXK	Unknown		Bluetooth	Address Public 78:9...	Detected
		☒	8		Josh's AirPods	Headset	3/30/2020 9:14:06 AM	Bluetooth	Address 7C:04:D0:8...	Detected
		☒	9		Josh's AirPods	Headset		Bluetooth	Address 7C:04:D0:8...	Paired
		☒	10		Josh's AirPods	Unknown		Bluetooth	Address Public 7C:0...	Detected
		☒	11		Hue Lamp	Unknown		Bluetooth	Address Random E...	Paired
		☒	12		Hue Lamp	Unknown		Bluetooth	Address Random E...	Paired
		☒	13		ENVY Photo 6200 series	Unknown		Bluetooth	Address Public 82:C...	Detected
		☒	14		Charge 3	Unknown		Bluetooth	Address Public C4:...	Paired
		☒	15		AugPHP4	Unknown		Bluetooth	Address Public...	Detected

Physical Analyzer Parsed Bluetooth from Josh's iPhone

Keep in mind that a device can be detected and then later paired to your iOS device. Above, you can see that Josh's AirPods were “Paired” (Line 9) and the last time they were “Detected” was on 3/30/2020 (Line 8). This is useful to the investigator because it indicates the disconnect time for the device (when pairing stopped). In short, Physical Analyzer will show two entries for a paired Bluetooth device—one that shows it was paired, and another that indicates the last time the paired device was detected.

>> Device Connectivity Translate Go to ▾

Device Name: Josh's AirPods

Device type: Headset

Timestamp: 3/30/2020 9:14:06 AM

Connectivity method: Bluetooth

Connectivity nature: Detected

Artifact Family:

Source Repository Path:

Source:

Source file: [DarArchive/root/private/var/containers/Shared/SystemGroup/9140AD4D-45D5-49D5-8A8-1CD264CF295D/Library/Preferences/com.apple.MobileBluetooth.devices.plist:0x3D1 \(Size: 7312 bytes\)](#)

Device Identifiers

Address 7C:04:D0:89:89:A0

Physical Analyzer Detected Bluetooth from Josh's iPhone

Let's take a closer look at the plist that tracks paired devices. The `com.apple.MobileBluetooth.devices.plist` tracks all Bluetooth device pairing – not just *low energy* devices. This is where vehicles come into play.

If you needed to prove a person was driving hands free when a call or text was sent, this file would be a great place to start. However, the timestamp is tricky. Apple is notorious for storing timestamps in UTC, so why would this be any different, right? Well, it is different. An example from Josh Hickman's iOS device is shown below (this way you can validate what we are showing you).

#	Offset	Length	Value	Source
1	0x28	0x11	DeviceConnectivity.KeyValueModel.Value: 7C:04:D0:8...	/root/private/var/containers/Shared/SystemGroup/9140AD4D-45D5-49D5-8AA8-1CD264CF295D/Library/Preferences/com.apple...
2	0x3D1	0x1C	DeviceConnectivity.DeviceName: Josh's AirPods	/root/private/var/containers/Shared/SystemGroup/9140AD4D-45D5-49D5-8AA8-1CD264CF295D/Library/Preferences/com.apple...
3	0x428	0xA	DeviceConnectivity.DeviceType: Headset	/root/private/var/containers/Shared/SystemGroup/9140AD4D-45D5-49D5-8AA8-1CD264CF295D/Library/Preferences/com.apple...
4	0x43F	0x4	DeviceConnectivity.StartTime: 3/30/2020 9:14:06 AM	/root/private/var/containers/Shared/SystemGroup/9140AD4D-45D5-49D5-8AA8-1CD264CF295D/Library/Preferences/com.apple...

Physical Analyzer Hex View of com.apple.MobileBluetooth.devices.plist from Josh's iPhone

Notice the date/timestamp? This timestamp for when connectivity started is stored in UTC with an adjustment to the user's "base" time zone. This may not make sense when looking at Josh's iOS data because he connected in his "base" time zone. This is the key thing that helped that detective we previously mentioned.

To further explain this, Heather tested some Bluetooth connections while in Paris in March of 2018 (her initial research). Even though her AirPods (Hank's AirPods) were detected and paired in Paris, it did not reflect in the Europe/Paris time zone (UTC +1).

Below we can see the data parsed in Physical Analyzer, which is using UTC+0 as the time setting. The date shows March 16, 2018 at 06:22 UTC.

#	Device Name	Device type
1	Hank's AirPods	Headset
2	Sesame 2	Unknown
3		Unknown
4	UP24	Unknown

Device Connectivity

Device Name: Hank's AirPods

Device type: Headset

Timestamp: 3/16/2018 06:22

Connectivity method: Bluetooth

Connectivity nature: Detected

Artifact Family:

Source Repository Path:

Source:

Source file: Heather Mahalik's iPhone/containers/Shared/SystemGroup/systemgroup.com.apple.bluetooth/Library/Preferences/com.apple.MobileBluetooth.devices.plist:0x22D (Size: 3022 bytes)

Device Identifiers

Address 7C:04:D0:A6:EE:82

Physical Analyzer Bluetooth Parsing of Heather's Test iPhone

The "LastSeenTime" value is stored in UTC but with the time zone adjusted for America/New York – Heather's base time zone (UTC-4:00). If you weren't aware of this tricky way Apple is storing this data, your entire investigation could be thrown off!

The screenshot shows the Physical Analyzer File Format Viewer interface. The 'File format viewer' tab is active, displaying a JSON-like structure of Bluetooth device information. The device name is 'Hank's AirPods'. The 'LastSeenTime' is highlighted with a red box and contains the value '1521181350'. Other fields include 'ServiceGaming', 'DefaultName', 'ServiceA2DP', 'ServiceAAC', 'ServiceRemote', 'ServiceBraille', 'ServiceNetSharingUser', 'SdpCacheSize', 'DeviceClass', 'Name', 'ServiceWiAP', 'ServiceWirelessCarPlay', 'ServicePhoneBook', 'LastAVRCPVersion', 'SdpCacheVersion', 'DeviceIdVersion', 'DeviceIdVendor', and 'DeviceIdVendorSource'.

Physical Analyzer File Format Viewer of com.apple.MobileBluetooth.devices.plist from Heather's Test iPhone

The screenshot shows the DCode v5.0 Time Decoding interface. A table lists various time formats and their corresponding timestamps. The 'Unix Seconds (UTC)' row is highlighted with a red box. To the right, the 'Value Input' section shows the 'Format' set to 'Numeric' and the 'Value' set to '1521181350'. A yellow callout bubble points to the 'Unix Seconds (UTC)' row with the text 'Shows EDT even though I was in Paris!'.

Name	Timestamp
Apple HFS+	1952-03-15 01:22:30.00000000 -05:00
Google Chrome (UTC)	1601-01-01 00:25:21.1813500 Z
Google Chrome	1600-12-31 19:25:21.1813500 -05:00
GPS Time (UTC)	2028-03-20 06:22:30.00000000 Z
GPS Time	2028-03-20 02:22:30.00000000 -04:00
Microsoft Ticks (Local)	0001-01-01 00:02:32.1181350
Nokia Series 30 (UTC)	2098-03-16 06:22:30.00000000 Z
Nokia Series 30	2098-03-16 02:22:30.00000000 -04:00
Unix Seconds (UTC)	2018-03-16 06:22:30.00000000 Z
Unix Seconds	2018-03-16 02:22:30.00000000 -04:00
Unix Milliseconds (Java Time) (UTC)	1970-01-18 14:33:01.35000000 Z

DCode being used for validation from Heather's Test iPhone

To further validate this, Matt did some testing on his phone as well. The results are confirmed – *com.apple.MobileBluetooth.devices.plist* stores the timestamp according to the phone's own local time.

```
▲ A0:56:B2:4C:B3:73 : dict = {
    DefaultName : AsciiString = Handsfree
    LastHandsfreeVersion : data = 06 01
    ServiceA2DP : AsciiString = Unknown
    PhonebookSyncSettings : integer = 31
    LastAVRCPControllerSupportedFeatures : data = 41 00
    ServiceAACP : AsciiString = Unknown
    ServiceRemote : AsciiString = Unknown
    ServiceNetSharingUser : AsciiString = Unknown
    ServiceBraille : AsciiString = Unsupported
    LastSeenTime : integer = 1582224378
    Name : AsciiString = Subaru BT
    DeviceClass : data = 08 04 76 00
    ServiceWiAP : AsciiString = Unknown
    ServiceWirelessCarPlay : AsciiString = Unknown
    ServicePhoneBook : AsciiString = Unsupported
    EncryptionKeySize : data = 10
    LastAVRCPControllerVersion : data = 06 01
    PhonebookSyncGroup : integer = -1
    LastHandsfreeSupportedFeatures : data = 2F 00
    ServiceGATT : AsciiString = Unknown
    initiateSDPMirroringState : boolean = False
```

Physical Analyzer File Format Viewer of com.apple.MobileBluetooth.devices.plist from

Matt's iPhone

In the image above, you can see Matt's Subaru Bluetooth connection. The *LastSeenTime* stored here translates to 2/20/2020 at 6:46 PM. Matt was at the car dealership on the evening of February 20th trading in his Subaru for a Jeep. The 6:46 PM timestamp is accurate for the last time (locally) Matt disconnected from the Subaru. To reiterate: This value is stored according to local time *not* UTC, and the value *LastSeenTime* will reflect the time of disconnection from the Bluetooth device.

Original	date	format
1582224378	2/20/2020 6:46:18 PM	Unix seconds

Timestamp decoded using Sanderson's Date Decoder v1.2.1

Matt also tested this with his laptop - where he could actively notate the connect and disconnect times. In a rare display of vulnerability, I (Matt) am sharing a snippet of my handwritten notes relating to this specific test. As you can see below, the times I recorded in my testing match the times stored in the plist.

Matt's handwritten notes displaying the computer disconnect time of 10:28 AM on 3/24 matches the time stored in the plist...

```
ServiceRemote : AsciiString = Unknown  
ServiceBraille : AsciiString = Unsupported  
ServiceNetSharingUser : AsciiString = Unknown  
LastSeenTime : integer = 1585045717  
Name : AsciiString = DESKTOP-23HR3VJ
```

...The screenshot from the same plist highlighting the computer connection time...

Original	date	format
1585045717	3/24/2020 10:28:37 AM	Unix seconds

...And that same time decoded via Sanderson's Tool.

Can you imagine the confusion this was causing? Why does this file rely upon your “base” or home time zone even if you connect in another part of the world? We do not have an answer to this question, but it's pretty cool when you uncover artifacts like this and share it back to the community.

In summary, we know that it's not always easy to put the pieces together during investigations. That is why it's important to share the research you have completed. During our Podcast “Carved From Unallocated,” we discuss why test data is important.

Josh Hickman provided test data and documentation to support it. Download his images and use it to validate your tools. If you're brave enough, extract your own iPhone and explore these files – they are available in a normal extraction (you don't need a full-file system). At Cellebrite, our goal is to make your life easier and educate you along the way.

References:

[1] Josh's checkm8 extraction:

https://drive.google.com/file/d/1GQP_y1340LHcq8eiDOYKAma62Q8GEyN5/view?usp=sharing

[2] Josh's checkra1n extraction: <https://thebinaryhick.blog/2020/04/16/ios-13-images-images-now-available/>

DFIR Review

The purpose of the research was clear and the paper was easily readable. The authors utilized the Hickman datasets, which is helpful because the reference material is available to everyone. The documentation of the files of interest from the test device along with the description of what is stored in each file was helpful. One of the reviewers indicated that there is additional commentary which is not necessary.

Future Work

It could be helpful to track all known Bluetooth artifacts (e.g. KnowledgeC and other system traces) from the start to the finish of a connection so investigators know where else to look if one particular data source isn't consistent with expectations.

It is not immediately clear why certain paired devices (e.g. AirPods) only appear in other.db while others (e.g. Apple Watch) are listed in paired.db. One possible hypothesis is that if the device pairing is recorded in the com.apple.MobileBluetooth.devices.plist file, it will not be recorded in the paired.db file. Additional research on these files would be valuable.

It would be useful to understand how the files are affected if the user chooses to forget the Bluetooth device. Research into the values in the LastSeenTime and LastConnectedTime fields would also be valuable.

Additional research surrounding the time zone might also prove to be useful. For example, with the AirPods connecting in Paris, did the device automatically change to the time zone that it was in or did it remain in its "base" time zone? If the device automatically corrected to the Europe/Paris time zone (UTC +1:00) and it still recorded the time stamp as America/New York (UTC -4:00), was the behavior documented more than once?

Reviewers

Eric Eppley (Methodology Review)

Nickolas Ligman (Methodology Review)

Doug Metz (Methodology Review)

Linda Shou (Methodology Review and Verified Review using Author Provided Datasets)