EDL Mode with Cellebrite's UFED
By Scott Lorenz
7-19-2018 – Draft
Edited by Andrew Rathbun

Document Purpose: To describe the various methods, techniques, and tips for placing devices in EDL mode and extracting them using the UFED. This document is designed to change and update as needed.

General Overview: This document is designed to be a general reference for placing devices in EDL mode and using Cellebrite's UFED 4PC and UFED Touch to extract the devices using decrypting and non-decrypting options provided in UFED extraction software. All tests were conducted using the UFED 4PC but procedures should be the same or similar using UFED Touch devices. Advanced methods of device extraction, utilizing Cellebrite's EDL exploit, are also discussed.

Requesting Access to Mobile Device Forensics and Analysis forum : When requesting access or membership in the forum, please include information about who you are (name), what agency you are from or your business or company, and your interests in the forum or purpose. That information in your initial request will save time and prevent forum moderators from having to send requests for more information.

Links: Throughout this document there are links to photographs, diagrams, and other documents. Links may be in the text of the document or by clicking on a photograph or diagram. It may be necessary to download this PDF in order for those links to work properly as opposed to just viewing it in your web browser. The links are to photos and documents in the resources folder of the Mobile Device Forensics and Analysis forum. The Links in the Table of Contents lead to that corresponding section of the document.

Continual Updates: I am continually making updates to this document. That includes minor errors, changes in support or function of the UFED, and general formatting.

Special Thanks: Thanks to Andrew Rathbun from the Michigan State University Police Department for editing this document.

I am just a user of Cellebrite's software, taking advantage of exploits and tools they create. Cellebrite is not the only tool I use for forensic examinations. Like most examiners, I use many tools and techniques to extract data

from mobile devices and PCs. I did not collaborate directly with Cellebrite in writing this paper. I don't have access to inside information regarding upcoming exploits, other than what Cellebrite releases publicly or in forums. That means I can be wrong regarding my assessment of the limits and abilities of Cellebrite's EDL exploits.

Overview of EDL Mode

For those not familiar with EDL mode and specifically Cellebrite's use of EDL mode, I recommend you first read Shahar Tal's "*Practical Guide for Qualcomm EDL Physical Extractions*". That PDF document is located in the Resources Folder/EDL Extractions, on the Mobile Device Forensics and Analysis group. In that same folder is the PDF version of the "Cellebrite EDL Webinar 2-21-18". That will also give a general overview of EDL extractions – via Cellebrite's UFED. It is always best to refer to the source (developers) of forensic techniques and tools for the most accurate information. Although I was a presenter in that seminar, it did not and could not cover all techniques, methods, and abilities of EDL extractions. So I hope this document and its linked supplements will fill in some of the gaps.

Current information on EDL and UFED capabilities

In the world of mobile forensics something will change five minutes after publication. I will date certain portions of this paper so that modifications can be made when updates are made or support for devices or processors changes. There are links to diagrams and flow charts and step-by-step guides on EDL extractions. Those will likely change with future UFED releases and changes to device manufacturer's hardware and software. I will try to make modifications or at least date items so users can assess whether the information is still correct or relevant.

EDL Research and Development

Examiners need vendors with heavy R&D to continue doing what they are doing. We want vendors like Cellebrite to pass tools down to those who will use them legally and responsibly. The exploit provided by Cellebrite is a powerful tool for many reasons and probably for some reasons that many examiners may not have considered. Cellebrite's EDL exploit can be used by inexperienced users but it also allows experienced users the ability to perform some advanced maneuvers to reach encrypted data that would normally be off limits. So while we want access to more devices from companies like Cellebrite, we also have to do research and experimentation to be able to get the most out of exploits like EDL. Methods for getting devices into EDL mode varies significantly across the universe of phones.

Do any other vendors use EDL?

Cellebrite is not the only vendor that utilizes EDL mode as an exploit. Other vendors and examiners use EDL mode as a tool to extract data from devices using other techniques and software. This document is not designed to review or cover the many different vendors, software, or hardware in the world that make use of EDL mode even though I have used some. I do believe Cellebrite stands apart in their use of the EDL exploit for many reasons, but especially their ability to decrypt data.

The techniques I use and describe in this document, are designed be a general overview of the various aspects of EDL mode and working with various devices using Cellebrite UFED software. There is often more than one way to achieve EDL mode and often more than one way to pull the same device using UFED software via EDL methods.

Why use EDL for that Extraction?

I try to use the simplest, safest, most effective technique available when dealing with evidence from actual cases. There is often more than one way to extract a physical image from a particular device. EDL mode may or may not be the easiest method to accomplish the same task with that device. This paper is a demonstration of Cellebrite's use of EDL – limited to my ability with it. EDL is another tool and another method to add to your inventory of tools. So for purposes of experimentation and demonstration, in this paper I sometimes intentionally pick the most difficult method to perform a procedure just to demonstrate and show that it will or will not work. I also intentionally break devices to demonstrate what is possible with the EDL exploit.

What is EDL Mode?

Shahar Tal describes EDL simply in his Practical Guide in which he refers to a feature of devices running Qualcomm processors known as (Emergency Download) Mode, "…designed to allow low-level access to the chipset for device analysis, repair or re-flashing." EDL mode was not developed by Cellebrite. EDL is a feature of Qualcomm processors that Cellebrite and others exploit in order to extract data from the device. Like every other method for extracting data from mobile devices, achieving EDL and exploiting EDL varies greatly from device to device. For forensic tools, the EDL exploit also varies greatly from vendor to vendor. Although I am writing about Cellebrite's use of EDL, I have tried other vendors for EDL. My use of those other vendors has not been what I would call "exhaustive" as with my use of Cellebrite, but the significant difference in vendors that is very obvious is Cellebrite's ability to decrypt using their EDL exploit. Just because other vendors or methods can extract data by exploiting EDL, does not mean that extraction will be decrypted.

Can all Qualcomm devices placed in EDL mode be extracted?

No. Based on questions I have received on EDL, there may have been some initial confusion regarding devices that can be placed in EDL mode versus devices that can be extracted using Cellebrite's exploit of certain devices having been placed in EDL mode. Just because a device can be placed in EDL mode doesn't mean it can be extracted using Cellebrite's EDL method.

Most devices running Qualcomm processors can be placed (forced) into EDL mode. Cellebrite can take advantage of EDL mode (extract the device) only on specific devices or on a wide range of devices running a specific model of Qualcomm processor. For example, most devices running the Qualcomm MSM8909 can be decrypted and extracted by Cellebrite using their EDL exploit. Alternatively, for example, I can place most devices running the Qualcomm MSM8210 in EDL mode. However, to my knowledge, none of them can be extracted using Cellebrite's EDL exploit. Of course there may be another vendor or method for exploiting a particular device or processor using EDL, but I am focusing on the method provided by Cellebrite's UFED.

Why doesn't Cellebrite use the EDL exploit on older phones running Qualcomm processors?

I personally get asked this question frequently. Of course I can't speak for Cellebrite, but I will take some liberties based my modest experience with mobile forensics. Cellebrite was asked this question when they first released their EDL exploit. First of all, what Cellebrite has done and is doing with the EDL exploit is difficult. We sometimes take for granted the ability to push a button and extract decrypted data from a locked device. Cellebrite's EDL exploit requires a lot of R&D as does many of their other exploits. Devoting time and R&D manpower to breaking software from the past is not always practical or possible, especially when there are many working exploits already in place on those older devices.

Does it harm devices to place them in EDL mode?

I always tell my Criminal Justice students not to use absolutes when dealing with the law. I therefore fear using absolutes when talking about digital forensics, but the general answer to the question "Does EDL damage a device?" is no. I say that relying on Cellebrite's expertise and my modest experience in placing hundreds of devices in EDL mode using many different methods. I have some individual test devices that I have placed in EDL mode easily over 100 times, mostly by shorting them, with no adverse effect.

Please do not misinterpret my statement to mean that no harm can come to devices by attempting to place them in EDL mode. In my experience with EDL, it is the method of creating EDL that harms devices and not EDL mode. It is most often user error that damages a device. One of the reasons EDL mode was created was for when something goes wrong during the booting process. EDL is often the result of "something wrong". Shorting out a mobile phone is definitely violating the terms of any warranty and most cell phone manufacturers and service providers would say it is dangerous.

With regard to using eMMC shorting to force phones into EDL mode, shorting the wrong location can damage a phone permanently. Disassembly of phones can damage phones so that they may not be extracted or turned on

again. Using ISP and JTAG techniques can be performed repeatedly on the same device without affecting it, or those techniques can be done the wrong way once and destroy it.

Having training in ISP or JTAG and experience soldering is very beneficial when shorting devices into EDL mode. Using a microscope is highly recommended. Experience with device repair and disassembly is also very helpful. Knowing where to short is accomplished by training and practicing all of those disciplines. This is one of the many reasons why ISP, JTAG, and Chip-Off training is still essential. In my opinion, those classes will benefit you in many ways even in a world of increasing encryption.

Can EDL bypass user passcodes and pattern locks?

Please keep in mind that user passcodes and pattern locks discussed here are not the same as Secure Startup. Secure Startup will be discussed in another section of this paper. Yes, Cellebrite's EDL exploit works on devices with user passcode and pattern locks. That includes the decrypting method and the non-decrypting method of extraction. With the non-decrypting EDL extraction, booting is not required. The non-decrypting method is a low-level pull that, although takes place via processor exploits, is very much like an ISP pull. ISP delivers data to us via direct access to the phone's storage with no processor interaction, as does Chip-Off. With ISP, we are not receiving the data through a USB port. We are receiving the data through the Data lines connecting the processor to the phones storage. With ISP that is usually Data0 but depending on the tool we use for ISP, more than one Data line may be used by soldering into Data0 – Data3 or more, for example. Thus with ISP, user passcodes are irrelevant.

The same is true with the non-decrypting EDL extraction. Cellebrite is using EDL mode to access and exploit the processor and then command it to dump the phone's storage through the USB port, as opposed to ISP which pulls data directly off the storage. With the non-decrypting EDL method, this is done pre-boot, which is why extracting data from devices that are not encrypted can be done on phones that are damaged and can't boot. It is this reason that EDL is something I push to be used on damaged devices, especially devices that are not encrypted.

With devices that are encrypted, Cellebrite will use the EDL exploit to force the phone to boot and apply their decrypting bootloader. The device will boot into Android and to the user screen just as if you powered it on yourself. The user passcode or pattern lock is still in place and you can interact with the screen and see that the lock is still in place. However, because the decrypting bootloader has been applied, the fact that the phone is still locked doesn't matter. Using the decrypting EDL method in the UFED means the phone must boot, locked or not.

The same is true for using ADB to create EDL, either having the UFED do it automatically or the examiner doing it through Command Prompt. It is necessary to know the passcode and unlock the phone in order to access Developer Options and USB Debugging, but there is no need to remove the user passcode or pattern for the decrypting extraction to work once the settings are in place.

What is the difference between the decrypting and non-decrypting methods of EDL extractions with the UFED?

Aside from the obvious answer that one decrypts data and the other doesn't, for the examiner, the most important answer to that question is: the process used by Cellebrite. I do not have the knowledge, access, or expertise to do anything but describe this in general terms. The first thing I tell anyone who asks is that, generally, the non-decrypting method is much easier. By easier, I mean there are fewer steps involved, no need for the phone to boot, and no requirement to create EDL more than once during a single extraction.

Based on the questions I frequently receive, both on and off the forum, the non-decrypting method is much easier than everyone imagines. Devices don't have to boot or power on at all for the non-decrypting procedure. Before Cellebrite released their EDL exploit, there were many devices that required ISP or Chip-Off in order to obtain a physical image. With the EDL exploit in the UFED, I can extract many of those same devices in about 20 minutes

without a battery, without ever turning the phone on, and without soldering or disassembly – depending on the device.

Non-decrypting Method

On devices that are not encrypted, using the non-decrypting method, either under a supported device profile, or by selecting "Smart Phones/PDAs Qualcomm" will be a quick and easy, low-level extraction that will not require the device to boot or even function. This method can be performed on badly damaged devices and in most cases does not require a battery.

EDL works well even if the device doesn't

When I get a question about extraction possibilities for damaged devices, not only is EDL a consideration for me, it is often my first consideration. Using EDL to extract devices that are not encrypted requires very little activity or functionality from the phone – no battery, no booting, and no screen interaction. Although the process is not exactly the same for encrypted devices, Cellebrite's EDL method still works well on damaged devices that are encrypted. I have devoted a section for damaged devices at the end of this paper but felt it was worth mentioning here.

Why remove the battery?

When I say, "does not require a battery for devices that are not encrypted" I mean most of the time the battery is just in my way for this extraction. I don't need it. I don't need the phone to boot and I don't want the phone to boot. Booting means the device can start up and thus I have to consider all of the implications of starting a device that was seized in unknown condition or may have been shut down and stored without activating Airplane Mode. I don't have to worry about charging a battery. Devices that will not boot at all can be extracted like this. Multiple extractions without the battery will not change the hash. Any time I can retrieve evidence from a device without booting the device, I will use that method – either exclusively or before other extractions that require that device to boot.

All that is needed is a working USB port, although I have been able to extract many devices without a working USB port (discussed later). Of course if the device is in the same condition as depicted with the Z812 photo (board only), eMMC shorting or the EDL cable will be needed to create EDL Mode. Working devices are needed for creating EDL via ADB and generally, some button combos that create EDL. There are some exceptions for button combos to create FTM and DFU (discussed later).

In terms of the effect on the image hash, EDL (non-decrypting) is similar to extracting a device via Chip-Off or ISP. The phone is not required to boot and thus the hash value of the image will not be affected even with multiple pulls (provided the phone is not booted between pulls). On devices I have tested with this method, multiple pulls did not change the image hash.

The non-decrypting method is the easiest, procedurally, of the two methods. The vast majority of devices I pull with this method are done without the battery and many times with only the logic board connected to USB. With few exceptions, a power button is not needed. After creating EDL, the device will go through two steps and immediately start to dump. If you have a device that is not encrypted, the non-decrypting selection is the method you should use. I have created a workflow chart of steps for using the non-decrypting EDL method in the UFED. (EDL Non-decrypting Pulling Steps)

Decrypting Method

On devices that are encrypted, select, "Smart Phones/PDAs Decrypting Qualcomm" method. This method requires the phone to boot and therefore is not as easily deployed as the non-decrypting method. With few exceptions, devices require a battery for this method and the phone must be in general working order. Most mobile phones require a battery to boot and operate, even with USB power. But as with everything, there are exceptions to that.

The decrypting EDL extraction can generally be done without a working screen or without the screen attached at all because there is no user interaction necessary on the device screen – unless using ADB decrypting method (discussed later). There is also another exception to the general rule regarding no screen interaction. That exception involves a device with secure startup with the passcode known to the examiner (also discussed later).

When referring to write-ups on specific phone models, please keep in mind that Cellebrite makes continual changes and improvements to their software. So write-ups by me or others may not work the same way or be necessary due to an improved version of the UFED. I have seen this manifest itself with steps that are no longer required that once were.

A working power button may be necessary for this method to work. You may be prompted to place the device in EDL Mode more than one time in the same procedure. In this procedure the phone will go through 6 steps and then an executing phase. A second request for EDL Mode, if requested, will occur in Step 4. The hash value of the image will be different if more than one pull is made. You will see that there are several options for creating EDL Mode with a decrypting pull. Which option you select can affect whether or not the device will pull. With some phones, the procedure is very specific. This is because of the timing in which the UFED causes the device to boot with the decrypting bootloader. I have created specific write-ups on certain models known to have this issue. On many other models, I have identified a procedure that works most often or is most reliable. But all of that is a moving target so I am constantly changing, adapting, and ready to apply multiple methods to one device.

Some devices will work with many different methods or will take care of themselves after the first EDL Mode is created. If a second request for EDL is made during this procedure, the method I use most often is 11A, depicted in my decrypting EDL diagram. 11A works on many different brands and models of phones when using eMMC shorting (EDL Decrypting Pulling Steps). But my method and techniques are not the only way and may not be the best, or they may have been the best and now they are not needed. It's digital forensics, things will change.

What if I pull an encrypted device using the non-decrypting method?

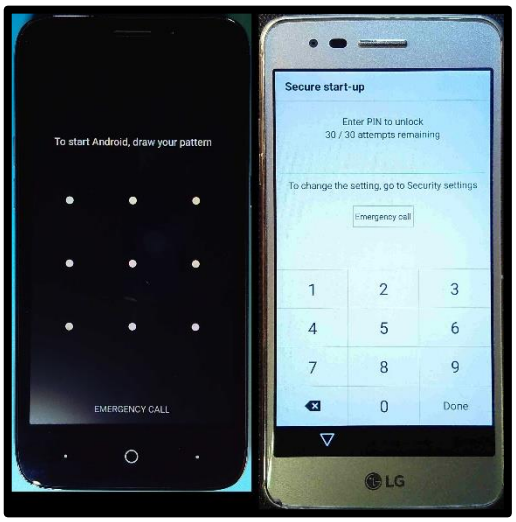
The device will pull just as easily as a device that is not encrypted, but the user data will be encrypted. To be clear, I can pull encrypted devices the same way I pull non-encrypted devices (no battery, etc.). Because you have told the UFED to use the non-decrypting method, it will perform a low-level extraction without regard for the condition of the user data. Extracting encrypted data will be just as easy as if it were not encrypted – the data is just useless because it is encrypted and the user data can't be decrypted after extraction.

Using the non-encrypted method on an encrypted device will not harm the device. Just like pulling an encrypted device via ISP or Chip-Off, you will end up with encrypted user data. Physical Analyzer will tell you the device's user data is encrypted when you begin opening the extraction. When that happens, simply deploy the Decrypting EDL method and extract the device again, which will then yield a decrypted image. The non-decrypting EDL method will only be useful on devices that are not encrypted, but is harmless to an encrypted device.

What happens if I pull a device that is NOT encrypted with the decrypting bootloader?

The device will extract using the decrypting method but this will not harm the device or change the user data. You will have a good, useable extraction and there is no need to pull it with the non-decrypting EDL method after that. The phone will boot for this process to complete so it will change the hash value of the image. Not knowing whether the device is encrypted or not is a definite possibility today as not every device comes encrypted by default. Some devices that are not encrypted by default can easily be encrypted by the user. Some devices running up to Android 7.1.1 may not be encrypted by default, but most are. So, using the decrypting EDL method will work on both encrypted and non-encrypted devices.

Secure Startup and EDL

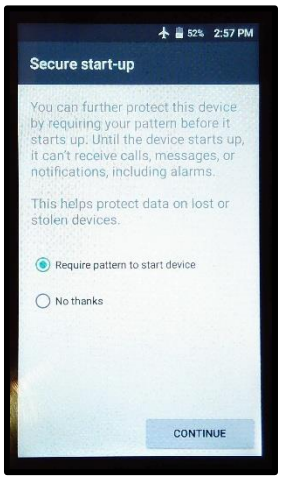


At the writing of this paper, Cellebrite may be able to bypass Secure Startup through Cellebrite Advanced Services (CAS). Whether or not a bypass of Secure Startup will be an exploit that can be passed down to users of the UFED…I hope that is possible in the future. Since Secure Startup means we will not be able to extract the device with any method, this should be a short section of this paper. However, there is information to be gleaned from discussing Secure Startup and it helps explain how EDL extractions work in the UFED. I have also had questions from examiners and investigators regarding Secure Startup and how it affects or does not affect seizure and storage of devices.

How do you know if a device has Secure Startup?

The easiest way is to start or restart a device which will lead you to the Secure Startup screen as you can see from the photograph. The screen may look different depending on device brand. Secure Startup is a feature of Android that allows the user to require a passcode or pattern to be entered before Android can be started. Thus, an examiner powering on a device will see this screen "To Start Android, draw your pattern" or enter your passcode – whichever method is picked by the user.

Does Secure Startup mean the device is encrypted?



Secure Startup answers the question we frequently have: "Is the device encrypted?" If the device has Secure Startup, the device is encrypted as this feature is only available with device encryption. The absence of Secure Startup does not mean the device is not encrypted. Secure Startup is an option given to a user when first setting up their device, or it may be added at any time by the user. Even though encryption may be enabled by default, Secure Startup is still a user option.

How do you know if Secure Startup is enabled on a running device?

Secure Startup only requires a passcode when the device is first powered on or restarted. This is separate from the user lock screen which requires a passcode or pattern when the screen is turned off or times-out. For devices that are first powered on or restarted, we will see the Secure Startup screen if it has been enabled. If the device is running and has a locked screen, there may be no way to know if Secure Startup is enabled on the device with just a manual inspection.

What happens if I try to extract a device with Secure Startup using the EDL method?

The answer is different depending on whether or not you use the non-decrypting method or the decrypting EDL extraction in the UFED. You might ask why an examiner would even attempt such an extraction. Consider a typical scenario in which the examiner may not know the device has Secure Startup enabled. If the device was seized in the off-state or if it was seized in the on-state and shut down, the person who seized the device may have only

seen the user lock screen before shutdown. So there is no way to know if Secure Startup is enabled until the device starts.

If I receive a device I believe may or may not be encrypted, I may attempt a non-decrypting EDL extraction first, for reasons I cover in this paper. That will be an easy pull for a device that will remain in the off-state for my extraction and will not change the hash. If the device is encrypted, I will find out when I attempt to open the image in Physical Analyzer. If it is, I simply attack the device with the decrypting EDL method to get my decrypted image. As discussed in this paper, there is no harm in using either method on devices whether encrypted or not.

A device with Secure Startup pulls exactly like a device without Secure Startup when using the non-decrypting EDL method in the UFED. The UFED will extract the device and when you open the image in Physical Analyzer you will be informed that the "User Data is Encrypted…" You will not be told that the device has Secure Startup nor will you see that on the device screen during the extraction when using a non-decrypting EDL extraction in the UFED. This is because the non-decrypting extraction does not require the device to boot – at all. This scenario also provides an excellent demonstration of how low-level the non-decrypting EDL extraction is and why it works on damaged and non-functioning devices so well. It will extract a physical image from a device running Secure Startup, which by definition and purpose prevents Android from starting. This is why non-decrypting EDL is virtually identical to ISP with regard to result.

In our scenario of the device seized without knowing whether Secure Startup is enabled, having extracted encrypted data with our first pull, we will now deploy the decrypting EDL method in the UFED expecting to get a decrypted extraction. It is only during that decrypting EDL extraction process that we will be informed the device has Secure Startup. I tested this exact scenario using a ZTE N9136 Prestige 2 running an MSM8909 loaded with Android 6.0.1 and encrypted by default. I put Secure Startup on the device and attempted decrypting and non-decrypting EDL extractions with the UFED.

Should a device be left running when seized to avoid Secure Startup?

I will answer that question considering only EDL extractions. Secure Startup means the device is encrypted and so even though the non-decrypting method doesn't require Android to start, the extraction will yield encrypted user data and for practical purposes will be useless. That means the decrypting EDL method will have to be deployed. Even if the device is seized in the on-state (running) a decrypting EDL extraction using any technique to achieve EDL Mode will require the device to boot from EDL Mode. That means we will run right into the Secure Startup screen when the UFED orders the device to boot during the decrypting EDL extraction. A bootable device is necessary to apply Cellebrite's decrypting bootloader. All of that means keeping a running device alive in hopes of avoiding the Secure Startup screen really doesn't matter if we are considering only EDL extractions. Other extractions or exploits that do not require booting to apply or a shutdown, may be different and thus keeping the device alive may be prudent with other scenarios.

If the Secure Startup passcode or pattern is known, must the Secure Startup pattern or passcode be removed for a successful decrypting EDL extraction in the UFED? (8-30-18)

This might be the only time interaction with a device screen will be necessary for EDL extractions – save ADB setup. We know that EDL extractions are not affected by screen lock. As long as the device boots, it doesn't matter that it boots to a locked screen.

Secure Startup with known passcode – UFED 4PC

Regarding Secure Startup with a known pattern or passcode, I tested this scenario using the ZTE N9136 again. I placed the device in FTM mode (discussed later) which allowed the UFED to apply the decrypting bootloader for an extraction. The UFED caused the device to boot from EDL Mode and the device behaved as normal and responded to the UFED command to boot. The device reached the Secure Startup screen requesting the pattern shortly after the UFED extraction reached Step 6. I drew the known Secure Startup pattern immediately which allowed the device to continue and boot. This caused a temporary hiccup in which I heard the UFED repeatedly handshaking

but the extraction was a success with this device. Although I didn't have to remove Secure Startup for the N9136, it may be necessary to remove Secure Startup completely with other devices if you have the known pattern or code.

Secure Startup with known passcode/pattern – UFED Touch2 (ZTE Z971) 8-30-18

I used the UFED Touch2 to extract a ZTE Z971 running Android 7.1.1 with an MSM8917 processor. This device is encrypted by default. I put a pattern lock on the device and set it up to require a pattern to start the device, thus enabling Secure Startup. I shut down the device and placed the device in DFU mode. With the battery connected I held volume down and volume up while connecting to the UFED Touch. In step 6 of the extraction process, the phone is commanded to boot by the UFED Touch2, but with Secure Startup it stops and prompts the user for the pattern. I entered the known pattern for this device, it continued with the boot process and the extraction was successful. Note the device still boots to the user lock, but that does not affect the extraction.

Device intake and testing for EDL Mode

Deciding which EDL extraction to use (decrypting or non-decrypting method) will depend on several factors that can include the state and condition of the device, whether it is known if the device is encrypted or not, and to some extent the experience and training of the examiner. Understanding the statistics on the prevalence of default Android encryption, knowledge of device processors, and relying on experience from other examiners with particular devices are all in play with regard to how to deploy EDL or any other exploit. Regarding user experience, I will give you a general rule regarding processor knowledge that will almost always prove true with regard to encryption. I will use three very common processors for this example. The MSM8916, MSM8909, MSM8917.

Generally speaking, of course…

- MSM8916 – not encrypted
- MSM8909 – may be encrypted or may not (refer to individual model)
- MSM8917 – is always encrypted

There can be exceptions regarding the MSM8916 and MSM8917, but statistically those assumptions above are true. There are many other processors that fall on the side of not encrypted with the MSM8916, or definitely encrypted with the MSM8917.

Methods for placing devices in EDL Mode

In general, the following options are available possibilities for placing devices in EDL Mode. Fastboot and ADB require unlocked devices, unless the user's pre-seizure settings or other exploits have or can enable them.

1. Key combinations – e.g. holding Vol Up and Vol Down (locked and unlocked devices)
2. Cable 523 (Cellebrite) or homemade EDL cables (locked and unlocked devices)
3. 'adb reboot edl' (unlocked devices)
4. 'fastboot oem edl' (unlocked devices)
5. DFU and FTM mode (locked and unlocked devices)
6. Test points (on some devices) (locked and unlocked devices)
7. eMMC fault injection (shorting) (locked and unlocked devices)

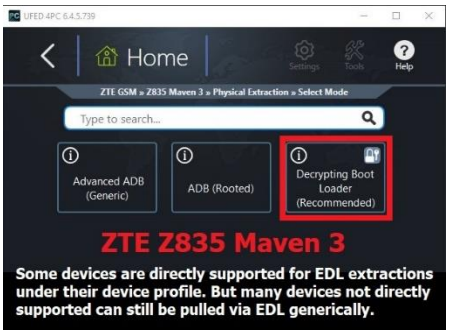
These seven methods are generally listed from least to most regarding level of invasiveness to the physical condition of the device itself – meaning phone disassembly or soldering. With regard to disassembly, especially for shorting, this is an area that can be risky if not carefully done. I highly recommend training in ISP, JTAG, Chip-Off, or device repair if possible. If not, I am a firm believer in practice and repetition. I have close to 600 phones in my inventory, most of which I have disassembled, repaired, chipped, used for JTAG and ISP, shorted and even cut into pieces.

I receive devices from multiple law enforcement agencies with varying degrees of training and differing operating procedures when it comes to seizing and storing digital evidence. I also receive a significant amount of damaged evidence or evidence that appears damaged but is in unknown working condition including items that have been stored for long periods of time. If the device I receive is not encrypted and cable #523 doesn't work, I may opt for Test Points (on some devices) or eMMC shorting before powering on the device or trying anything else, especially if I don't know the condition of the device or how it was seized. I may also try FTM or DFU without a battery to ensure I don't boot the phone. If I can pull it in the off-state without booting, I will do that first – even if it means disassembly and shorting.

Identify the device

UFED Direct Support

Check the device model for specific support for EDL extraction under the device's profile in the UFED (e.g. ZTE Z835). If the device is directly supported under its profile in the UFED, you can generally follow the instructions in the menus. If the device shows the Decrypting Boot Loader option, it is likely supported for EDL. Navigate through the menus under that profile and look for specific instructions for entering EDL Mode for that device.



Devices not specifically identified as supported in the UFED

Check the Device Processor (7-17-18)

If the device is not supported for EDL under the device profile in the UFED, it doesn't mean the UFED can't extract the device. Determine what processor it is running and check if that processor is one widely supported by Cellebrite for EDL extractions. **Widely supported** ***means that Cellebrite can use the EDL exploit to extract most devices running one of these processors***. There are exceptions, meaning there are a few devices running one of these widely supported processors that may fail. **Limited support** ***means that Cellebrite can extract selected devices running these processors***. There are also one-offs – devices running processors not listed in either category which may extract using the UFED's EDL exploit. Cellebrite will list known devices that fall on the list of limited support. There will be devices running limited supported processors and widely supported processors, not specifically listed by Cellebrite, which will extract via the EDL exploit in the UFED, using the generic menu selections. Using the UFED to try generic EDL extractions on devices not listed as supported via EDL will not harm the device. However, careless techniques used to create EDL mode can harm devices, so always use caution, research, and (whenever possible) test before attacking evidence.

Which Qualcomm processors can the UFED exploit via EDL Mode? (7-17-18)

Processors that can be exploited by the UFED via EDL Mode	
Widely Supported	**Limited Support**
MSM8909	MSM8996
MSM8916	MSM8917
MSM8936	MSM8937
MSM8939	MSM8940
MSM8952	MSM8953

Check for device encryption

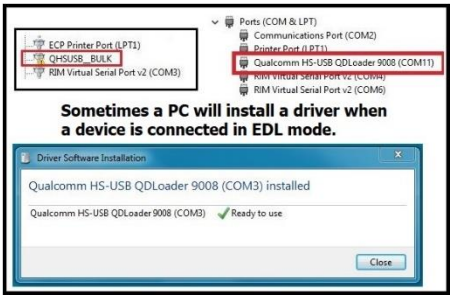
If the device was released running Android 6.X.X or higher, the device could be encrypted by default but there are many exceptions. Someone on the Mobile Device Forensic and Analysis Forum will likely know or have a good idea

about whether a device is encrypted or not. One of the benefits of EDL extractions is that trying different methods is generally harmless. So failing to extract, or extracting encrypted data, will only cost time. I see a significant number of new devices running Android 6 and 7 that are not encrypted by default.

Testing, creating, and implementing EDL Mode

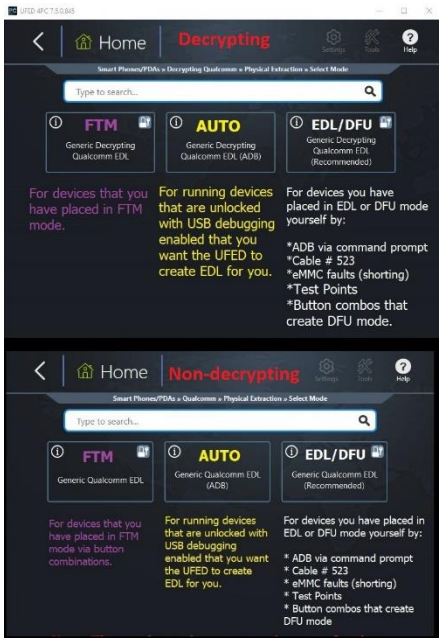
If the device is not directly supported in the UFED, but is running a processor that is supported for EDL extractions, you will have to test what methods will likely place the device in EDL Mode. For a list of button combinations and other methods to create EDL, open the UFED, select "Smart Phones/PDAs > Qualcomm > Physical Extraction" and navigate to the "waiting for device" screen. This will give you various instructions and methods of creating EDL Mode. I recommend highlighting and copying the instructions in that window and pasting them into a text or Word document. If you are using the UFED 4PC and are going to test for EDL using Device Manager, you will need to close UFED 4PC in order to see EDL Mode in Device Manager.

How do I check to see if the device is in EDL mode?



With most devices in EDL Mode, you will see a black screen and it may appear as if the device is off. The best way to determine if the device is in EDL Mode is by connecting the device to a PC via USB but with the UFED 4PC closed. If the device is in EDL Mode or if you place it in EDL Mode while connected to your PC, you should be able to see EDL Mode indicated in Device Manager e.g., "**Qualcomm HS-USB QDLoader 9008**…". Your PC may actually download a driver for EDL Mode if one is not available if you see, "**QHSUSB_BULK**".

Why would I test EDL in Device Manager before extracting a device?



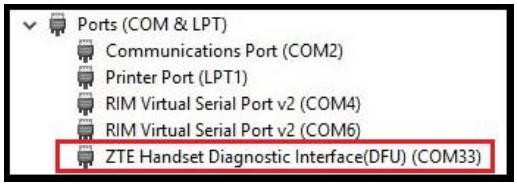
You don't have to test EDL on a device first. If you are sure the phone is supported, you can open the UFED, navigate to the "Waiting for Device" screen as seen in the screenshot, place the device in EDL Mode, and then hit "Continue" when you hear the handshake and the button becomes active. If the device was in fact in EDL Mode, it will likely pull with no issues.

However, EDL Mode is NOT the only thing that will trigger a handshake and activate the "Continue" button in UFED 4PC. Merely connecting a device with a battery and charging will most likely trigger this handshake, or the phone booting will trigger this handshake and activate the "Continue" button. Other modes available on phones like "DFU" and "FTM" mode will trigger a handshake. The "Continue" button can become active for several different reasons. If the phone is not in EDL Mode or if you put the phone in FTM Mode but fail to select the correct menu option, the extraction will fail. The diagram I created explains which menu option to choose, after you select the decrypting or non-decrypting method of extraction under the generic options.

So if the extraction fails, close the UFED 4PC, and try placing the device in EDL with Device Manager open. If your method places the device in FTM Mode, you will have to select a different menu option. By testing your method in Device Manager you can visually verify you are achieving EDL, DFU, or FTM Mode. You may just be connecting as an Android device. Once you verify this in Device Manager,

disconnect the device, open the UFED and navigate back to the appropriate "Waiting for Device" screen. Now perform the procedure you did during your test. If you have selected the appropriate menu options, the device should extract.

The diagram I created for how to extract a non-encrypted device will show you steps for testing ANY device for EDL. In other words, even if you are sure the device is encrypted but don't know how to place the device in EDL, use the steps in this diagram (EDL Non-decrypting Pulling Steps) to test your methods for creating EDL Mode. Once identified, employ the steps in this diagram (EDL Decrypting Pulling Steps) to perform the extraction on your encrypted device.

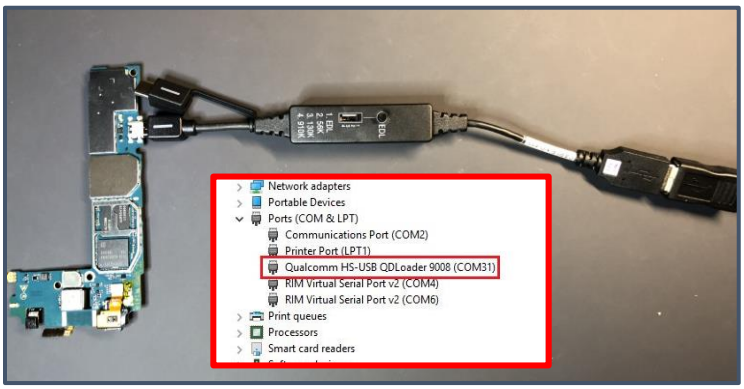


Typically EDL Mode results in an almost immediate audible handshake upon connecting a device via USB. There is sometimes a delay of a few seconds dependent on the device and your PC. DFU Mode also creates an almost immediate handshake and behaves much like EDL Mode regarding the audible handshake and immediate recognition in Device Manager. You may have already placed devices in DFU Mode and pulled them via EDL in the UFED without knowing it. From the examiner's perspective, the UFED treats DFU Mode just like EDL Mode. The menu options are the same and thus a device in DFU is generally no different than a device in EDL Mode. FTM does require a different menu selection (discussed later).

EDL Cables

In absence of known support for your particular device, I would recommend trying an EDL cable first. Try Cellebrite's cable #523 or another EDL cable if you do not have it. The device should be powered off for this test. If possible, remove the battery and test the cable using only USB power. If that doesn't work, try with the battery in the device. Be aware that if the cable does not place the device in EDL Mode immediately, the device may fully boot once connected to USB power if you have the battery inserted. This can be an issue if you are worried about the device being exposed to the network (not placed in Airplane Mode when seized, etc.). Most phones will not fully boot without a battery but there are some that will – some models boot with just USB power or will boot-loop – start to boot, then fail repeatedly. Remember you are doing coordinated steps. Just plugging in the EDL cable and pushing the button will not create EDL. Click on the diagram for the exact steps.

As described above with the ZTE Z812, not only is pulling a device via the non-decrypting method very easy and fast, it is also an effective way to test for EDL support for encrypted and non-encrypted devices with an EDL cable or eMMC shorting (covered later). Plugging most phones into USB with just the logic board (no battery) will do nothing. There are some phones that will try to boot but most will do nothing. Short those same devices with an EDL cable and the device will immediately handshake and be in EDL Mode when you release the EDL button. If I have a device that is locked and cannot be placed into EDL Mode via button combinations, this is the method I would use to test EDL. If EDL is not created immediately, I

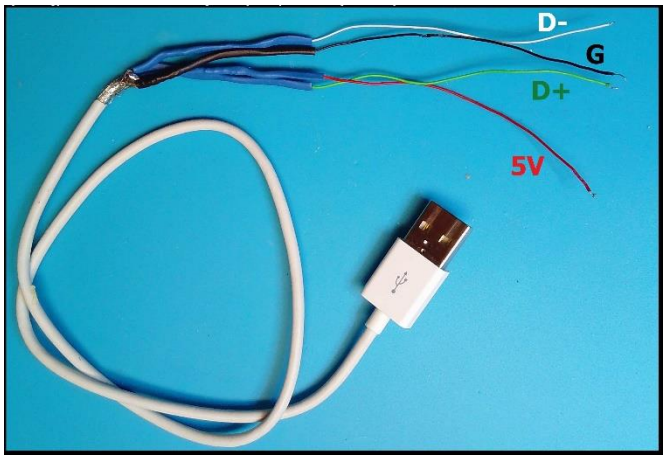


will try the same procedure with the battery in the phone. The procedure and order for pulling devices using the EDL cable is pretty straightforward. For a visual view of that procedure I am including a link to a visual flow chart that covers USING THE EDL CABLE or eMMC SHORTING from start to finish. How to create eMMC shorts on a device is also covered later in this paper.

Some phones will not be affected by the EDL cable at all. In those situations, pressing and releasing the EDL button may have no effect on the device. On some devices the EDL cable may sometimes cause a device to not boot properly or may trigger some audible handshakes when pushed, but the cable will fail to create EDL. Device Manager detects that a device is connected but pressing the EDL button caused the device to not be recognized. This means your coordination was off or the EDL cable will not work on that particular phone. The bottom line is that an EDL cable will just not work on some phones, just like button combos and ADB will not work on some phones.

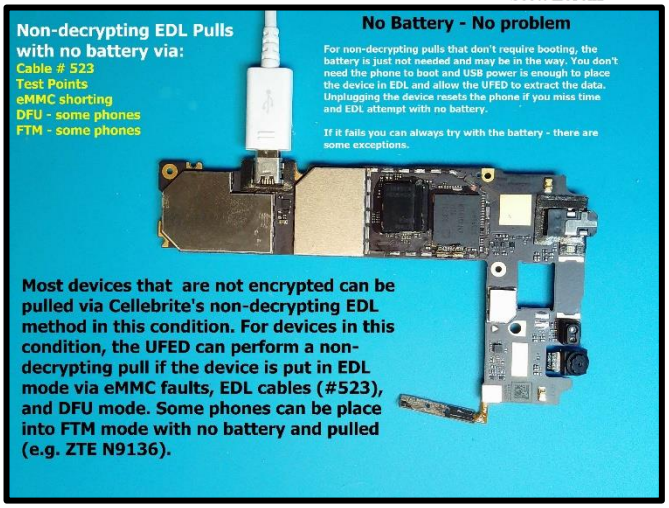
Using the EDL Cable "shorts" the device but it is not the same as creating eMMC faults

Using an EDL cable is "shorting" the device. By pushing the EDL button, a ground is applied to the D+ line (green line inside a USB cable). With Cellebrite's cable #523, that button is pushed and held while connecting the EDL cable to USB. When the button is released, EDL Mode is detected by the PC or the UFED. Do not confuse this data line (D+) with the Data lines that connect the eMMC to the processor. For those familiar with ISP, these are Data lines 0-7. They are separate, different, and not directly connected with the USB data lines. I will cover that in more detail later when covering USB testing and bypass.



EDL cables create EDL Mode by connecting ground to (D+) during connection to a USB port. When the ground is removed, the PC or the UFED will detect a device in EDL Mode – if that method is supported by the phone manufacturer. Phone manufacturers can disable this protocol and protocols that allow EDL to be created via key combinations, Test Points, and ADB. Creating eMMC faults (shorting) cannot be disabled and will generally always work on Qualcomm processors.

The ZTE Z812, was a non-encrypted but badly damaged device that I used to demonstrate the use of Cellebrite's cable #523. That example also demonstrated the use of the non-decrypting method of EDL extraction in conjunction with an EDL cable. In that situation, no battery is needed and the phone is not required to boot during the process. It is easy and fast and will work on most devices even if they are damaged. Most devices that are not encrypted, can be pulled in the same condition as seen with this photo of the Z899VL. This is because the non-decrypting method is low-level and does not require the device to boot.



An encrypted device can also be pulled with this method. Of course the user data is encrypted but when I am testing for device support, I will sometimes attempt to pull an encrypted device with this method just to make sure Cellebrite's EDL exploit will work on the device. I do this because the decrypting method of pulling phones can be challenging, even on supported devices. I want to make sure a failure to extract using the decrypting method is based on coordination and booting, and not because the device is just simply not supported. The non-encrypted pull is simple and easy, and if it doesn't work it usually means that processor and/or that device is not supported. If the non-decrypting method works, then I know I have a supported deivce. I will have to repull it using the decrypting method, which will generally require a battery and some reassembly for the phone to boot. Phones that will not go into EDL Mode with the EDL cable can still be shorted into EDL via eMMC shorting or Test Points.

Button Combinations

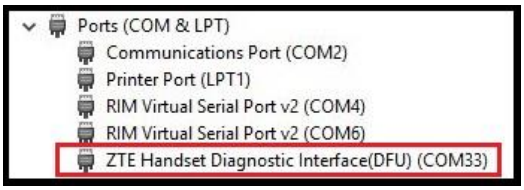
If you have a device in an unknown condition or are worried about the device not being in Airplane Mode, you should conduct this test in a Faraday environment as attempting button combinations may fully boot the device if you miss a step or are off on coordination.

I covered two basic extractions via EDL in the EDL webinar in February. A PDF of that PowerPoint presentation is in the resource folder in the Mobile Device Forensics and Analysis forum. The "Resources Folder" contains many of the tools and guides needed for EDL extractions, including write-ups on specific devices and ISP pinouts for many devices supported for EDL extraction. The Cellebrite EDL Webinar PowerPoint is located there also, inside the EDL Extractions folder. In that PowerPoint I cover basic extractions on the Alcatel 5044R using button combos to create EDL and the ZTE Z812 using the EDL cable #523 to create EDL.

Without specific knowledge and experience with a particular device, the easiest method to extract a device via EDL Mode is if it is directly supported under its device profile in the UFED. In the case of the Alcatel 5044R, EDL can be easily created via button combos and because that specific device is supported in the UFED, simply follow the instructions and it will walk you through the process. Even though that device was encrypted and pattern locked, the process is easy and automated with minimal user interaction.

What do button combinations really do?

Button combinations may not directly place the deivce in EDL Mode but instead place the device in DFU or FTM Mode. Cellebrite then uses those modes to exploit the device and place the device in EDL Mode. Duplicating the button combo steps with the UFED closed will allow you to see the condition of the phone in Device Manager. This is important to understand because button combinations can be used generically to extract devices that are not directly supported in the UFED. Knowing what mode the button combinations create will be important so that you know which menu selection to make with generic Qualcomm methods. That will be discussed in detail in the section covering DFU and FTM modes.



Automated ADB via UFED's direct support or generic support with unlocked devices

Devices that are unlocked or the passcode is known, means the examiner can enable USB Debugging, Stay Awake and "Unknown Sources" under Developer Options. With all of those settings enabled, Cellebrite can then use ADB to place the device in EDL Mode in order to take advantage of their EDL exploit on supported Qualcomm processors. Cellebrite can pull devices running processors supported for EDL under the generic Qualcomm method for both decrypting and non-decrypting methods. I recommend using these methods (decrypting or non-decrypting) if you have an unlocked device.

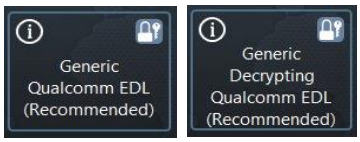
When using the automated ADB method, as long as you have enabled USB Debugging, Stay Awake, and Unknown Sources on the device, Cellebrite will initiate EDL for you. Cellebrite will also pull the device with the user lock in place. In other words, you can unlock the phone with a known passcode and enable USB Debugging and other options, but it was not necessary to disable or remove the user lock on devices I tested. By the time the UFED reboots the device to extract, the exploit is already loaded. You will see the device boot to the user lock screen, but no further action or unlocking is needed.

For the automated ADB method, just follow the instructions in the UFED prompts. You may be required to disconnect and reconnect the USB cable during this process but the UFED will handle the process of creating EDL via ADB. If you have a device seized in working order with access to Developer Options, and it is running a processor supported for EDL, I would recommend this method as a first try, especially for encrypted devices.

Using the automated function means that you don't have to worry about coordinating Command Prompt with the UFED 4PC, or moving your device from your PC to the UFED Touch after creating EDL via Command Prompt. For a more detailed explanation of those two methods, see "EDL Mode Via ADB with UFED 4PC Generic and Manual". That PDF is located in the resources folder on the Mobile Device Forensics and Analysis forum. I use the non-encrypted selection in that paper, but the steps are the same for the decrypting method - just a different menu selection.

Manual ADB or Fastboot used by the examiner to create EDL

For some users who just like to do things for yourself, you can use ADB via Command Prompt to create EDL Mode using "**adb reboot edl**" command. Users may want to verify a device is actually entering EDL Mode if an automated extraction fails or to verify this command will work on your PC before attempting this extraction with the UFED Touch.



You can also create EDL manually using ADB via Command Prompt and extract the device via the UFED. This is a different procedure than allowing the UFED to automatically create EDL via ADB. That means different menu selections. If you choose to create EDL yourself via Command Prompt or Fastboot, **DO NOT** select the UFED's ADB method of extraction. The **generic decrypting or non-decrypting method** of extraction should be selected for any method that involves you creating EDL outside of the UFED. That includes button combinations, shorting to create

eMMC faults, Test Points, EDL cables, or EDL created by ADB or Fastboot using Command Prompt. When you create EDL yourself, don't tell the UFED to do it via ADB – it will fail. I have a specific EDL write-up for those control freaks who just want to control the world via Command Prompt (EDL Mode Via ADB with UFED 4PC Generic and Manual).

****Note** – Some devices will immediately trigger a handshake sound when rebooting to EDL after the "**adb reboot edl**" command and you will see the device in EDL Mode in Device Manager. If the device reboots to a black screen but does not create a handshake or appear in Device Manager, disconnect and reconnect the USB cable from the PC. The phone is likely in EDL; the PC just can see it. This should trigger the handshake and refresh Device Manager. The phone remains in EDL Mode even after disconnected from USB because you presumably have a battery in it if using ADB. *Devices placed in EDL Mode will generally remain in EDL Mode until all power is removed (battery included), or they are forced to restart*.

Devices not supported for entering EDL via ADB or Fastboot commands

Some devices running processors supported for the EDL exploit, will not go into EDL Mode via an ADB command – either by your command or Cellebrite's automated process. On those devices, the **"adb reboot edl"** command will reboot the device, but instead of rebooting to a blank screen (EDL), the device will just fully boot as normal. You can test this by closing the UFED 4PC, connecting your unlocked, enabled, and booted device and then giving the command. If the device reboots to a black screen, it will likely be in EDL Mode. Check Device Manager. Failing to reboot to EDL does not mean the device cannot be extracted with the UFED. It just means you will have to find another method to create EDL.

FTM and DFU Mode to create EDL

Some devices can be placed in Field Test Mode (FTM) Mode or Handset Diagnostic Interface (DFU) Mode with button combinations and the UFED uses those conditions to exploit them. This is still the EDL exploit at work, it is just a different path to achieve EDL. The ZTE Z835 Maven 3, is directly supported for the EDL exploit in the UFED. When navigating to this device profile you will see instructions for button combinations to perform the extraction. However, there are several different paths that can be used to extract the Z835. The Z835 Maven 3, was sold as an AT&T prepaid phone and was running Android 7.1.1, but it was not encrypted by default. The UFED pulls the Z835 under its supported profile using the decrypting method. We know this because the extraction process used 6 steps and an execution phase. The device is also required to fully boot. Because we know the Z835 is not encrypted by default, we can also extract a full unencrypted physical using several non-decrypting methods available in the UFED.

*Using the **decrypting method to pull phones that are not encrypted** will not harm the evidence and is forensically sound.*

ZTE Z835 using DFU Mode

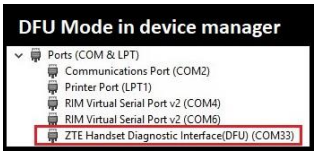
ZTE's Handset Diagnostic Interface is normally achieved by pressing Volume Up and Volume Down while connecting the device to USB. This is the same instruction given under the Z835 supported device profile in the UFED. On devices I tested and including the Z835, DFU is created immediately with an audible handshake very much like eMMC shorting and Test Points. On devices I tested, devices can enter DFU without the battery connected and using the non-decrypting EDL method, the device can be extracted without the battery. The extraction is identical to shorting a device that is not encrypted and pulling it without the battery. So even though the ZTE Z835 is directly supported in the UFED under its device profile for a decrypting EDL extraction, my experience with the device tells me that it is not encrypted by default and can be pulled under generic options using the non-encrypted method. I use the same button combos described under the Z835 profile instructions – hold Volume Up and Volume Down while connecting to USB. DFU will result almost immediately and trigger an audible handshake. Because we are selecting a non-decrypting pull, the

*Pulling an **encrypted device with a non-decrypting extraction** will not harm the phone or the evidence, but the data extracted will be encrypted. If that happens, pull the device again using the decrypting method.*

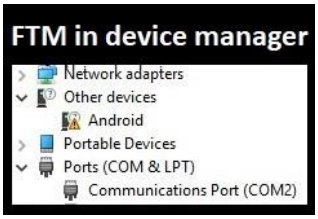
device will begin to dump after 2 steps and, most importantly, will not boot during the extraction.

How do I know if my device is in FTM or DFU Mode?

Checking to see if your device is in FTM, DFU, or EDL is all done the same way – by checking Device Manager. The device screen will most always be black for DFU and EDL Mode. FTM mode will generally have a screen that identifies it is in FTM. But each of these modes looks different in Device Manager. Most phones can be placed in DFU and EDL Mode with no battery in the device. FTM requires a battery for most devices. DFU and EDL Mode are similar in many ways, from the examiners perspective. DFU and EDL Mode will generally be created almost immediately when the condition designed to create EDL and DFU is applied and the device is connected to USB.



With EDL for example, applying an eMMC short will create an immediate handshake when connecting to a PC. The device will immediately appear in Device Manager as in EDL Mode. The same is true with DFU Mode on devices I have tested. With the device in the off-state (with and without a battery), holding down Volume Up and Volume Down together and connecting to USB at the same time will create DFU almost immediately. You will hear an audible handshake and Device Manager will show the device in DFU Mode. So if you weren't looking at Device Manager, the reaction of your PC (audible handshake) and the appearance of the device appear to be the same with EDL and DFU Mode.



Not only do EDL and DFU cause your PC and the device to behave the same way, the generic menu selections for devices in EDL Mode and DFU Mode are exactly the same. In other words, your menu choices for selecting the Qualcomm (recommended) selection for either encrypted or non-encrypted devices will be the same. From the examiners point of view, it is really not even necessary to know if you have the phone in DFU or EDL Mode because the UFED will ultimately use both of those modes to achieve the same result. Examiners need to know this because those unfamiliar with DFU and how it works may not start an extraction thinking something is wrong. An examiner may think they have done something wrong or think the phone will not extract if they check Device Manager expecting to find EDL Mode and instead find DFU. For the decrypting and non-decrypting method of extraction in the UFED, they are both the same.

ZTE Blade Spark placed in FTM by holding Volumn down and connecting to USB.

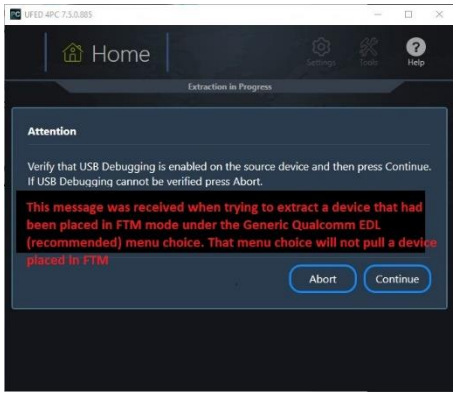
FTM behaves differently and requires separate menu selections than DFU and EDL

With FTM, a battery is needed, but exceptions exist. FTM is reached by holding either the Volume Up or Volume Down buttons while connecting the device to USB. The behavior of the device is different with FTM as opposed to DFU or EDL. With DFU and EDL, the examiner gets immediate feedback and a result. When attempting FTM, starting with the device in the off-state with a battery inserted, press and hold either the Volume Up or Volume Down button (not both) and connect the device to USB. The phone will appear to begin to boot as

FTM can also be achieved by holding vol-down or vol-up while holding he power button with the battery inserted. This is before connecting to USB. The phone may start to boot once and restart once before success.

you will see the ZTE booting process begin. Continue to hold the volume button you chose. You may have to hold that button for up to 20 seconds on some devices I tested. If supported and you selected the correct volume button, the phone will display FTM as shown in the example of the ZTE Blade Spark created by holding the Volume Down button. Device Manager will also indicate the device is in FTM on devices I tested. I also checked Command Prompt and it showed a device with a question mark where you would normally see the device recognized by serial number.

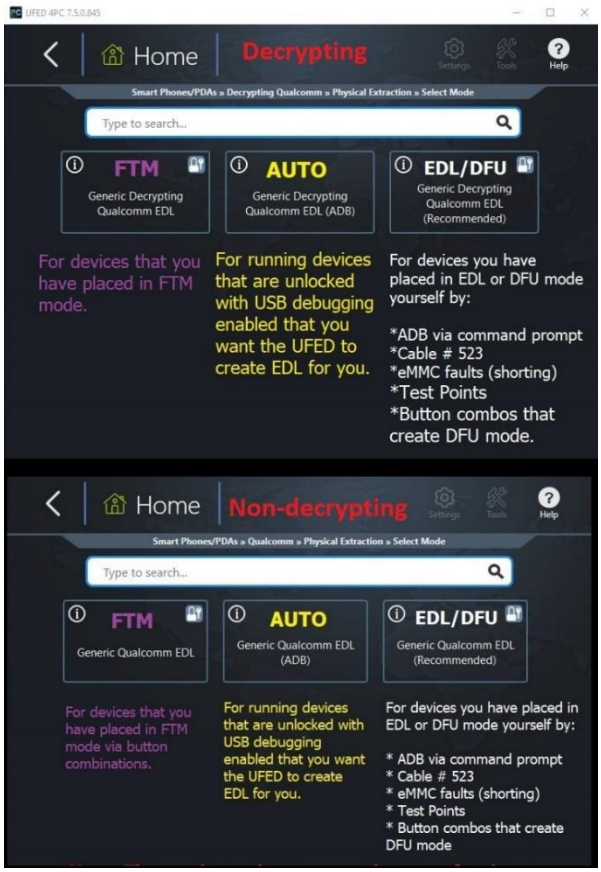
FTM method for generic EDL extractions



Unlike DFU and EDL, which are seamlessly interchangeable with menu selections in the UFED, FTM requires the use of a rarely used button in the UFED workflow of extraction steps. So when placing a device in FTM Mode and attempting an EDL extraction of a device under the generic Qualcomm options, the correct button must be selected. Note that these options are the same and you must make the correct selection under decrypting or non-decrypting method. Selecting the Generic Qualcomm (recommended) method for a device that you have placed in FTM will fail. Instead of going to steps and 1 and 2 and then to a binary dump, you will receive a message to enable USB Debugging. That message is incorrect, given what you are trying to do and what just occurred. For FTM extractions, it means you selected the wrong menu option.

Why doesn't Cellebrite just tell me what button to push all the time?

Cellebrite is probably the most user friendly mobile forensics programs on the planet (my opinion). Most of the time the UFED does tell us what button to push. But there are some things that are out of Cellebrite's control and must be left to user discretion and selection. There are so many ways to create EDL Mode and many of those methods must be done outside of the UFED and involve significant, physical manipulation and disassembly of the device. If the choice is to only include directly supported devices with explicit instructions on each device that leaves no room for error or a more flexible tool to that allows experienced users to find their way – I choose the latter.



I created the following diagram to help encapsulate the menu selections based on the different methods of creating EDL, either by you or the UFED. Most of the EDL extractions I do are done under the Generic Qualcomm EDL menus, because I prefer to select my method of EDL and I am comfortable with all of the Generic Qualcomm solutions and options. There are so many devices running Qualcomm processors that it would be impossible to field test each one. Note that those menu choices are true for both all Generic Qualcomm EDL extractions to include the decrypting and non-decrypting methods. So after you select decrypting or non-decrypting, the menu choices are the same after that. The UFED can use its decrypting EDL tool with devices placed in EDL, DFU or FTM by all of the techniques in that diagram.

Shorting a device into DFU Mode

There is one more issue I will cover in this section, even though it can be covered under eMMC faults as well. I have discussed using eMMC faults to short devices into EDL Mode and various strategies for doing that. There are other things that can occur when shorting devices that are not EDL but may go unnoticed or may cause an examiner to believe they have done something wrong. Later in this paper I go into which ISP points can be shorted to create eMMC faults. I have put out some diagrams in the past that indicate shorting all ISP points create EDL Mode. That included D0-D7, CMD and CLK. There are some exceptions I encountered a while back and explored more when testing devices for this paper. Shorting the CLK pin on some phones, will cause the device to go into DFU Mode instead of EDL Mode. When that occurs it is true for both sides of the CLK pin. It only happens on some devices and I have not tested the CLK pin on enough devices to put a percentage on it but I have verified it repeatedly.

So the good news is that it will not affect an EDL extraction. As we know now, from the point of view of the examiner, the UFED will handle a device in DFU Mode the same as a device in EDL Mode. Of course

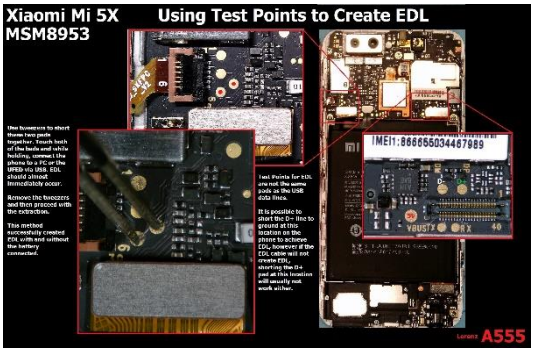
the UFED is doing more behind the scenes, but an EDL extraction will occur with the exact same menu selections. It is worth mentioning because I have advocated that some examiners short the CLK pin on some devices simply because it was more accessible given the board layout and heat shields. A user who wants to test his or her shorting result on the CLK pin during an extraction, might be confused when they look in Device Manager and find that they have shorted the device into DFU Mode. No worries, it's normal and it will extract.

As an academic exercise, I tested CLK shorting on the following devices and you see the results below. Other ISP pins on these devices produced EDL. Only the CLK produced DFU. There is a link attached to each device with access to the diagram. It doesn't affect an EDL extraction so it is just FYI.

Phone	Processor	FTM	DFU	CLK short
ZTE Z971 Blade Spark	MSM8917	Vol Down - connect USB	Vol Up & Vol Down - connect USB	EDL
ZTE N9136 Prestige 2	MSM8909	Vol Up - connect USB	Vol Up & Vol Down - connect USB	DFU
ZTE N9560 Max XL	MSM8940	Vol Up - connect USB	Vol Up & Vol Down - connect USB	EDL
ZTE Z839 Blade Vantage	MSM8909	Vol Down - connect USB	Vol Up & Vol Down - connect USB	DFU
ZTE Z835 Maven 3	MSM8909	Vol Down - connect USB	Vol Up & Vol Down - connect USB	DFU
ZTE Z852 Fanfare 3	MSM8909	Fail	Fail	DFU

Test points to create EDL Mode

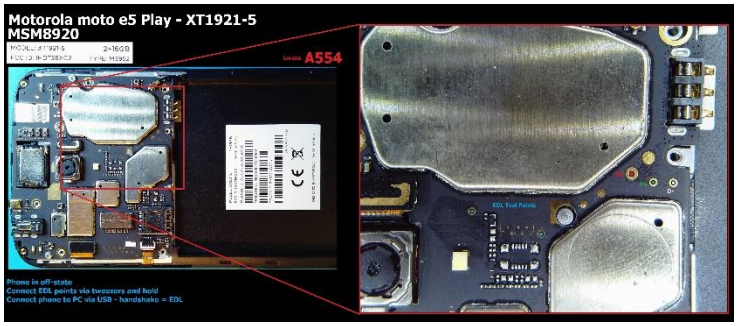
Test points are installed on some boards for the purpose of allowing access to EDL Mode. Although I have this method listed under shorting, actual Test Points are not shorting one pin to a ground on the phone as is the case with the EDL cable or eMMC shorting, which both create EDL by shorting data, clk, cmd, or D+ to ground. On phones I have researched and located Test Points, neither of the points is ever ground. Either one or both test for voltage (usually 1.8 volts).



As an example, I used the Xiaomi Mi 5X, running the Qualcomm MSM8953. This device is supported for Smart ADB in the UFED if unlocked and that method resulted in a successful physical extraction. A quick search of the internet reveals that many Xiaomi devices have Test Points that allow them to be shorted into EDL Mode. Most of the time shorting Test Points can be done with a pair of tweezers. Using the tweezers to create a bridge between the two points and then connecting the device to USB while holding the tweezers on each point. The device will almost immediately go into EDL Mode just as if you used the EDL cable, eMMC shorting, or DFU. My tests on the Xiaomi worked with and without the battery connected. Neither of the Test Points is a ground pin. One of the pins registers 1.8 volts with the battery connected and the other registers no voltage. *Note that Test Points create EDL Mode due to instructions provided in the bootloader code. Thus, they can be turned off or blocked (unlike eMMC faults) just as ADB and Fastboot commands to create EDL can be turned off.

Looking and probing for Test Points

On many brands, Test Points look like every other pad on the phone. Sometimes the presence of two pads near each other in a particular location can be a hint, but there are many round pads on all types of phones all over the board. An example of some typical Test Points can be found on the Motorola XT1922-7 moto g6 Play (MSM8920). You can click on the diagrams for a blown-up image. Note the Test Points I have marked, I picked to test because of their proximity to each other and location. I first tested them for voltage and ground. I look for pins that are not ground and may or may not have voltage. I test ground points with the phone in the off-state with no battery connected. I test for voltage on Test Points with the battery connected. Once I have my suspected pins, I disconnect the battery and USB. With no power to the phone, I place the tweezers on my suspected pins as shown in the Xiaomi diagram. With tweezers in place, I plug the phone into USB. If you have the correct points and your tweezers haven't slipped, EDL Mode will be almost instant. You will hear an audible handshake and Device Manager will register EDL.



So while proximity gave me a hint on the XT1922-7, another Motorola phone did not position the Test Points so obviously. The Motorola Moto E5 Play XT1921-5 (MSM8920) had me puzzled for a bit. The Test Points are positioned unusually far from each other. My reason for trying those two pins was that neither was ground and one was voltage. Click on the diagram for a blown-up view.

USB Taps are not Test Points

If you notice, I mark the USB taps on many of my diagrams to include both of the Motorola diagrams here. There are a couple of reasons for that. One reason I began marking USB taps is for USB bypass operations on damaged devices (discussed later). The other reason is to help distinguish between USB taps and EDL Test Points. I have seen diagrams on the internet marking USB taps as EDL Test Points. While USB taps can create EDL, they are very different. EDL created via USB taps is accomplished by shorting the D+ tap to the ground tap. When you do that with the USB taps on the logic board, you are essentially becoming an EDL cable. EDL cables do the exact same thing, they just short D+ to ground inside the USB cable. If the EDL cable doesn't work, shorting the USB taps won't work either (on all phones I have tested). EDL cables don't work on either of those Motorola phones, but they both have Test Points that create EDL. They can also both be forced into EDL via eMMC faults like most phones.

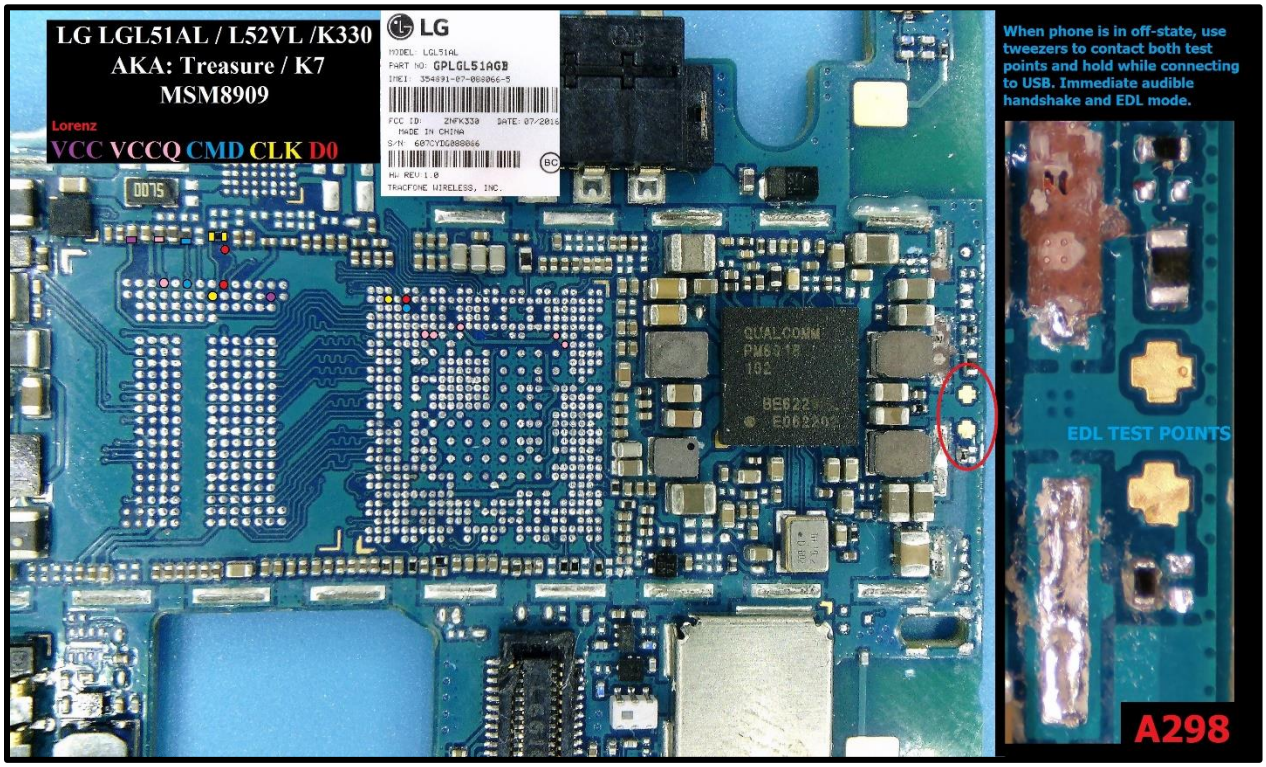
LG phones and Test Points – look for the sign of the cross

Many LG phones make finding Test Points easy. You will likely be stunned on how many LG phones have Test Points – old and new alike. All of them I have found have all been in the shape of a cross. No other pins on the phone look like this. So the proximity and shape make them a dead give-away. All EDL taps work the same way, connect the two points (tweezers) and the plug into USB. I have listed many LG phones in a table coming up. You can follow the links to those diagrams for example of LG Test Points.

Which is easier – eMMC shorting or Test Points?

EDL Points are easier than eMMC faults because no soldering is needed, the targets are bigger, and Test Points are generally found in accessible locations. Many times, you will not have to flip the logic board to reach the storage or processor. As with anything else, there are exceptions to that rule. If the Test Points are located on the flip side of

a board, you may have to get creative when deploying the decrypting bootloader as you will need the battery in place for the phone to boot. So there are a few phones that eMMC shorting may be preferable, logistically, even though Test Points are available.



List of sample phones with Test Points – links to diagrams included

I am including a list of some of the phones I had in my inventory with Test Points. Most of the phones on the list are LG. Many phones don't have Test Points so with some brands, you may never come across Test Points. Each of these phones has a link to the diagram with Test Points and ISP points. As you can see from the list of phones, the LG phones have Test Points on older phones and newer phones. Some of these phones are not supported for EDL extraction by the UFED – yet – but will be. Many of the processors shown are on the potential up and coming list of limited support by Cellebrite.

Phone	Processor	Test Points
LG L64VL X Charge	MSM8917	Yes
LG LS770 Stylo	MSM8916	Yes
LG H811 G4	MSM8992	Yes
LG-H860 G5	MSM8996	Yes
LG H918 V20	MSM8996	Yes
LG L52VL K7 / Treasure	MSM8909	Yes
LG L61AL K10	MSM8909	Yes
LG K428 K10	MSM8909	Yes

LG K120 K4	MSM8909	Yes
LG K371 Phoenix 2	MSM8909	Yes
LG L33L Sunset	MSM8916	Yes
LG L44VL Rebel	MSM8909	Yes
LG LS676 Tribute	MSM8909	Yes
LG LS777 Stylo 3	MSM8940	Yes
LG LS992 G5	MSM8996	Yes
LG M150 Phoenix 3	MSM8909	Yes
LG MS210 Aristo K8	MSM8917	Yes
LG L82VL Stylo 2	MSM8916	Yes
LG M257 Harmony K20	MSM8917	Yes
LG LS665 Tribute 2	MSM8916	Yes
Motorola XT1922-7 moto g6 Play	MSM8920	Yes
Motorola XT1921-5 moto e5 Play	MSM8920	Yes
Xiaomi Mi 5X	MSM8953	Yes

Creating eMMC faults (shorting)

As I stated earlier, Cellebrite always attempts to make the extraction process as automated as possible for any method used via the UFED. However, despite their best efforts, a certain degree of examiner interaction with a device is necessary with many extractions and especially with certain EDL extraction techniques. In order to take full advantage of Cellebrite's EDL exploit on as many devices as possible, it is necessary for the user to do more than just button combinations, EDL cables, or navigating menus in device settings on the target device. In fact, with EDL extractions it may be necessary to become intimately familiar with the device. Some methods require a significant amount of research, testing, disassembly, and some soldering.

Initiating eMMC faults, also referred to as "shorting", is one of the most reliable methods to create EDL. To be clear again, using an EDL cable is shorting a device. EDL cables simply create a short in the USB cable by grounding the (Data +) line. However, do not confuse the (Data +) line in a USB cable with Data lines connecting the eMMC to the processor. They are not the same.

Creating EDL by shorting the exposed CMD, CLK, or Data locations on a device will normally always work whether or not other methods of creating EDL on that same device have failed. So if another method of placing a particular device in EDL works, eMMC shorting will also work on that device. If no other method of placing a particular device in EDL works, shorting will most likely still work. Most devices running Qualcomm processors can be placed in EDL Mode via eMMC shorting. Whether or not they can be extracted once in EDL Mode is a separate issue. I have yet to find a Qualcomm device that I could not place in EDL via eMMC shorts. That doesn't mean there can't be a significant number of devices that can't be shorted. I just haven't come across one yet. I have come across some devices in which locating places to short was very difficult and even more difficult to short once I found them. Click on the Motorola XT 1650-02 as an example. That phone is running an MSM8996, and locations that worked on other phones running that processor were not creating EDL for me. The phone was also running Universal Flash Storage (UFS), which doesn't prevent shorting but it is not like eMMC storage (discussed later).

Don't forget to remove the short

Shorting (eMMC) is the only method that requires the examiner not only to create the condition that produces EDL, but then **requires the examiner to remove that condition before proceeding with the extraction**. With other methods requiring the user to create EDL, the action that created EDL (button combos, cable #523, or Command Prompt) creates a temporary or momentary condition that creates EDL, which Cellebrite can remove during its extraction process. Cellebrite's cable #523, has a spring-loaded pressure release button to create the short that produces EDL. EDL is visible when the button is released, thereby allowing the UFED or Windows to detect a device in EDL Mode connected via USB. The phone is in EDL Mode, but the short is removed by releasing the button.

With eMMC shorting, you must create the short, apply power that creates EDL Mode to the device, and then remove the short from the device after EDL is created. The device will not pull while shorted even though it is in EDL Mode. Thus devices that are damaged and are permanently in EDL by accident or intentional destruction, will not be extracted until that short or damage is located and removed or repaired. ***A common error for EDL extractions is that examiners either forget to remove the short that created EDL, or have accidentally damaged a device or created a bridge when soldering on the device. When this happens and you hit "Continue", the device will generally not proceed past Step 1, and will eventually fail with an extraction error***. Failing to remove the short has never permanently damaged a device for me, but you will have to start the process again. Disconnect the device, remove battery or reset, and then short again. See these PP slides (Shorting eMMC – Solder & Clip Method).

Phone disassembly

Depending on the model of the device, there can be significant disassembly involved with this method of creating EDL. With some devices, it may be necessary to use heat to remove screens or back covers of devices, in order to access the logic board. The heat shields on some devices can also be robust and require heat and/or destruction to remove. To illustrate this further, watchMoto Heat Shield Removal Video and Moto Heat Shield Removal Close Up. (See Moto Heat Shield Removal Video). This is where damage to a device can occur. Resistors and capacitors can be accidentally dislodged when removing heat shields or unseen shorts can occur. On some devices, Data, CLK, and CMD lines run just below the surface of the logic board under a thin coating. These lines can be severed by careless use of a razor or other tool used to pry off heat shields.

Experience with phone disassembly and repair, ISP, JTAG, and Chip-Off is helpful for this method. Generally speaking, soldering is necessary in most circumstances. I recommend using a microscope for this procedure to ensure accuracy and to avoid inadvertently damaging the device.

Shorting ISP points

Many examiners have not had ISP training so I understand the confussion and questions I get regarding "shorting". So generally, when the term "shorting" is used, it is referring to shorting the same locations examiners use when pulling a device via In-System Programming (ISP). These are not the same locations used for JTAG proceedures. JTAG taps will generally not create EDL Mode when shorted. That may also be confussing because JTAG taps look like Test Points, found on some devices, that can be used to create EDL Mode.

Shorting the Data lines connecting the processor to the eMMC or shorting the Command or Clock line is the most effective, reliable way to create EDL on devices running Qualcomm processors. When EDL cables, button combinations, and ADB commands fail to place devices in EDL Mode, eMMC shorting will most likely always work. Whether supported for extraction or not, most Qualcomm processors can be forced into EDL Mode via eMMC

shorts. And by the way, that also includes shorting devices with Universal Flash Storage (UFS) chips found on newer phones.

Why ISP, JTAG, and Chip-Off training and techniques are still relevant

Wth more and more devices encrypted by default, fewer examiners receive training on ISP, JTAG and Chip-Off. Also as a result of encryption, fewer pinout diagrams of devices are created because pulling the data via ISP and Chip-Off will result in an encrypted extraction, if the phone is encrypted. However, there are good reasons why training in these areas is very useful and relevant today:

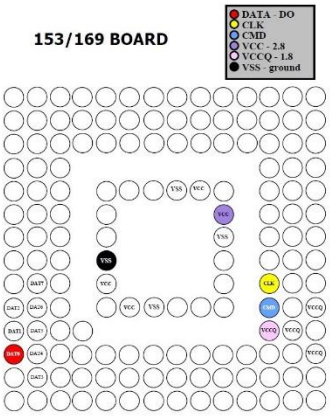
- Many phones released with Android 6 and 7 are still released without being encrypted by default.
- Feature phones that can be exploited by JTAG are still manufactured and sold new today.
- The world is full of older devices that are not encrypted and still in use or are not in use but contain evidence of a crime. These phones can be exploited by JTAG, ISP, and Chip-Off.
- Many EDL extractions require phone dissassembly and shorting ISP points to create EDL Mode.
- Certain flasher boxes require an ISP connection to a device to remove FRP locks from devices and other functions.
- Chipping and pinnning modern processors will allow you to access badly damaged devices using exploits like EDL – even if the devices are encrypted (examples to follow).
- The skills and equipment for ISP, JTAG, and Chip-Off are necessary to take full advantage of exploits like EDL.

The Z799VL aka Z798BL or Z899VL is a phone that was challenging and generally untouchable (if locked) for a while before Cellebrite's release of EDL extractions. These are the Majesty Pro and Majesty Pro Plus devices. The Z799VL and Z798BL were encrypted by default, while the Z899VL was not encrypted by default. They all have the same logic board and pinout and all must be shorted into EDL Mode because no other methods work to create EDL with these three models. Before the EDL exploit, I was able to access data on the locked Z899VL via ISP or Chip-Off. Having the ISP pinout for these devices and knowing how to dissassemble and short them, meant I could take immediate advantage of the EDL exploit released by Cellebrite.

Pinning devices and locating EDL short points

To get a good idea of how devices work behind the scenes I always find it beneficial to understand where everything is located and how it is connected. To do that, I like to take test devices and chip and pin them, going beyond the usual ISP pinouts. I have done that with Majesty Pro series of devices. Pinouts showing the locations of ISP points have proved useful for EDL extractions. EDL cables, button combinations, and ADB commands will not work on all phones. When those options do not create EDL Mode, "shorting" or creating eMMC faults will be necessary.

Pinning devices for ISP can be done by chipping off the memory (eMMC) of a test device and using a multimeter to trace known locations under the eMMC to exposed locations on the logic board, so that direct access to the eMMC can be achieved without removing the chip on the target device (the evidence). These pinouts are to what examiners are referring when asking for "pins" or a "pinout" of a particular device, either for an ISP extraction or locations to "short" for EDL extractions. There are many different types of storage and BGA layouts. There are three primary BGA patterns found on the vast majority of devices. Clicking on the BGA diagram will take you these diagrams in the MDFA resource folder.



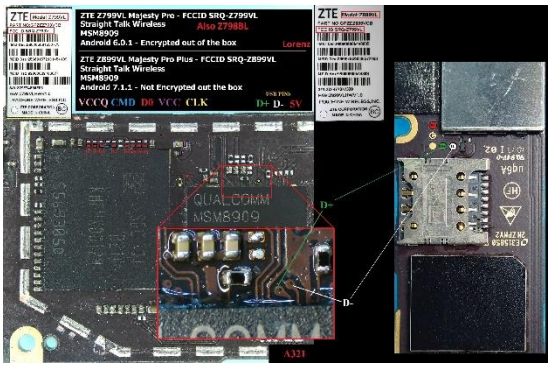
The pins needed for an ISP pinout include the Data0 or **D0**, Command (CMD), Clock (CLK), VCCQ (1.8 volts) or VCC (2.8 volts). These same locations are present under the processor with the exception of the VCC, which is usually not found under the processor. For purposes of EDL, VCC and VCCQ should be avoided.

Shorting voltage pins can produce EDL Mode, but it can be dangerous as well. For extracting devices via ISP, Data0 (D0) is the only data line that is needed, however there are eight (8) data lines (D0 – D7). There are some flasher boxes that can take advantage of the other data lines to extract data faster from a device via ISP. The vast majority of pinouts available today, only include the Data0 (D0) pin locations.

Finding ISP points without chipping a test device

Kim Thomson wrote an excellent paper on ISP Recognition. That is, locating ISP pins without chipping a device (ISP Recognition – Kim Thomson). Most of the time I can find a location to short devices into EDL without chipping or seeing a pinout. I do that using some of the techniques Kim talks about in his paper and just recognizing patterns I know exist on particular phones and processors. Many devices that run the same processor have similar patterns regarding board layout and ISP locations. I created a couple of diagrams that show patterns that exist on devices running the MSM8909 and the MSM8916.

Which ISP points create EDL?



All of them can. When looking at an ISP pinout, you will generally have a choice of possible locations to short to create EDL Mode on the phone. All ISP pin locations (D0-D7, CMD, and CLK) can create EDL when shorted. The CLK pin can create DFU Mode on some phones as I discussed under the FTM and DFU section of this paper. In many of my previous posts and papers, I often use the command (CMD) location to short to create EDL, but it is not necessary to restrict yourself that that pin. You can choose a pin that is easiest to access and solder. I have not tested every pin on every phone but I purposely marked and tested all of them on ZTE's Majesty Pro series of phones. All of the locations (D0-D7, CMD, and CLK) produced EDL when shorted. As you can see by the diagram, the data pins are easier to access and solder than is the CMD pin. When it comes to EDL Mode, there are no "degrees" of EDL. EDL created by the CMD pin is no different than EDL created by the CLK or Data0 pin.

What is the difference between the ISP data lines and data (D+) line in a USB cable?

As shown in the Majesty Pro diagram, I have marked the USB taps. Regardless of the type of USB cable, whether it is micro USB or USB-C, there are only 4 lines of concern if I want to exploit or bypass USB for device extractions. The D+ and D- line communicates with the processor and runs from the USB port to the processor. The D+ line is the green line inside a USB cable and I mark it as green in my diagrams. The D- line is white and I mark it accordingly in my diagrams. The D+ line is the line that is shorted to create EDL Mode on devices that support that protocol. But that protocol can be turned off by manufacturers and that is why EDL cables don't work on all devices.

The Data lines D0 -D7, run between the device storage and the processor. So when the processor receives information from the device storage (eMMC or UFS for example) that data is sent out the USB port through the USB Data lines. Information from the device storage goes through the processor and out through the USB data lines to the UFED. Shorting the Data lines between the processor and the storage is referred to as creating eMMC faults or shorting. Shorting those lines, along with the CLK and the CMD, are the methods most reliable for creating EDL.

Shorting procedures and equipment

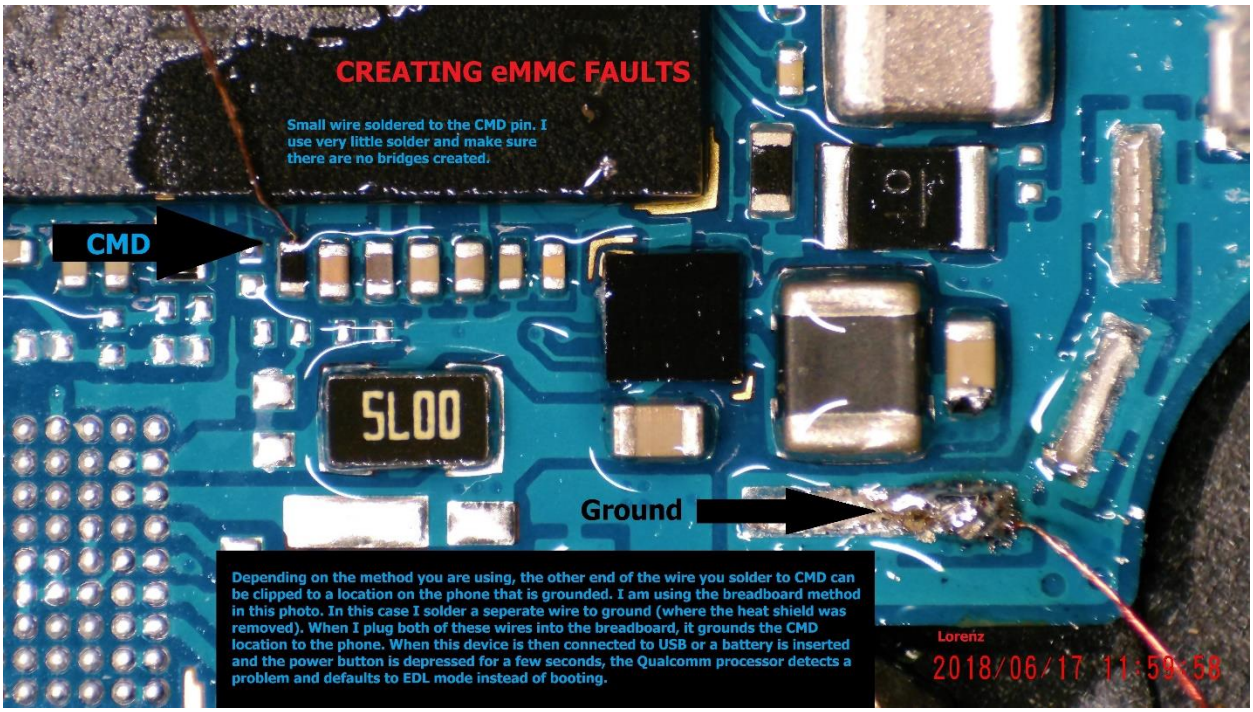
I think there is a lot of confusion and misunderstanding regarding eMMC faults and the procedures surrounding shorting. Most of the questions I get involve this issue or some problem resulting from attempting to short devices. Shorting the ISP points on a device is fairly simple compared to actually pulling the device via ISP, which require multiple connections. By shorting the Data, CMD, or CLK locations you are simply connecting those points to ground and then plugging in a USB cable or holding the power button. The Qualcomm processor then detects an

issue caused by the ground and immediately diverts to EDL Mode. As long as the device stays powered at that point, it will remain in EDL Mode. But before Cellebrite can exploit EDL Mode that you created by shorting the device, **you must remove the short that created EDL Mode**. I think this is where some confusion occurs. By removing the short you are not taking the device out of EDL Mode, you are removing the condition that put the device in EDL Mode. Behind the scenes, the UFED will need to control the device and it cannot do that if the device remains shorted.

When creating eMMC faults by shorting, you are connecting a wire to one of the ISP locations already known or mapped out on that device. The other end of that wire is then connected to any part of the phone that is grounded. Heat shield frames are what I use frequently for this. If I am using the clip method, I will clip the wire to the heat shield frame after soldering the other end of the wire to an ISP point. Of course you do all of that with the phone in the off-state with the battery removed and disconnected. With the short created, applying power to the phone will send the device into EDL Mode.

Is soldering necessary?

When creating eMMC faults, I always solder. It is easier to control and I am comfortable with my soldering skills. If you have never soldered anything before, I recommend practicing on something that doesn't matter first. Most ISP locations are very small and a microscope will be needed. It is possible to operate without a microscope in some situations where large pads have been identified and mapped, but errors can occur during soldering which can affect the ability of the UFED to extract the device. Bridges can be created accidentally which keep the device permanently shorted and thus will fail to extract until that bridge is removed.



There are three general methods of shorting that I will cover here. Let me be clear that these are not the only methods and may not be the best. I have my preferences based on the equipment and experience I have. Other examiners may have a different method that works well for them. I am including a link to the diagrams for these methods. One of the diagrams (breadboard method) does a visual step-by-step walkthrough of an extraction demonstrating when and how the short is applied relative to the extraction process and requests in the UFED. All of these diagrams will be in the resources folder on the Mobile Device Forensics and Analysis forum.

Shorting Methods - Diagrams:

- Needle Short Method
- Clip Method
- Breadboard Method
- eMMC Shorting with Tweezers

Shorting and pinning devices with Universal Flash Storage (UFS)

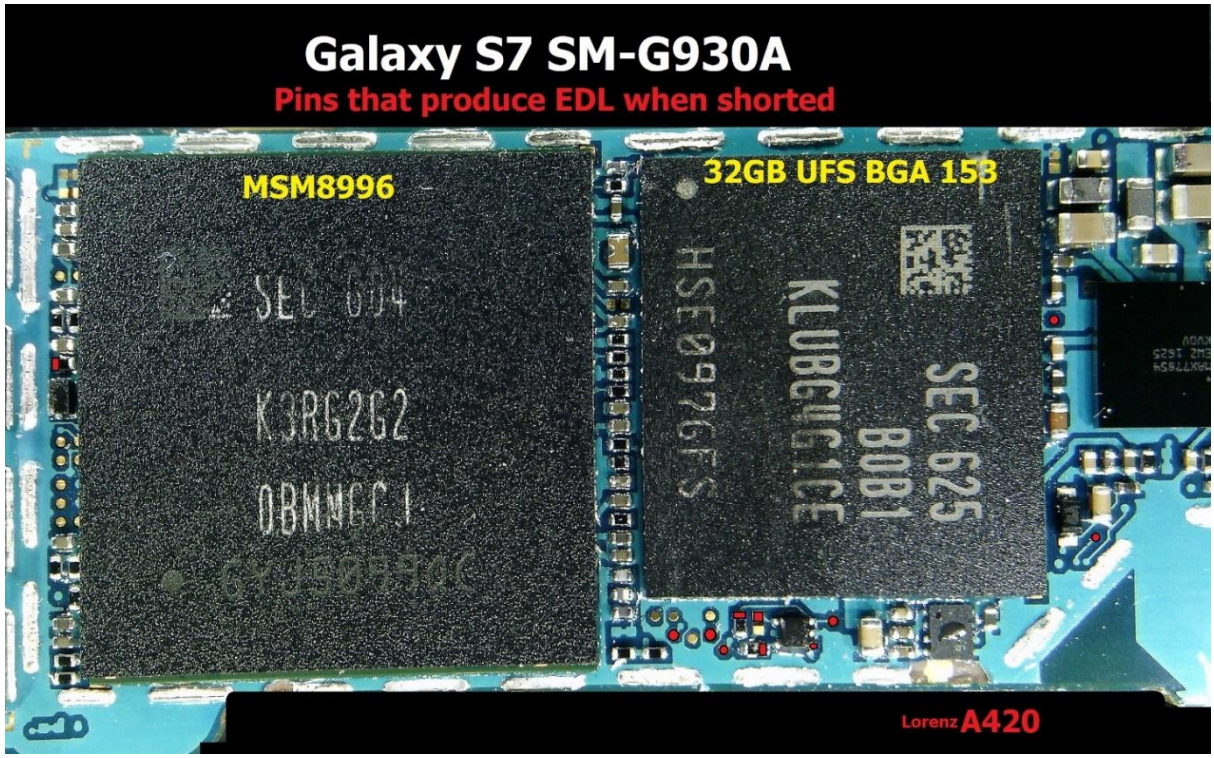
Shorting devices with Universal Flash Storage (UFS) is procedurally the same as shorting devices running eMMC storage. What makes this endeavor more challenging is that there are really no pinouts created for phones running UFS storage. So without reliable pinouts for locating ISP points as with eMMC chips, it means fishing for those points is necessary.

UFS is an advancement from eMMC chips. Because UFS chips were designed much differently that eMMC chips, ISP was not possible with them like it was with eMMC. Therefore, there were no pinouts created that tell us the location for ISP. I remember the first devices, of which I was aware and cared about, that took advantage of UFS storage was Samsung Galaxy - specifically the Galaxy Note 5 and the Galaxy S6. Those devices were running UFS BGA095 chips. Dediprog manufactured a chip reader and adaptors for UFS chips and I purchased one in late 2016 with a UFS BGA095 and BGA153 adaptor. The Galaxy Note 5 and S6 were not encrypted by default and thus we could chip those devices and obtain a physical as with eMMC chips. I used my UFS BGA095 adaptor many times on the Note 5 and S6 before there was a non-chipping exploit developed.

There was no ISP solution for UFS chips that developed and soon after UFS emerged, default encryption began to take hold. The few times I used my UFS BGA153 adaptor, was on test devices and they were all encrypted. Whether ISP was possible with UFS chips or not, there was no point with default encryption. Fast forward to today and we want to short ISP locations to take advantage of Cellebrite's EDL exploit, but we don't know where to short devices running UFS chips. But those locations exist and can be located.

Samsung Galaxy S7 UFS BGA153 with MSM8996

Except for the older phones running UFS BGA095, nearly all phones you encounter today running UFS chips will be encrypted by default. Most of the UFS chips I run across are all BGA153. Many Samsung Galaxy S7 devices run the Qualcomm MSM8996 processor and the UFS BGA153 storage. Just like phones using eMMC storage, there are numerous locations that, when shorted, create EDL Mode. Cellebrite can use the EDL exploit to obtain decrypted physicals from some S7's running the MSM8996, via ADB. That means the device must be unlocked and/or USB Debugging enabled. At the writing of this paper, Galaxy S7s shorted into EDL were failing to extract using the UFED decrypting method. I am including a diagram of the Galaxy S7 that shows all of the short locations I located that produce EDL.



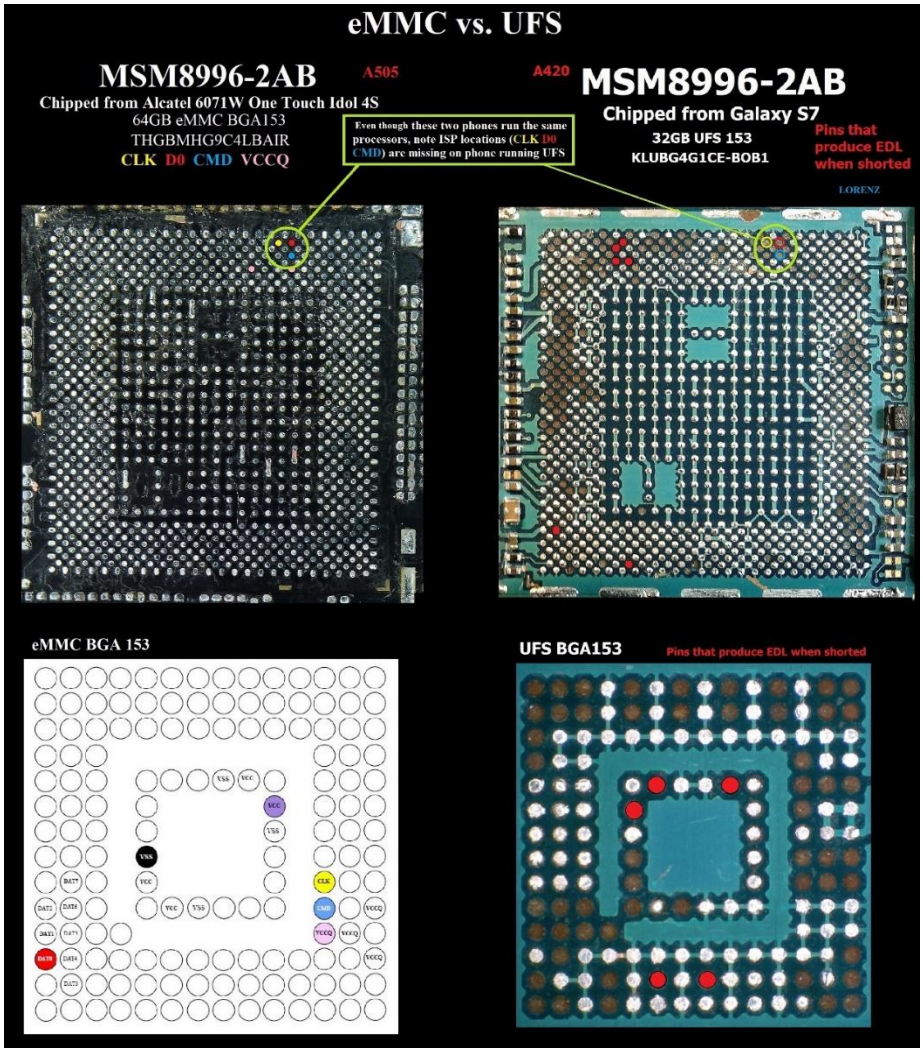
All of those pins on the S7 diagram reliably create EDL when shorted. This will be true on many phones running UFS storage. All of my testing of devices to locate shorting locations is done without the battery in the phone. I use soldering and the breadboard method to do my testing. This way I can be sure of the connection and with the battery removed I can be sure the device has no power until I connect to USB. Most of my testing for EDL short locations is done with only the logic board – no screen. That is because I have to remove the logic board to get to many of these locations and I don't need the device's screen to tell me I have achieved EDL. Device Manager will tell me when the device is in EDL.

Reverse pinning processors and UFS to locate short points

I have pinned many processors on test devices for ISP and JTAG in the past for purposes of helping me get into other devices running that same processor. Knowing the location of JTAG and ISP points on a processor is very useful. Is that still true with devices running UFS storage? Yes, in some situations it is. There are some devices I have encountered in which I had a hard time finding short locations just by

fishing and probing. Knowing points under a processor or a UFS chip that have been pinned and marked as connected to short locations on other devices can help when trying to locate short locations on a device running that same storage or processor. As I mentioned earlier, there has been much less work on pinning newer phones running eMMC storage due to encryption because an ISP extraction of an encrypted device would be useless for collecting user data. The same is true for devices running UFS storage, which will almost certainly be encrypted and because of their architecture, ISP will not work on UFS – that is, it was never researched very much because of encryption.

I am including a diagram comparing the eMMC BGA153 to the UFS BGA153. They look identical when chipped but they are completely different. This diagram also provides a visual of how the same processor can be differently configured if that processor is used on a device running eMMC and a device running UFS. So I can still pin UFS storage, but I am not identifying the pins as CLK, CMD, or Data. I am only identifying the pins as creating EDL Mode when shorted. That is all that is really useful at the moment with UFS. The same is true of processors mounted on phones running UFS storage. The MSM8996, is what I use as an example to show that pin configuration is based on the storage when it comes to locating short locations for EDL.



Troubleshooting EDL extractions and advanced methods made possible by the EDL exploit

There are many things that can go wrong and cause an EDL extraction to fail. Most of them are minor and harmless. There are some things that can go wrong and destroy the device and thus the evidence will be lost. I think the solution to both major and minor issues are both the same – attention to detail and preparation.

Some issues related to extraction failures are not the fault of the examiner. Examiners will receive devices that range from virtually new to completely destroyed. Sometimes working devices may have an unseen issue that prevents extraction. The failure can be misdiagnosed as not supported, when in fact it is something that can be corrected or bypassed all together.

Attention to detail

I am confident this is an issue because of the questions I get and because I have done it myself. If you get in a hurry and try to rush through, sometimes minor details can be overlooked and the extraction will fail. When the extraction fails, we often make assumptions that may not be correct.

Check for EDL and check that EDL can be removed

The same is true with shorting devices into EDL Mode. Remember that with eMMC faults, you must be able to remove the short that creates EDL in the first place. I get many questions from examiners who do everything correctly but the device will just not extract. Because I have been there myself, I can usually give them some ideas. A common error occurs when an unseen bridge is created via some overzealous soldering or the wire used to create the short becomes damaged and does not create a short or gets pinched when reassembling a device for an extraction and creates a permanent short that remains even after the clip is removed.

When an extraction fails in Step 1, in the UFED when using eMMC shorting, close the UFED 4PC if that is what you are using. Connect the shorted device and see if EDL is appearing in Device Manager. If it doesn't appear, you are not creating EDL in the first place and the extraction will fail. If EDL appears in Device Manager, remove the short and the battery and then connect the device to your PC again. This time you should not see EDL Mode. If you do, you have a bridge or the wire with the clip is pinched or exposed. Many times examiners will check for EDL, but will not check to make sure EDL can be removed. A permanently shorted phone will not extract.

Preparation

Do as much research on a device as you can before attempting an EDL extraction. Especially if it is one that requires dissassembly or shorting. Determine what processor it is running and make sure shorting is the only method to create EDL on the device. Is the device encrypted? An EDL extraction on a device that is not encrypted is much easier than using the decrypting method on some devices, as was discussed in this paper. Contact others on the forum who can provide some insight on everything including, encryption, short locations, methods, tools, and dissassembly.

There are things that can go wrong during dissassembly if not done carefully. I think this area is sometimes overlooked. I specifically created a video on removing heat shields a few months ago after some requests for that. Sometimes getting access to the locations to be shorted can be the most challenging part of the extration. Research and watch teardown videos on the internet. Beware that sometimes people can leave out steps so there's nothing wrong with watching a couple of videos on the same device. Most teardowns videos will not include removing heat shields and things that only forensic examiners would want to do.

Get test devices

It is sometimes difficult to convince supervisors, adminstrators, and prosecutors that you need to buy a phone so you can destroy it and/or practice what you are about to do. I have many people who will just give me phones because they know what I do with them. I get a lot of donations to my cause from people who know and trust me and also receive old evidence to be destroyed.

Andrew Rathbun provided a good idea that may provide a way to feed test phones to the forensic examiners at police departments. In his particular example, abandoned phones are brought as lost and found and must be stored for 90 days before they can be turned over to either the local domestic violence shelter or to the digital forensic unit to be used as test devices. Most departments have general orders or standard operating procedures for disposal of abandoned property that could be modified to benefit the digital forensics unit.

Extractions of badly damaged, encrypted devices using the EDL exploit

I receive a significant number of devices that have been damaged or have been intentionally destroyed. One of the reasons I wanted to learn JTAG, ISP and Chip-Off was to gain access to damaged devices. In the past, those methods have allowed examiners to extract many devices which could never be started again. Those methods are still very useful and relevant today as I have tried to emphasize in this paper. With that said, we can't deny that encryption has impeeded all aspects of digital foresics significantly, especially concerning devices that have been significantly damaged or destroyed.

At this moment, chipping a device or using ISP to extract data from an encrypted device is virtually useless. The physical process still works, but the user data is encrypted. There are companies working on a solution - that is the possibility of decrypting data after it has been extracted from the device that encrypted it. For now, I think we can safely use a broad generalization and say, it can't be done. I would like to have to eat those words in the future.

Using Cellebrite's EDL exploit on encrypted devices that don't function

One of the things I realized when I began experimenting with EDL was not just the decrypting aspect of Cellebrite's EDL exploit. That is exciting and impressive in itself but there are other types of extractions and lock bypasses that will yield decrypted data. One of the benefits of EDL extractions is that the device can be extracted via EDL without the need to interact with the device screen at all. Introducing eMMC faults and placing the device in EDL Mode can be done with no requirement to see or interact with a screen. From that point, the UFED can take control of the device and extract decrypted data.

With damaged devices, the USB port becomes one of the most important parts of the device. If the phone is broken in half and the USB port is beyond repair, can an encrypted device be extracted? We need the device to give us the data through the USB lines, but if the port is gone what can we do? Removing the storage is out due to encryption. The solution to the problem is really as simple as using skills learned from JTAG and ISP training. Not just the soldering, which is important. I am talking about pinning devices for USB data and power connections.

Damaged devices

Damaged and broken devices means you have to locate alternate locations on the device to tap into the USB data lines. When utilizing the decrypting EDL method in the UFED, the device also needs to boot. That means it may be necessary to duplicate power and communication normally done through USB and the battery. All of those things are possible and can be done with a multimeter, some soldering, pinning, and a lot of research and patience.

There are a lot of little details related to finding these locations and getting it all to work. I will include some links to photos and diagrams showing examples of sucessful EDL extractions of badly damaged devices and leave it at that for now. Specifics and pinout methods will be in another document. The important point here is that EDL extractions can allow us to extract physical images from badly damaged, encrypted devices. In that respect, EDL gives us options for encrypted devices which are similar to the use ISP, JTAG and Chip-Off with non-encrypted devices.

Phone	Processor	Encrypted	USB Bypass	Decrypted EDL Extraction
ZTE Z799VL	MSM8909	Yes	Yes	Yes
ZTE N9136	MSM8909	Yes	Yes	Yes
Samsung J327VL	MSM8917	Yes	Yes	Logical Pull - not yet supported for EDL 7-13-18

Links to Documents and Information on EDL in this Paper

Item (Link)	Description
Practical Guide for Qualcomm EDL Physical Extractions	Shahar Tal's Introduction to EDL Extractions
Cellebrite EDL Webinar 2-21-18	PDF of Cellebrite's EDL Webinar on 2-21-18
EDL Non-decrypting Pulling Steps	Step-by-step guide for using the non-decrypting EDL method and testing any device for EDL - by Lorenz
EDL Decrypting Pulling Steps	Step-by-step guide for using the generic decrypting method for EDL extraction - by Lorenz
UFED Generic EDL Instructions	General Instructions given by the UFED on placing devices into EDL Mode on "waiting to extract" screen.
EDL Mode in Device Manager	What EDL looks like in Device Manager
Three Choices for Extraction	A description of each button's purpose on non-decrypting and decrypting EDL extractions - by Lorenz
Testing and Extracting with Cellebrite Cable # 523	Methods and steps in using Cellebrite's EDL cable #523 - by Lorenz
Non-decrypting EDL Pulls Without Battery	A visual description of a phones condition when extracted via the non-decrypting EDL method - by Lorenz
EDL Mode Via ADB with UFED 4PC and Command Prompt	A step-by-step guide to using automated and manual methods of creating EDL via ADB - by Lorenz
See Moto Heat Shield Removal Video	Video showing how to remove tough heat shields without damaging the device - by Lorenz
Moto Heat Shield Removal Close Up	Video showing how to remove tough heat shields without damaging the device - microscope view by Lorenz
eMMC Chip Pinouts	Diagrams showing pinouts of 3 eMMC chips - by Lorenz
ZTE Z799VL - Which Points Create EDL	Diagram demonstrating all possible eMMC short locations - by Lorenz
Needle Short Method	eMMC Shorting a device into EDL via a needle and a wire with no soldering - by Lorenz
Clip Method	eMMC shorting devices into EDL via a wire and clip
Breadboard Method	eMMC shorting devices into EDL via breadboard method
eMMC VS UFS Chips	Diagram showing the different between eMMC and UFS chips
Galaxy S7 SM-G930A EDL Short Points	Diagram showing multiple EDL short points on the Galaxy S7
Soldering Short Points	Diagram showing clean soldering of eMMC points
XT-1650-2 Hidden Short Locations	Diagram showing hidden short locations for creating EDL
LG Test Points that Look Like a Cross	A sample LG pinout demonstrating common LG Test Points that look like a cross - by Lorenz
Kim Thomson – ISP Recognition	A paper discussing techniques to locate ISP pins without chipping a test phone – by Kim Thomson
MSM8909 PIN PATTERNS	ISP pin patterns based on devices running the MSM8909 processor – by Lorenz
MSM8916 PIN PATTERNS	ISP pin patterns based on devices running the MSM8916 processor – by Lorenz
Soldering to ISP Points using Clip Method	PowerPoint slides showing solder to ISP points to create EDL using the clip method - by Lorenz
EDL Procedure for H1611	A procedure for shorting the Huawei H1611 - by Lorenz
EDL Procedure for Z799VL / Z798BL / Z899VL	Diagram showing step-by-step instructions for extracting the Majesty Pro Series of phones using EDL

Links to How-to Videos

DFU and FTM Extractions	Using button combos to place devices in EDL and FTM mode and extracting them
EDL Cable Testing and Extraction	Testing for EDL cable support and extracting with the EDL cable
eMMC Faults – Shorting	Shorting devices into EDL mode using soldering and needlepoint method by creating eMMC Faults
LG M150 Test Point Decrypting EDL	Using Test Points to create EDL on the LG M150 and extracting using decrypting bootloader
Motorola XT1650-02 EDL short Points	Locating the hidden points that create EDL mode when shorted on the Motorola XT1650-02

2nd EDL Webinar Videos – 9-12-18

01_Z971 Cable 523 Extraction	2nd EDL Webinar – use of Cable 523 for EDL
02_Z839 DFU Mode Extraction	2nd EDL Webinar – DFU mode procedure
03_N9136 FTM Extraction	2nd EDL Webinar – FTM mode procedures
04_MS330 Test Point Extraction	2nd EDL Webinar – LG Test points
05_Z719DL - Needle eMMC Fault Extraction	2nd EDL Webinar – needle used to short
06_Z852 eMMC Fault Extraction	2nd EDL Webinar - soldering
07_Coolpad 3636A Decrypting EDL - boot to charge-only	2nd EDL Webinar – Issue for applying decrypting bootloader to devices that boot to charge-only mode
08_Oppo R9S - Cable 523 Decrypting Extraction	2nd EDL Webinar - Issue for applying decrypting bootloader to devices that boot to charge-only mode