DFIR Review

iOS KnowledgeC.db Notifications

Scott Koenig

Published on: Nov 03, 2022

URL: https://dfir.pubpub.org/pub/g2v1z97i

Synopsis

Forensics Question: What are the different types of notifications we will have from the KnowledgeC.db and what do they mean? Can we determine if the user interacted with device after a notification was received and displayed on an iPhone?	
OS Version: iOS 14.7.1 (18G82) iOS 14.4.2 (18D70)	
Tools: Cellebrite UFED 4PC 7.47.0.247 Cellebrite Physical Analyzer 7.48.1.3 – Does not decode KnowledgeC.db /notification/usage Magnet AXIOM 5.4.0.26185 ArtEx 2.0.0.4 iLEAPP 1.9.4 – Does not decode KnowledgeC.db /notification/usage APOLLO 1.4	

Introduction

Cell phone use is routine. Our cell phones are really an extension of ourselves. We carry them around not only to make calls and messages, but they are also our daily planners, to-do lists, and entertainment resources. We use them at all times of the day – the alarms in the morning, email and social media all day, listening to music, and even reading books at night in bed. They can be a distraction, but does that stop us from checking them all day, especially when a notification pops up? Sometimes we just look to see what the notification is and move

on with our business. Sometimes, a notification needs to be handled right away. How do iPhones, or at least those running iOS 14, store notifications, and what happened with those notifications?

While using some commercial and some free forensic tools, I noticed very few of them decode the ***KnowledgeC.db/notification/usage*** data. The ones that do provide very little information about what the notification types mean.

Thanks to Sarah Edwards and several others who previously researched the ***KnowledgeC.db,*** we know it to be a great artifact. It can be used to determine a lot of device activities and a user's pattern of life, but can we use that data to determine if a user interacted with the device after it received a notification?

Based on previous research and publications, in conjunction this research, I believe not only can we determine if a user interacted with the device after receiving a notification, but I also believe we can determine how and when that interaction occurred.

Artifact Location:

- ***private\var\mobile\Library\CoreDuet\Knowledge***

The notification data I will be discussing is stored in the ***KnowledgeC.db ZOBJECTS*** table and***ZSTRUCTUREDMETADATA*** table**.**

The data I will be discussing in detail is:

- ***ZSTREAMNAME*** = ***/notification/usage***
- ***ZVALUESTRING*** = The notification types, which are listed below
  - ***Clear***
  - ***DefaultAction***
  - ***Dismiss***
  - ***Hidden***
  - ***IndirectClear***
  - ***Orb***
  - ***Receive***

· ***Z_DKNOTIFICATIONUSAGEMETADATAKEY__BUNDLEID*** = the bundle or application for which the notification is related. In the database, the bundle ID is only listed with a ***Receive*** notification type.

· ***Z_DKNOTIFICATIONUSAGEMETADATAKEY__IDENTIFIER*** = semi-unique identifier that can be used to link different notification types.

Note: I mention the ***Z_DKNOTIFICATIONUSAGEMETADATAKEY__IDENTIFIER*** as a semi-unique identifier because in some cases, like the Do Not Disturb notifications, the identifier repeats itself, but we can still use this to link the notification types together while analyzing the data.

Here is a link to GitHub for a SQLite query that might assist with analyzing the database

Device Settings:

When using this data in a forensic analysis, be sure to check the device notification settings. During testing, all applications tested had all notifications turned ON. These are the settings a user can change that could restrict the notification types you might encounter during an analysis. Figure 1 shows the settings menu for the Apple Messenger Application (com.apple.MobileSMS). Please check out the resources section to review additional research.

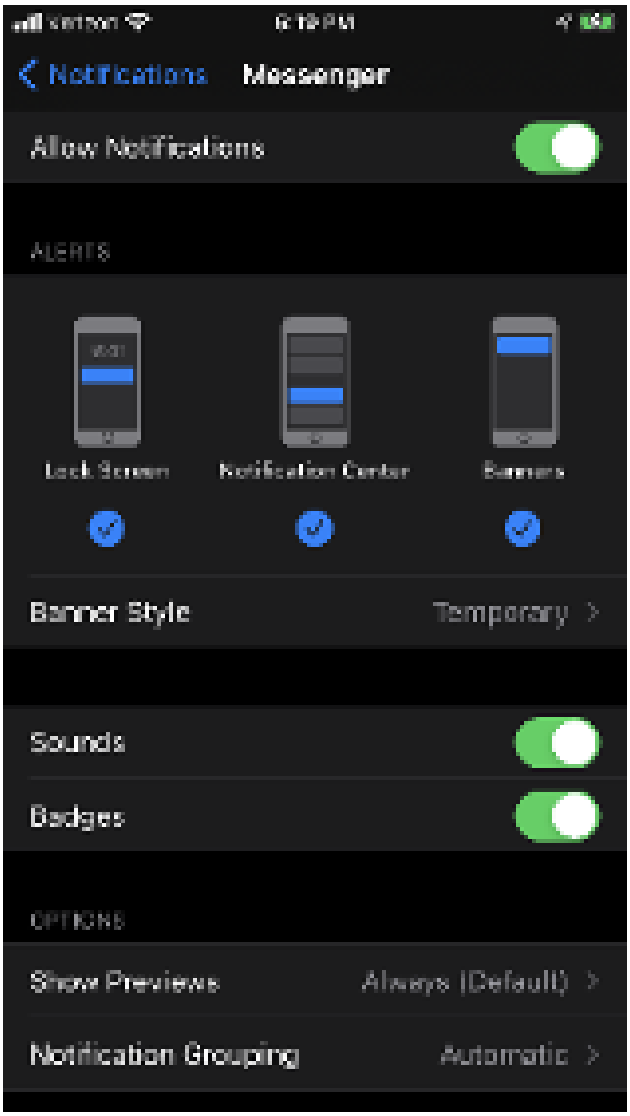

Figure 1

Research and Testing:

The following sections will demonstrate how I was able to determine each notification type and how I recreated them in testing.

Note: During testing, Magnet AXIOM, ArtEx, and APOLLO parsed the ***KnowledgeC.db*** notifications. Cellebrite Physical Analyzer and iLEAPP did not. iLEAPP had a section for iOS notifications, but the data was

being parsed from***DeliveredNotifications.plist,*** not the notifications from ***KnowledgeC.db,*** review the resources for additional information.

During testing, the test device was connected to ArtEx via ArtExtraction – Live Connection. This allowed me to run multiple tests, and I did not have to repeatedly acquire full file system dumps. The acquisition methods and tools listed above were used to validate what was being displayed in ArtEx.

Note: If you are a curious how to perform your own testing using ArtEx, here is a link to a recorded session of Cellebrite's Ctrl + Alt + Del where ArtEx creator Ian Whiffin discusses how to use the ArtEx Live Connection to conduct research: https://www.cellebrite.com/en/using-artifact-examiner-artex-to-investigate-an-artifact-on-a-device/

Receive Notification Type:

A ***Receive*** notification type is when a notification is received and displayed on the device. Depending on user interaction and device status the notification could be viewed from the springboard, the Lock Screen and/or the Notification Center.

Figure 2 has an example of a ***Receive*** notification type in ArtEx. During testing, this was created by sending the test device a text message (SMS). The test device screen came on and displayed the notification. After a few seconds, the screen automatically turned OFF and went dark. I did not touch or interact with the device or the screen.

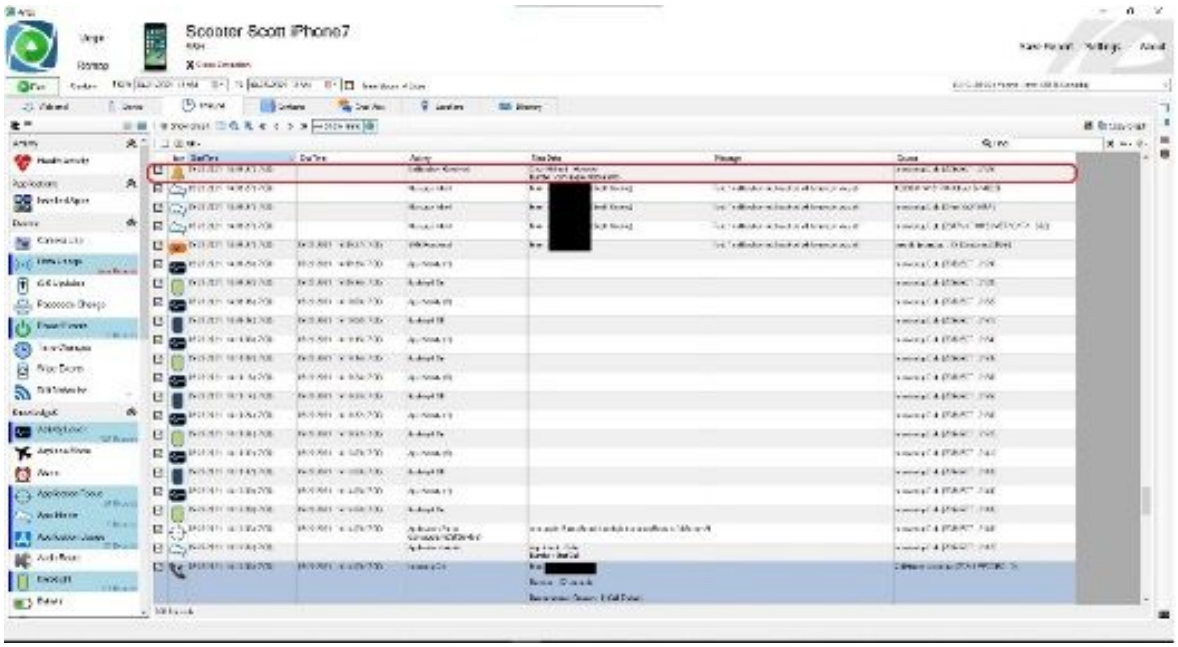

Figure 2

I turned the screen ON and OFF several times, using side button. During that time, I captured a screenshot of what the notification looked like on the device, seen in Figure 3. This did not affect or change the notification as it remained visible on the Lock Screen.

Figure 3

At 2:14 PM, I made a phone call to the test device, which was unanswered. When the phone call was received by the device and the InCallService application was brought into focus. A ***Receive*** notification type was created, seen in Figure 4 and Figure 5.

Icon	Start Time	End Time	Activity	MetaData	Message	Source
	09-23-2021 14:14:00 (-7:00)	09-23-2021 14:14:20 (-7:00)	Backlight Off			knowledgeC.db [ZOBJECT : 2151]
	09-23-2021 14:14:20 (-7:00)	09-23-2021 14:14:32 (-7:00)	App Activity (1)			knowledgeC.db [ZOBJECT : 2158]
	09-23-2021 14:14:20 (-7:00)	09-23-2021 14:14:32 (-7:00)	Backlight On			knowledgeC.db [ZOBJECT : 2157]
	09-23-2021 14:14:22 (-7:00)	09-23-2021 14:14:23 (-7:00)	Application Focus (com.apple.InCallService)	com.apple.SpringBoard.backlight.transitionReason.fullScreenAlert		knowledgeC.db [ZOBJECT : 2154]
	09-23-2021 14:14:23 (-7:00)		Application Intents	App Intent : Calls Bundle : StartCall		knowledgeC.db [ZOBJECT : 2155]
	09-23-2021 14:14:23 (-7:00)		Notification Received	Clear Method : Receive Bundle : com.apple.mobilephone		knowledgeC.db [ZOBJECT : 2156]
	09-23-2021 14:14:23 (-7:00)	09-23-2021 14:14:23 (-7:00)	Incoming Call - Unanswered	From : [redacted] Duration : 0 seconds Disconnection Reason : 6 (Self Rejected)		CallHistory.storedata (ZCALLRECORD : 3)
	09-23-2021 14:14:32 (-7:00)	09-23-2021 15:34:48 (-7:00)	App Activity (0)			knowledgeC.db [ZOBJECT : 2186]
	09-23-2021 14:14:32 (-7:00)	09-23-2021 15:34:44 (-7:00)	Backlight Off			knowledgeC.db [ZOBJECT : 2184]

Figure 4

Figure 5

At 3:35 PM, the test device received a phone call from an unknown source. The phone call was unanswered. There was not any user interaction with the screen. After the phone stopped ringing, the screen turned OFF and went dark. A ***Receive*** notification type was recorded via the ***KnowledgeC.db***, seen in Figure 6 and Figure 7. You will notice a ***Receive*** notification type for the voicemail which followed the unanswered phone call.

Note: During testing, when I answered or declined an incoming phone call, a ***Receive*** notification type would not be logged in the ***KnowledgeC.db***. A ***Receive*** notification type would be logged in the ***KnowledgeC.db*** when a phone call was missed/unanswered and when a voicemail was received.

These are examples of a ***Receive*** notification type that occurred on an iPhone with iOS 14.7.1. These types of notifications will be recorded when a notification is received by the device and when it is displayed on the device screen. In Figure 7, we have four notifications on the test device. There has been no user interaction with the device screen or the notifications.

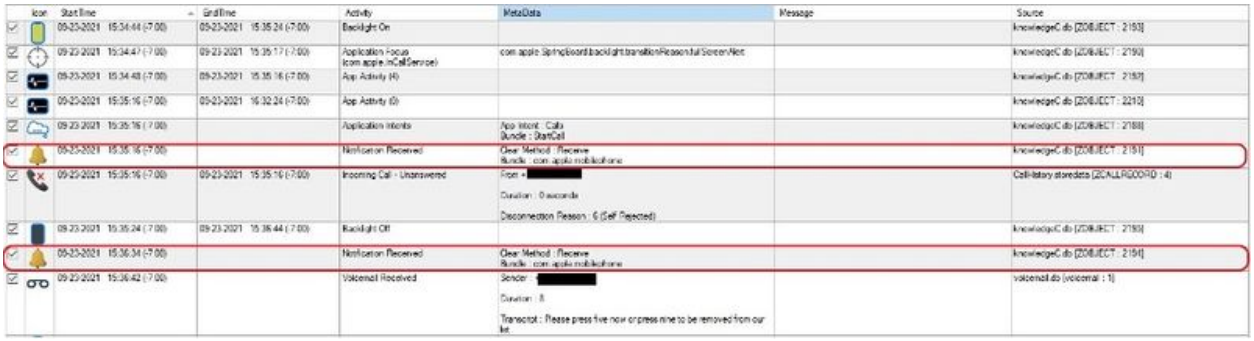

Figure 6

Figure 7

Note: If an application is in focus on a device and new application data is received for that specific application, no ***Receive*** notification type will be recorded. If the application is running in the background, a ***Receive*** notification type will be recorded. Example: if the Apple Messenger application is in focus and additional messages are received, a ***Receive*** notification type will not be recorded in the ***KnowledgeC.db***.

Note: During testing, there was a time when I did not have mobile data service and attempted to send two photos via multimedia messenger. When I attempted to send the messages to an Android device, the iPhone testing device received notifications which indicated the messages failed. The device recorded

two ***Receive*** notifications. These notifications had the same ***Z_DKNOTIFICATIONUSAGEMETADATAKEY__IDENTIFIER***, so again, just a reminder, these identifiers are semi-unique. There is a chance to have duplicates.

Hidden Notification Type:

A ***Hidden*** notification type will be recorded when a notification is hidden from the Lock Screen notification area. This area can be viewed both when the device is locked and when a user swipes down on the screen to see the Notification Center.

During testing this occurred a few different ways:

· If a device is locked and it receives a notification, it will be displayed on the Lock Screen. When a user unlocks the device and accesses the springboard or an application, the notifications that were displayed on the Lock Screen, will no longer be displayed, thus they are hidden, and ***Hidden*** notification types will be recorded in the ***KnowledgeC.db.***

· If a device is unlocked and the user is navigating the springboard or an application is in focus and a notification is received, a Banner Notification will be displayed on the screen. These notifications will be listed in the Lock Screen area until the device screen turns off, either by a user or a device setting.

Note: Please review my previous blog about how to determine what value was set for the display auto-lock.

In Figure 7, there are four notifications displayed on the Lock Screen. At 4:38:56 PM, the test device was unlocked and I clicked on one of the SMS notification banners. An open button appeared to the left of the notification banner. Instead of clicking on the open button, I clicked on the home button unlocking the device. At that time all the notification banners were hidden from the Lock Screen. I checked the Lock Screen and verified that there were no notification banners visible, seen in Figure 8.

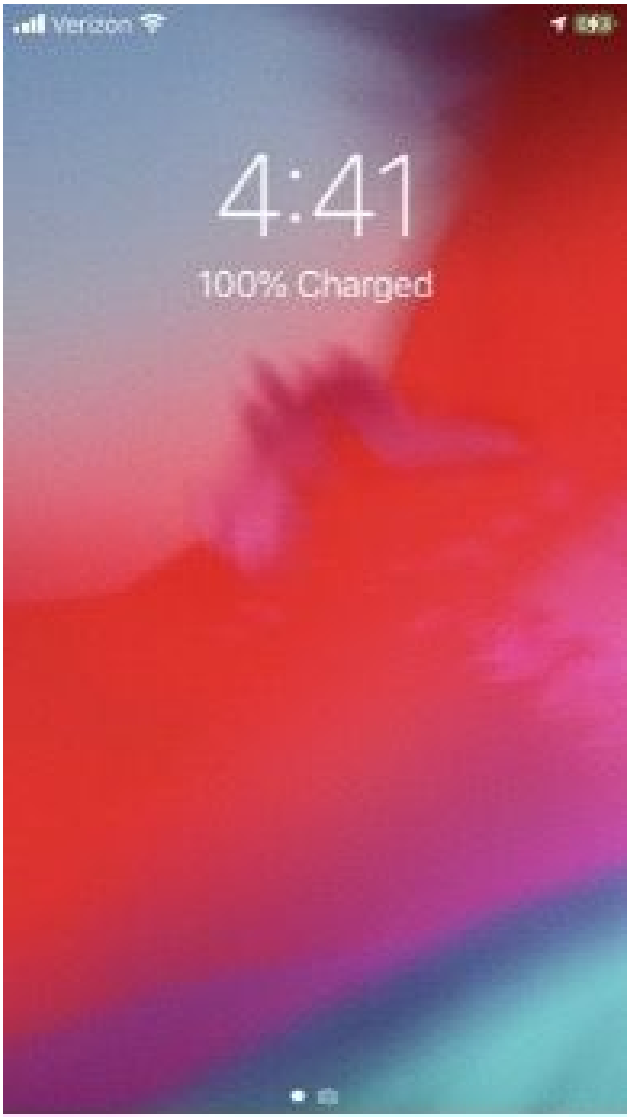

Figure 8

While reviewing the notifications in ArtEx, seen in Figure 9, I noticed there are four ***Hidden*** notifications logged at the time I unlocked the device. I researched if there was any way to link the ***Hidden*** notification type to the ***Receive***notification type. Reminder, bundle identifications are only recorded with a ***Receive*** notification type.

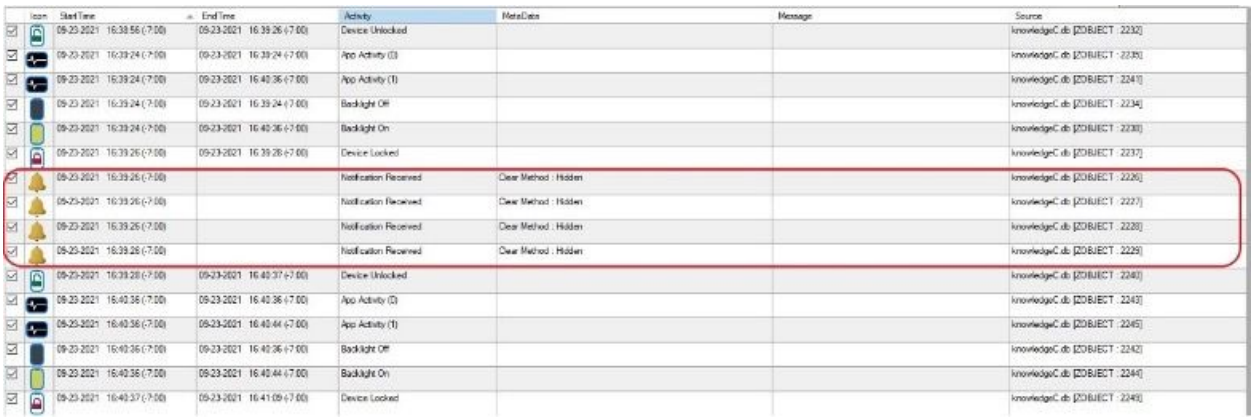

Figure 9

As seen in Figure 10, I noticed Magnet AXIOM was parsing the ***ZSTRUCTUREDMETADATA table Z_DKNOTIFICATIONUSAGEMETADATAKEY__IDENTIFIER*** with the notifications. After additional testing, I was able to determine this is the value that can be used to link related notification types together.

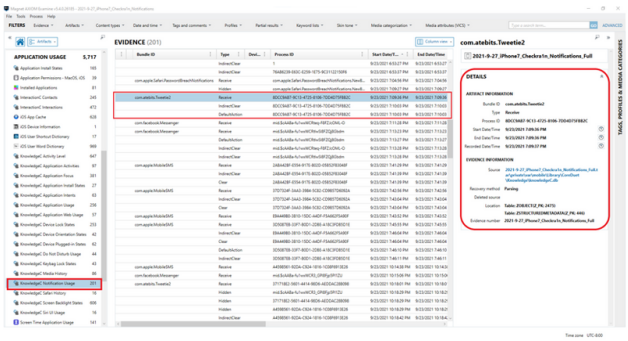

Figure 10

In Figure 11, we can see the data stored in the ***KnowledgeC.db*** when the 4 notifications were received. Then we can see when the notifications are hidden from the lock screen and when they are cleared from the Notification Center. Later I will discuss the ***IndirectClear*** notification type and describe how and why this happened, see the ***IndirectClear*** section for more details.

Figure 11

In the next example, the test device received a SMS notification, Facebook Messenger notification and a Twitter notification. When the notifications were received there was no user interaction with the device. The screen turned ON and OFF on its own. After the three notifications were received, the test device was unlocked by clicking the home button and all notifications were hidden from the Lock Screen. These notifications can still be visible in the Notification Center.

In Figure 12 we can see the activity for the three notifications that were received and hidden. When the notifications were hidden from the Lock Screen, ***Hidden*** notification types were recorded for each notification. Figure 12 shows how these actions look like on the device and in the ***KnowledgeC.db***.

Figure 12

The previous examples had user – device interaction. Based on the testing, ***Hidden*** notifications can be both user and non-user initiated.

In Figure 13, the Notification Center is checked for any active notifications, which there are none. The device is unlocked, and the Facebook Messenger is brought into focus. While the application was in focus, the device received a notification for a SMS message. There was no user interaction with this notification, and it disappears from the screen on its own. Another message is received, and another Banner Notification is displayed, it also did not have any user interaction and disappears on its own.

The Facebook Messenger application was sent to the background, and we can see the messenger application still has a badge notification count, these will only be cleared after the application data is viewed or handled within the application. When the notifications were received, the messenger application was running in the background. All the applications were closed, except for the Facebook Messenger application. While the Facebook Messenger application was in focus, a Facebook Messenger message was received. Notice the device did not display a Banner Notification, as previously seen with the SMS notifications.

After the screen is turned off and the device is locked, we can only see the two SMS notifications are displayed in the Notification Center. The Facebook Messenger message did not generate a ***Receive*** notification type and will not be listed in the ***KnowledgeC.db***.

Figure 13

Clear Notification Type:

A ***Clear*** notification type occurs when a user manually swipes left on the notification banner displayed in the Lock Screen or in the Notification Center, by doing so, reveals a clear button. When the user presses the clear button for an individual notification or the clear all buttons. It removes the notification from the Lock Screen and the Notification Center.

Note: A corresponding ***IndirectClear*** notification type will also be recorded with a matching timestamp as the ***Clear***notification type. Review the ***IndirectClear*** notification type section for more details.

In Figure 14, the test device receives a notification for an incoming SMS message. Then at 7:45:55 PM, another notification is received for a second incoming SMS message.

The user accessed the Lock Screen notifications, swiped left on the second notification, and pressed the clear button. The notification was removed from the Lock Screen and will not be displayed in the Notification Center. After analyzing the database, when the clear button is selected, that specific notification will have both a ***Clear*** notification type and an ***IndirectClear*** notification type with the same timestamp.

The user then accessed the Lock Screen, swiped right on the notification banner, and clicked the open button. The Messenger application was brought into focus and the message was viewed within the application. After analyzing the database when the open button was selected, that specific notification would have both a ***DefaultAction*** notification type and an ***IndirectClear*** notification type with the same timestamp. Review the ***DefaultAction*** notification type section for more details.

In Figure 14 we can see what these actions look like on the device and the data recorded in the ***KnowledgeC.db***.

Figure 14

Dismiss Notification Type:

If a device is unlocked and the screen is ON when a notification is received, if the user swipes up on the notification, before it disappears on its own, a ***Dismiss*** notification type will be recorded in the ***KnowledgeC.db***.

In Figure 15, the test device receives a Facebook Messenger message notification. The user swiped up on the notification to dismiss it. Then a SMS message notification is received, and the user swiped up on the notification to dismiss it. The user then locks the device and views the Lock Screen and Notification Center. The two notifications that were dismissed are no longer visible in the Lock Screen notification area, but they are displayed in the Notification Center.

The SMS notification is cleared from the Notification Center and the Facebook Messenger notification is opened, thus bringing the Facebook Messenger application into focus. Figure 15, shows what this looks like on the device and how the data is recorded in the ***KnowledgeC.db***.

Figure 15

IndirectClear Notification Type:

An ***IndirectClear*** notification type will occur when a notification is no longer displayed in the Notification Center.

In Figure 16 the test device as already received two notifications, one from a SMS message and another from Twitter. An additional Facebook Messenger message notification is received. At this time, the test device is locked, and the screen is turning ON when the notifications are received, then back OFF on its own after the notification has been displayed. Another Twitter notification is received and displayed. Special thanks to Kevin Pagano (@KevinPagano3) for his assistance with an additional notification during testing! An additional SMS message notification was received.

The user unlocked the device by pressing the home button. This user action then hid the notifications from the Lock Screen and a ***Hidden*** notification type was recorded for each one of the notifications that were displayed on the Lock Screen.

After the notifications were hidden from the Lock Screen, the user checked the Notification Center, and all the past notifications are still visible. The user entered the Notification Center and used the clear button to clear the Facebook Messenger notification.

During testing, I received a phone call from someone reminding me that my vehicle warranty was expired. After receiving the phone call notification, the device was unlocked, and the notification was hidden from the Lock Screen, then the user cleared it from the Notification Center. Then the user cleared one of the SMS notifications from the Notification Center. The user then unlocks the device and views the springboard. Notice that all the badge notification counters are still visible.

In Figure 16, we can see what this looks like on the device and how the data is recorded in the ***KnowledgeC.db***.

Figure 16

An ***IndirectClear*** notification type will be recorded when the application is opened which has pending data/badge notifications that have not been viewed.

DefaultAction Notification Type:

The ***DefaultAction*** notification type occurs when a notification is received and is used to open the application to view the data.

We have already seen some examples of the ***DefaultAction*** notification type, but we will review it and show how it was replicated during testing.

In this first example, Figure 17, the device was unlocked, and the user was navigating the springboard. The device received a SMS message notification, which was displayed on the device in a banner notification. The notification banner was clicked, and the Apple Messenger application was opened to view the data. You will notice the ***DefaultAction*** and ***IndirectClear*** notification types are logged one second apart.

Figure 17

Note: When a notification is used to bring an application into focus, the method listed for bringing the application into focus will be *com.apple.SpringBoard.transitionReason.externalrequest*. I'll be doing more research into application in focus methods for both iOS 14 and 15 and will be writing something soon, so stay tuned.

In Figure 18, the second example, we will be reviewing the data stored in the ***KnowledgeC.db*** and attempt to determine what happened on the device, based on what we have already learned in this blog. Based on previous testing I believe:

· On 10/1/2021 at 19:13:14 UTC, the device was unlocked, the screen was on, and the device received a Facebook Messenger notification.

· At 19:13:16 UTC, the device user swiped up on the notification to dismiss it.

· At 19:13:48 UTC, the notification was hidden from the Lock Screen.

· At 19:14:23 UTC, the device user accessed the Notification Center and opened the notification, which then brought the Facebook Messenger into focus.

Figure 18

In Figure 19 and can see what these device events look like in ArtEx.

Figure 19

Orb Notifications:

I have not been able to determine exactly what ***Orb*** stands for and if anyone could provide some insight it would be appreciated.

During testing, I would receive an ***Orb*** notification type in the ***KnowledgeC.db*** when a notification was received, and I interacted with the notification on the device screen. When I pressed and held the notification, the application would open in a small sub-window on the device. I could send messages or preform other actions within the application from this small sub-window. The following are some examples:

Do not Disturb Notification:

During testing, I received several Do Not Disturb While Driving notifications. While the notification was displayed on the Lock Screen, if I clicked on the notification a sub-window would appear and an option to select, ***I'm Not Driving***would be displayed. When this small sub-window with this option would be displayed on the screen, I would receive an ***Orb*** notification type in the ***KnowledgeC.db***, seen in Figure 20.

Figure 20

Apple Messenger:

During testing, I was able to replicate the ***Orb*** notification type by sending a SMS message to the test device, then clicking on the notification and opening the Apple Messenger application in a small sub-window, which allowed me to interact with the application in the small sub-window, which included sending a text message, while the device was locked as seen in Figure 21.

Figure 21

Josh Hickman Image Testing:

After my testing, I decided to test my knowledge of these notifications and loaded up Josh Hickman's iPhone SE iOS 14.3 image into ArtEx.

Facebook Messenger:

In Josh Hickman's documentation there is a section for Facebook Messenger, which is displayed in Figure 22. Notice in Figure 22, his test device sends and receives several messages, media messages and video calls.

Name: Facebook Messenger
Version Number: 297.0.0.29.116
Install Date: 2021-01-30
Install Time: 10:52

Username: 919-579-4674

Note: Some chat data was loaded from previous conversations. See data from the Android 10, iOS 13, and Android 11 images.

Date	Time	Action	Message
2021-02-01	15:38	Login to app	
2021-02-04	14:32	Sent message	Ok I'm here. Got sidetracked
	14:33	Received message	No worries. Ill send over a photo in a minute.
	14:34	Liked received message	
	14:35	Received picture	(*Ping 127.0.0.1*)
	14:36	Saved picture	(*Ping 127.0.0.1*)
	14:38	Sent message	Thanks. I'll send one over.
	14:41	Sent picture	(*A swing*)
	14:42	Received message	Thanks.
	14:43	Incoming audio call	(~1:30)
	14:45	Outgoing audio call	(~1:30)
	14:47	Incoming video call	(~1:30)
	14:50	Outgoing video call	(~1:30)
	14:52	Incoming message (Secret)	This is the start of a SECRET chat
	14:53	Outgoing message (Secret)	Interesting. Look like no audio/video calls.
	15:14	Incoming message (Secret)	Yah this whole secret thing is interesting
	15:16	Outgoing message (Secret)	Here come a picture.
	15:17	Sent picture (Secret)	(*Teeter*)
	15:18	Received picture (Secret)	(*Ironic*)
2021-02-18	14:24	Started sharing live location	
	14:31	Stopped sharing live location	
	14:33	Sent static location	(*312 Lively Oaks Way, Holly Springs, NC 27540*)
	14:35	Received static location	(*208 Meares Bluff Lane, Holly Springs, NC 27540*)
	14:37	Started receiving live location	
	14:44	Stopped receiving live location	

Figure 22

Figure 23 is that same timeframe viewed in ArtEx. Notice between 14:32 though 15:18, no new notifications were being received on the device. This is because the Facebook Application was in focus, and everything is occurring in real time on the device.

At 15:24:14 there is a Receive notification type for Google Duo (com.google.Tachyon).

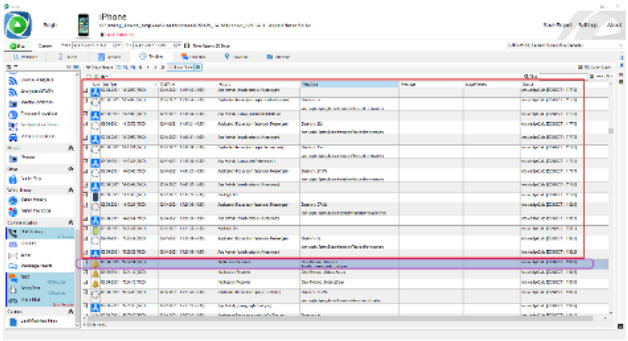

Figure 23

Google Duo:

In Josh Hickman's documentation there is a section for Google Duo, which is displayed in Figure 24. Notice in Figure 24, there is documentation that on 2/4/2021 at 15:24, a note is received, which contained a message *What is this??*

Name: Google Duo
Version Number: 116.0.17657
Install Date: 2021-01-30
Install Time: 10:57

Username: thisisdfir@gmail.com

Note:

Date	Time	Action	Messages
2021-02-02	13:53	Login to app	
2021-02-04	15:24	Received Note	What is this??
	15:25	Sent Note	I have no idea. Just roll with it.
	15:27	Incoming audio call	(1:45)
	15:30	Outgoing audio call	(1:45)
	15:33	Incoming video call	(1:34)
	15:36	Outgoing video call	(2:00)

Figure 24

Figure 25 is that same timeframe viewed in ArtEx. Notice there is a ***Receive*** notification type, followed by a ***DefaultAction***, then an ***IndirectClear*** notification type. This indicates that when the notification was received on the device, the user used the notification to bring the application into focus. We can see in ArtEx the application started in focus at 15:24:20, which is one second after the ***DefaultAction*** notification type was logged.

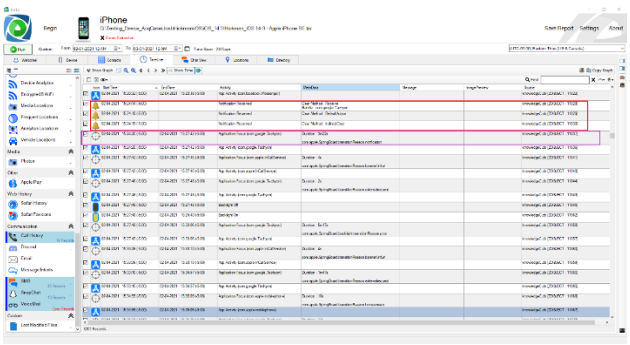

Figure 25

Messenger Application:

In Josh Hickman's documentation there is a section for Messages, which is displayed in Figure 26. Notice in Figure 26, there is documentation that on 2/15/2021 at 13:06 (Eastern Time) an iMessage is received. Because Josh Hickman's device is set to Eastern Time and I am in Pacific Time, for this example I will be referencing artifacts in UTC.

Name: Messages

Note: Location sharing completed through Find My is documented here in addition to the Find My app. They appear as system messages in this app.

iMessages were sync'd with iCloud. Some messages were sent/received via Messages on macOS and will be denoted here with (*macOS*).

Date	Time	Action	Message
2021-02-15	13:04	Sent iMessage (*macOS*)	So this first iMessage is coming from the Mac.
	13:06	Received iMessage	Nice. Are you using a VM?
	13:08	Sent iMessage (*macOS*)	Yes sir. I will send a bookend message when I switch to iPhone.
	13:09	Received iMessage	Awesome. Don't forget we need to test out replies, too.
	13:10	Sent iMessage (*macOS*)	This is a reply to the message at 13:09. I don't think I've ever tried this within macOS before.

37

iOS 14 Image Image Created by: Joshua Hickman

	13:11	Received iMessage	Does it look any different? (*This message is a reply to the message at 13:10*)
	13:12	Sent iMessage (*macOS*)	Nope. Looks like it does in iOS. Just bigger (*This message is a reply to the message at 13:11*)
	13:14	Sent iMessage	I have now switched to iPhone
	13:15	Received iMessage	What else are you going to do on the Mac?
	13:17	Sent iMessage	Some stuff I'm [sic] Safari, Notes, Music, and Calendar. Because I'm on a VM, there isn't a true WiFi connection, so Maps isn't fully working.
	13:19	Sent iMessage	Oh, and Photos.
	13:20	Received iMessage	I was going to say we should probably share a photo album. What about Files? *This message is a reply to the message at 13:19*)

Figure 26

According to Josh Hickman's documentation his device received an iMessage at 18:06 UTC. Can we answer the following questions?

· Did the device display a notification on the screen for this message?

- Did the user interact with the screen if a notification was displayed?
- How was the notification cleared from the device?

In Figure 27, we can see in ArtEx at 18:04:18 UTC, the device was unlocked, but notice an application was not in focus. A message was sent at 18:04:46 UTC, but was not sent from the iPhone, it was sent from a synced Mac.

At 18:06:37 UTC, a ***Receive*** notification type was received on the device, thus because the device is unlocked, a banner notification would have been displayed on the screen.

At 18:06:37 UTC, a ***Dismiss*** notification type was recorded, thus when the banner notification was displayed on the device, the user interacted with the screen and dismissed the banner notification.

At 18:06:49 UTC, the Messenger application (com.apple.MobileSMS) was brought into focus via the home screen.

Note: If the notification was used to open the application a ***DefaultAction*** notification type would have been recorded and the method of bringing the application into focus would have been different.

At 18:06:52 UTC, a ***IndirectClear*** notification type was recorded because the application was opened, and the user viewed the message. The notification will no longer be displayed on the Lock Screen or the Notification Center.

Notice in Figure 27, there are several messages being sent back and forth. Some from the synced Mac and others from the iPhone, but notifications are not being recorded. This is because the Messenger application was in focus when these messages were being sent and received.

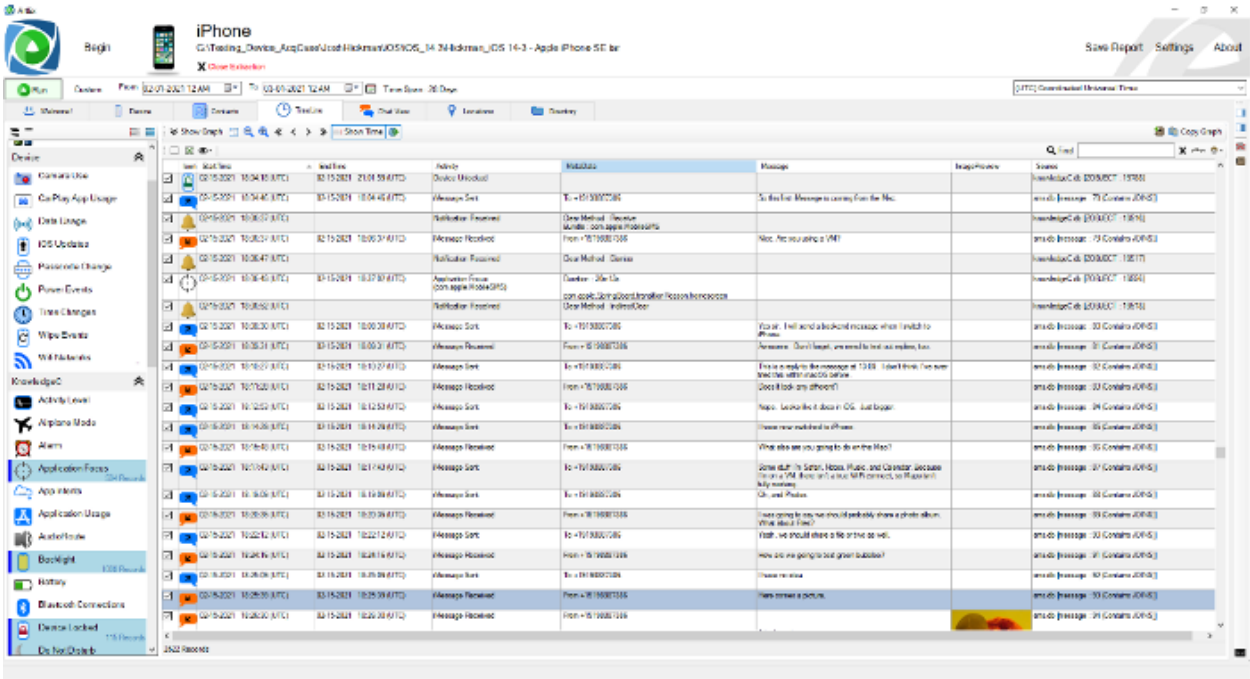

Figure 27

In Figure 28, I have combined ArtEx and the ***KnowledgeC.db*** date in a video to again show how this works together.

Figure 28

Considerations:

Most of this testing was done on a device with iOS 14.7.1, but I believe the results of this testing should be true with other versions of iOS 14.

I plan to conduct some additional testing on how device notifications are stored if the device is connected to a vehicle with and without CarPlay. I have done some preliminary testing, and it appears to be very similar to what has already been detailed in this blog. I will add any additional information learned to this blog later.

Conclusion:

I believe it has been demonstrated there are certain types of notifications that can be used, in conjunction with other device data, to prove whether a user interacted with a device at a certain time.

Based on testing ***ZVALUESTRING*** is the notification types and each notification type is created when:

- Clear = Notification was cleared by a user via the Lock Screen or Notification Center
- DefaultAction = An application is opened via a notification
- Dismiss = Notification was dismissed by a device user when the notification was received
- Hidden = When notifications are hidden from the Lock Screen
- IndirectClear = When notifications are cleared from the Notification Center
- Orb = Is triggered via user interaction when an application is opened in a small sub-window
- Receive = Is when a notification is received and displayed on the device

References:

- August 2018 – Sarah Edwards

o https://www.mac4n6.com/blog/2018/8/5/knowledge-is-power-using-the-knowledgecdb-database-on-macos-and-ios-to-determine-precise-user-and-application-usage

o https://objectivebythesea.com/v3/talks/OBTS_v3_sEdwards.pdf

- August 2019 – Christopher Vance

o https://blog.d204n6.com/2019/08/ios-12-delivered-notifications-and-new.html?m=1

- October 2019 – Ian Whiffin

o http://www.doubleblak.com/m/blogPosts.php?id=2

- Josh Hickman's Test Device images:

o https://thebinaryhick.blog/

DFIR Review

Determining what notifications, if any, were displayed on a mobile device is a common question asked of forensic examiners. Taking it a step further, understanding what a user of a device did when they were presented with a notification can provide user behavior patterns. The author of this paper demonstrated their understanding of the KnowledgeC and the gaps in research associated with notifications.

Reviewers found that some of the figures in the paper were missing or could not be viewed.

As the Notification Center may be available from the lock screen, examiners should be cautioned on attributing interactive behavior to a a specific individual without performing additional analysis.

Future Work (provided by DFIR Review)

Reviewers are interested in seeing additional values for ZVALUESTRING and additional data that may be associated with device usage. Reviewers also suggested not using video attachments in the submission, if possible, as they do not publish well.

Future work on this topic could include testing newer iOS versions to see if anything has changed and testing if additional forensic tools can verify this information. Reviewers also suggested trying different types of applications such as email and also looking at reminders and other alerts.

Reviewers

Jessica Hyde, David Loveall (subreviewer) (Methodology Review, Validated Review Using Reviewer Generated Datasets)

Troy Pugliese (Methodology Review)

Aricia Kulm (Methodology Review)

Zheng Jie Chan (Methodology Review)