Fully revised to cover IoT, wearables, UAS devices, and more

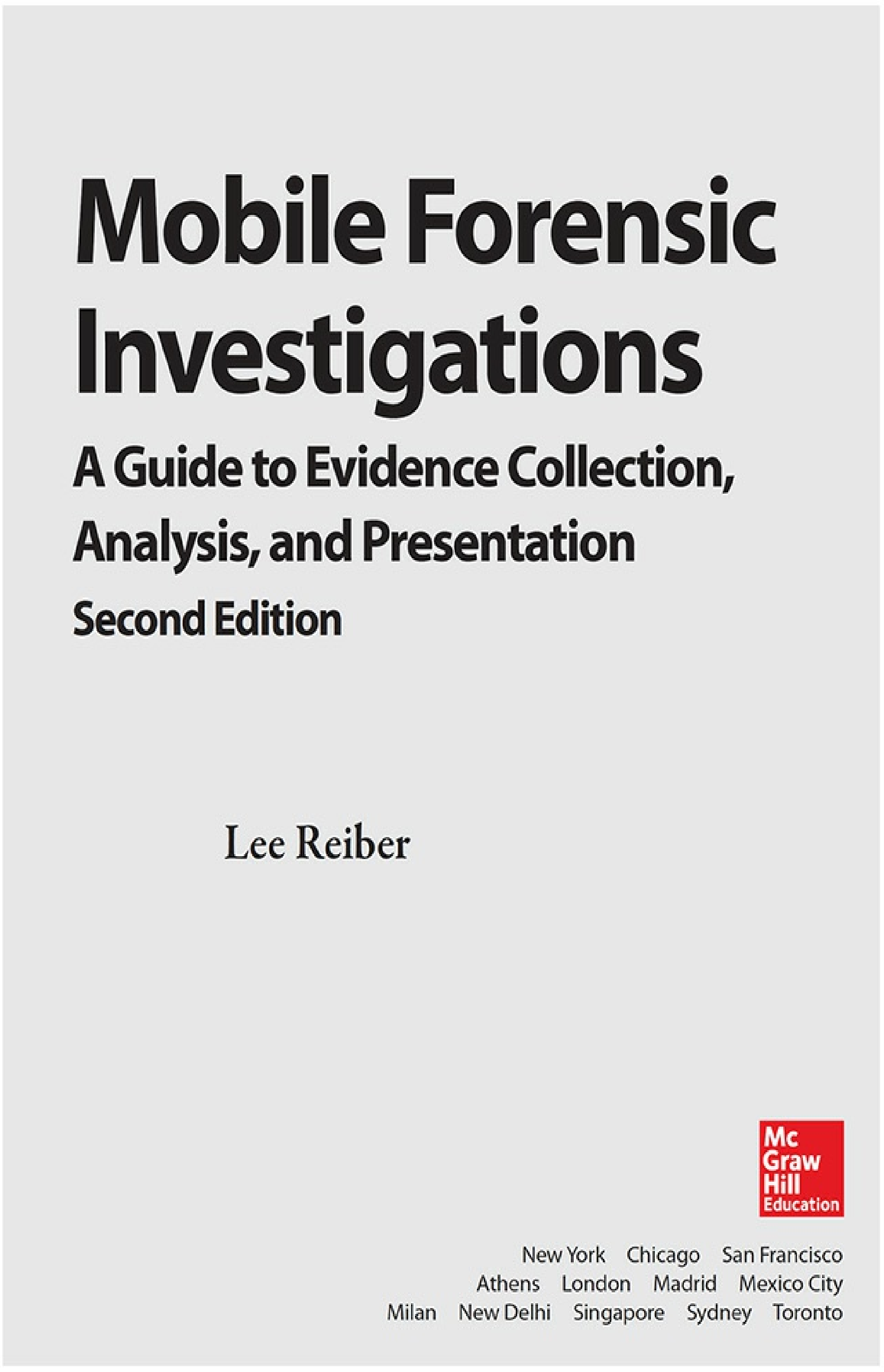
MOBILE FORENSIC INVESTIGATIONS

A Guide to Evidence Collection, Analysis, and Presentation

SECOND EDITION

Mc Graw Hill Education

Lee Reiber



Mobile Forensic Investigations

A Guide to Evidence Collection, Analysis, and Presentation

Second Edition

Lee Reiber

Mc Graw Hill Education

New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

ISBN: 978-1-26-013510-7
MHID: 1-26-013510-1

The material in this eBook also appears in the print version of this title: ISBN: 978-1-26-013509-1, MHID: 1-26-013509-8.

eBook conversion by codeMantra
Version 1.0

This book is dedicated to my wife, Amy, my biggest critic, who has put up with 20 years of geek speak and continues to push me to be the best human being possible; to my family who continually supports me; and to the digital forensic community who continually inspires me.

About the Author

Lee Reiber started his journey as a member of the Boise Police Department in Boise, Idaho, where he conducted digital forensic investigations until 2009 after almost 15 years of service. During the last few years at the police department, Lee's training company, Mobile Forensics, Inc., became one of the most prominent training companies in the United States for mobile forensics, training hundreds of students from law enforcement, Fortune 500 companies, and academia. MFI specialized in instructing its students on how to interpret and analyze mobile device data collected with multiple tools. It was Lee's research that produced discoveries in data formats, date and time configurations, and file system artifacts that are still used in training today. Due to the reputation of MFI, based upon the company's success in research, development, and training in mobile device forensics, MFI became a part of a global software company in 2009. Lee was promoted to Vice President of Mobile Forensics, and he created and developed the mobile forensic curriculum and mobile forensic software solution, both of which are still a part of the company's offerings. In 2015, Lee departed the company and became the COO of one of the most recognized mobile forensic software companies in the world, specializing in deep data analysis of mobile device, cloud, IoT, and UAS artifacts.

Lee has testified as an expert in mobile forensics in both criminal and civil cases during his 20-plus-year career and consulted for both international and domestic companies requesting mobile forensic assistance, mobile device collections, data analysis, and data interpretation. Due to his aptitude for deep analysis, Lee is frequently called upon to assist in high-profile cases involving data from mobile devices when alien data is encountered. Lee has written more than 50 articles on mobile forensics, has been featured in both national and international magazines and print media, and has lectured around the world on mobile forensics and cyber security. This book is a product of the 20 years spent staring at HEX and ignoring the push-button philosophy.

About the Technical Reviewer

Michael Robinson is a senior cyber threat analyst and digital forensic examiner for a large international company, where he specializes in mobile device forensics, computer forensics, and intrusion analysis. Prior to this role, Michael was a senior digital forensic examiner for customers in the U.S. intelligence community, where he performed computer and cell phone exploitation and analysis. Previously, Michael performed computer forensic examinations in the FBI's Investigative Analysis Unit, where he assisted special agents with counterintelligence and criminal cases. Michael is the former CIO of the U.S. Department of Defense's Business Transformation Agency, where he oversaw all information technology and information assurance operations for the agency, including overseeing all incident response and forensic investigations.

Michael was the Program Coordinator and an Adjunct Professor for Stevenson University's Master of Science in Cyber Forensics. At Stevenson, he was the recipient of the Rose Dawson Award for outstanding adjunct faculty member of the year. He was also an adjunct professor in George Mason University's Master of Science in Computer Forensics. He taught courses in mobile device forensics, intrusion analysis, and cyber warfare. He holds a Bachelor of Science in Chemical Engineering, a Master of Science in Information Assurance, a Master of Science in Forensic Studies (concentrating on computer forensics), and a graduate certificate in Applied Intelligence. Michael has presented at numerous national and international conferences, including DEF CON, the DoD Cyber Crime Conference, U.S. Cyber Crime Conference, CEIC, InfoSec World, and the BCISS Conference on Intelligence Analysis. He has authored more than a dozen journal articles and a book on disaster recovery planning for nonprofit organizations.

Contents at a Glance

Contents

Introduction

Mobile Forensic Investigations: A Guide to Evidence Collection, Analysis, and Presentation, Second Edition, is a comprehensive, how-to guide that leads investigators through the process of collecting mobile devices, cloud sources, IoT devices, wearables, and UASs, then moves to analyzing the collected data and disseminating their findings. This book was created from the many questions received during training courses, lectures, research, and interviews over many years and a desire to impart the answers to new, seasoned, and advanced digital forensic examiners.

Until now, no direction or guidance has been available to students and practitioners other than a few manuscripts and many vendor-specific training courses. Unfortunately, examiners have been left to figure out mobile forensic procedures and techniques for themselves, and often, at least in the digital forensic circles, mobile forensics can still be referred to as the "Wild West" of digital forensics—just point and click. Now with the world of IoT, wearables, vehicle systems, and drones, a new digital field has emerged, but the same problems persist. By trusting only in automated tools, most examiners today do not fully understand the methods and processes used, so this term often fits. It is the goal of this book to change this mentality and move the examination of a mobile device into today's required standards.

This book is intended not only to educate new students coming into the field or those looking for a career in mobile forensics but also to inform examiners who have been conducting mobile forensics for years. It helps both student and examiner understand what constitutes processes and procedures, how to formulate an examination, how to identify the evidence, and how to collect the various devices, and it culminates with advanced tools and methods the examiner can use to uncover data that most tools forget.

This book can be read from cover to cover, but it can also be used to consult individual chapters during an examination. With the many tables and figures outlining mobile device file systems, targeted files, and digital gold, the student and examiner can use *Mobile Forensic Investigations: A Guide to Evidence Collection, Analysis, and Presentation, Second Edition,* for reference during many examinations.

The first two chapters help expose the reader to the world of mobile forensics and clearly define the differences and similarities between mobile forensics and computer forensics. Chapters 3 and 4 bring in new concepts with IoT devices, infotainment, wearables, UAS, and cloud stores—the new frontier. Chapters 5 and 6 outline the steps an examiner should take when coming into contact with mobile device evidence, including how to handle the evidence, and include information on the types of mobile forensic tools and the multitool approach. Chapters 7 through 10 begin the exploration into the first examination by setting up the collection environment and defining what problems can be encountered, along with ways to fix them for both collections and data analysis. Chapters 11 through 16 are all about the data. This includes determining what type of data should be expected within the various mobile device file systems, IoT devices, wearables, UAS, and cloud stores. More importantly these chapters speak to what type of data should be expected in a standard collection versus an advanced collection and how to decipher and decode advanced data from iOS, Android, Windows Mobile, and BlackBerry devices. Chapter 17 discusses how to present the data and how to become a mobile forensic device expert. This chapter explains that without proper documentation detailing the process from collection to analysis, the recovered evidence is often confusing and could be inadmissible.

A student or an examiner in mobile forensics must be prepared for tomorrow today. *Mobile Forensic Investigations: A Guide to Evidence Collection, Analysis, and Presentation, Second Edition,* provides a tremendous start.

1 Introduction to the World of Mobile Device Forensics

In 2014, Cisco's Visual Networking Index (VNI) Global Mobile Data Traffic Forecast Update indicated that the number of mobile devices in use exceeded the world's population. In February 2017, the same study by Cisco revealed that "mobile devices and connections in 2016 grew to 8 billion." In the mobile world, wearables are the latest and most easily noticeable digital devices. Cisco reported that there were 325 million wearable devices in use globally, with a subset (11 million) having a cellular connection as well. It should come as no surprise that mobile device sales have outpaced PC sales three-to-one since 2003, as reported in a study conducted by the National Institute of Standards and Technology (NIST). And between 2010 and 2016, the ratio increased to almost five-to-one, as reported by MarketWatch in 2016.

Statistically, the examination of mobile device data should be proportionally similar to computer examinations, but unfortunately, this is not the case. In actuality, computer evidence is still more prevalently used in civil and criminal cases, but use of mobile device evidence is on the rise, though not at the rate of induction of the actual devices. In fact, in each and every venue in which law enforcement has been surveyed, participants have unequivocally stated that they receive more requests for the analysis of mobile device data than for any other type of electronic evidence.

A common theme with mobile forensic experts in both law enforcement and enterprises is the overwhelming inundation of electronic evidence from mobile devices, which is increasing at an alarming rate —so much so that the groups I've surveyed from both law enforcement and corporate organizations indicate that they've hired or assigned specialists who examine and collect only data from mobile devices. What is truly alarming is the fact that most of these examiners also indicated that little consideration is given to the actual content of the mobile device when a computer is also part of the electronic evidence scene. When an American adult spends an average of 34 hours on the Internet using a mobile device, versus only 27 hours using a PC (as reported in 2016 by Nielson, a global research company), shouldn't a forensics examination reflect that? This mentality is based primarily upon the limited amount of information available on correctly processing, analyzing, and reporting on the data from a mobile device, whereas computer forensics has been time-tested and is accepted by examiners and courts across the globe.

The proliferation of mobile devices will only increase with the world's population growth and as the population's dependency on technology accelerates (see Figure 1-1). Now should be the time to accept and understand that the information contained on a mobile device details, documents, and exposes the thoughts, actions, and deeds of a user substantially more than any data stored on a personal computer.

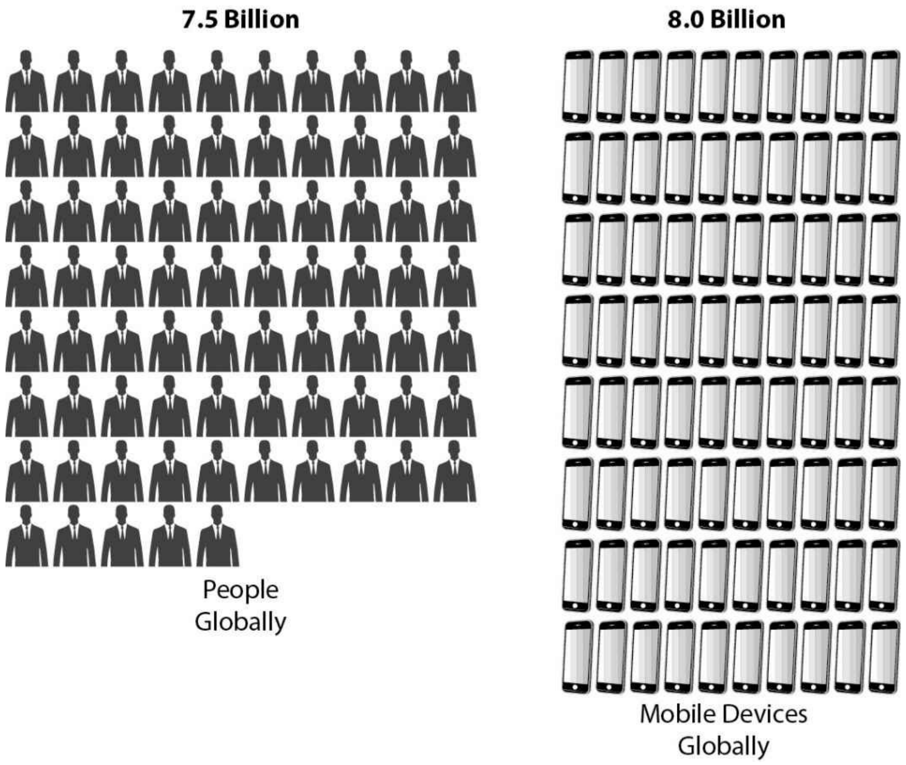


FIGURE 1-1 **There are currently more mobile devices on Earth than people.**

With this unprecedented amount of electronic evidence comes the need for highly skilled mobile device forensics investigators. This book is a comprehensive, how-to guide that leads investigators through the process of collecting mobile devices, analyzing the data found, and disseminating their findings. This holistic approach to mobile device forensics will not only feature the technical approach to data collection and analysis, but it will also explore the legal aspects of mobile device forensics. It is widely known that today's digital forensics examinations have been "dumbed" down, with a heavy reliance placed on using tools to extract and collect data. These methods have bred a systemic influx of "data collectors," rather than mobile device forensic examiners. This book attempts to counter this trend as it approaches and readies not only the newest examiners, but seasoned digital experts as well, with skills and knowledge that surpass the simple "easy button" approach.

The successful examination of digital evidence from a mobile device requires an intimate understanding of the data, and not the medium that produced it. This fact is especially important when it comes to the examination of the artifacts left behind by a mobile device. Often the unique aspects of mobile devices are overlooked by examiners and students when encountering these small-scale digital devices, because their first impressions are often clouded by the actual size of the device. It is a common belief that the small size of the device implies that it contains only primitive and rudimentary data, and this equates to the belief that the examination of the device should be consistent with its physical size. The size of today's device has no bearing on the storage, however, with most of today's smart devices having a storage capacity far greater than that of PCs less than ten years ago. In practice, many believe data forensics should be as easy as inserting a flash drive and hitting the "find evidence" button. However, because of this tactic, most examiners, and quite honestly the majority of examinations, fail at the critical part of the evidence life cycle—the dissemination of the facts.

In short, the size of today's modern smart device is in no way an indication of its storage capacity. Moreover, it is important to remember that although the physical device that holds the data may be different, the data and artifacts produced by the underlying operating system are what the examiner must focus on.

An examiner who must testify to his or her findings will quickly realize that it is not acceptable to testify to the fact that he or she pushed the "get evidence" button and then believe that the resulting reasoning/explanation will suffice in court. An examiner must be ready to answer the question, "Where did the phone book come from?" and not answer, "From the phone." The true expert digital forensics witness will be ready to give the location of the phone book's contents as they relate to the phone's file system and to specify how the user data was not altered during the examination. Unfortunately, for most, this is a daunting task because it takes additional training, for which many admit they have neither the time nor the resources. By reading this book, understanding the principles discussed, and, most importantly, working through the various scenarios, your experience as an examiner will have already surpassed the majority of the people conducting mobile device examinations today.

The primary goal of this book is to suggest methods and procedures for the collection and analysis of digital data from a mobile device so that your examination will stand up to the scrutiny of court. This book also aims to dispel the mobile device size myth and to prove that today's mobile devices contain more relevant and transactional data than a personal computer 100 times its size. Armed with a full understanding of the wealth of information available for discovery contained in this book, you will be prepared with not only the knowledge of how to use mobile device forensics tools to conduct investigations but also with a holistic understanding of how and what these tools are doing.

A Brief History of the Mobile Device

Welcome to the world of mobile device forensics. The examination of data is no different from the examination of the words in any book or manuscript. To understand the words in a book, the reader must have an understanding of the letters that form the words and the words themselves to form a thought or sentence. Without this initial understanding, reading would not be possible. The same goes with understanding a mobile device. To be a truly good forensic practitioner, the examiner must understand the specimen. Before we begin a discussion of mobile forensics, let's begin with a history of the evolution of mobile devices themselves.

Mobile phones for cars and other vehicles preceded the handheld mobile phone and by all accounts were initiated by Bell Labs in 1946 in St. Louis, Missouri. This book will focus on the handheld mobile device that followed a little more than 26 years later.

Martin Cooper

In 1973, Martin Cooper from Motorola began developing a device that would enable a person to walk the streets while making calls without the need for attached wires—a portable handheld mobile phone. On April 3, 1973, Cooper demonstrated the portable phone with a call to Joel Engle of Bell Labs, Motorola's main competitor, to let Engle know he was calling on a portable handheld phone from the streets of New York City. The call was routed via a base station Motorola installed atop the Burlingame House into the AT&T

landline telephone system. Cooper had been working on the development of a radio phone for many years. Ten years after he used this device to call his rival, it was brought to the mainstream market. This radio phone employed the same technology used by the military to communicate on the battlefield.

Cooper developed the DynaTAC (DYNamic Adaptive Total Area Coverage) 8000x portable phone, which enabled users to call another portable phone, landline, or radio phone. This device was approved by the U.S. Federal Communications Commission (FCC) on September 21, 1983, and was offered commercially in 1984. It weighed 2.5 pounds, with the battery contributing the majority of the size and weight. The talk time was approximately 20 minutes, and it took approximately 10 hours to charge the device between uses. This first device was, by today's standards, impractical due to its size and weight. Picture the 1980s TV show *Miami Vice* and the main character played by Don Johnson, with white cotton pants and matching jacket, with a large, angular phone and protruding black antenna—that was the DynaTAC 8000x.

The price tag of the device made it extremely impractical for widespread use against the prevalent pager or beeper of the age. When mobile radio devices came to market, prices ranged from U.S. $2000 to $4000, and they seldom worked outside of only a few major markets. The DynaTAC 8000x was priced at $3995.

Despite the introduction of the DynaTAC, car phones continued to be more popular, mainly because of the large mobile phone price tag. By 1984, the car phone was a large device with a very heavy base and typically a curly cord that ran to the handset. These devices had much better transmission and reception rates because of the constant battery power that exceeded the current ranges of the DynaTAC. Car phones were typically mounted and hardwired in vehicles and looked like a normal kitchen phone of the day. Users wanted phones that were more portable, however, so manufacturers began making the devices capable of being removed from the vehicle. The *bag phone* was born. ("Bag phone" was not the official name, but it stuck simply because the device could be carried in a zippered bag.) Impractical and extremely heavy, this device was not around for long. Who is to say what the impact of the electrical radiation might be?

Size Evolution

Shortly after the DynaTAC, cellular devices began to be manufactured in smaller form factors. In 1989, the Motorola MicroTAC was released. This device was much smaller than the DynaTAC series but still large and expensive enough that mainstream consumers were not buying. It was not until 1996 when the StarTAC series entered the market that things changed. The first cell phone to make a sizable impact with consumers was the full clamshell feature phone, which could be carried around in a pocket or small purse. The price point for the Motorola StarTAC was $1000. Because these portable devices only made and received calls, the price tag versus necessity was still out of reach for most consumers, however, so the progression away from the landline to the cell phone was not immediate.

The price for the mobile device started to drop when the technology, components, competition, and assembly became cheaper. In the mid- to late 1990s, there was a shift to make the smallest device possible, but in the early 2000s, the size paradigm started to shift the other way. Today, consumers are speaking into mobile devices the size of small laptops, conveniently termed "phablet," for phone tablet. What has become important is the type and amount of data that can be transmitted, as well as what components are inside.

Data Evolution

Cell phone users quickly saw the need to send messages that they could type into a QWERTY keyboard or pound down the keys in a series of taps instead of speaking to someone. From this need, Short Message Service (SMS) was born. Limited to 140 characters, SMS enabled users to express themselves via a short sentence to other mobile device users. Today, 140 characters is still the limit. Concatenated SMS, or what some call Protocol Data Unit (PDU) mode SMS, is more widely used and provides 160 characters by changing from using 8 bits per character to 7. This was a significant advancement that moved the pager away from the businessman's belt loop and to the eventual demise of this type of communication service. The cell phone soon advanced away from the simple walkie-talkie. Mobile device users next needed storage for these text-based messages, and soon for pictures, videos, calendars, and e-mails as well—who would have foreseen the application revolution?

Note

The walkie-talkie, invented by Donald L. Hings, was first used as a portable field radio for pilots flying around Canada. In 1937, when the first walkie-talkie was built, Hings referred to the device as a "two-way radio," and it has been used ever since.

Storage Evolution

Most of the original mobile phones did not contain a media type that would support the storage of data, because most were engineered to use the space as a scratch pad to run the needed process, remove the process, and move to the next process. Storage of *nonvolatile* data, or data that would continue to exist without constant power, onto a device's memory chip was not possible initially. This meant that any data that was visible on the device would not be stored, and as soon as the device was shut down, all the data disappeared. Because initially the mobile phone was used only to make and receive calls, this was not observed as a limitation.

Because of this, no phone book storage or any other settings could be maintained. The first cell phones were merely a block of circuitry that had the same combinations of buttons included on the well-known landline. The functionality was the same—attempt to recall the number of a contact from memory and key in the number (think about that—having to remember all of those phone numbers).When TDMA (Time Division Multiple Access) devices in the United States began to transition to GSM (Global System for Mobile Communications), which had already arrived in Europe, device information such as a phone book and SMS could be stored onto the SIM (subscriber identity module). Storage areas for contacts, SMS, and last numbers dialed had already been built into the SIM standards developed by the ETSI (European Telecommunications Standards Institute), where this data could be written to and stored to the smart chip.

Since SIM cards were already storing phone data that was used in the authentication process, why not add values that stored the contacts, SMS, and last numbers dialed? The SIM card was already being used in the device as a "key" to the cellular network, assisting in the authentication of the device using stored values. TDMA devices could store a set number of contacts that users could look up and select to be dialed, plus a limited number of SMS (15–25 typically; that's it!) that could be reread, even if the device was shut off. Other carriers that did not have SIM cards started to develop devices that used nonvolatile memory so data could be stored and saved even if the device was turned off completely or the battery was removed or drained. The older feature phones used an acronym—NVM, for nonvolatile memory—to designate this type of data within their file systems and prefix the file and/or folder name. This was where data was actually written to the phone and the start to the recovery of a goldmine of data.

Volatile Memory and the Visor PDA

A great example of volatile memory and mobile devices is the Visor personal digital assistant (PDA). With the technology of all PDAs at the time, most had a whopping 8MB of RAM and did not store data for long if no power was supplied.

On to the story.

A Visor seized in a search warrant was brought in for examination. The power cradle had been left at the scene, and it was a Friday, so the device was going to have to sit in the evidence locker until the following Monday. Upon arriving back at the lab on Monday, the examiner tried to start the device, but it was immediately apparent that the device had lost all power—it would not boot up. A power cradle was located for the device at a local store and the device was charged. When there was significant power to the device, the examination continued. To the dismay of the examiner, there was nothing on the device—no user content anywhere. What was puzzling was the fact that a substantial amount of data *had* existed on the device and was outlined and documented by the officers who seized it. Had someone broken into the lab over the weekend and deleted the data from the device? Had the officer somehow deleted the data or caused it to self-destruct?

The Visor documentation was consulted, as well as manufacturer's documentation, which indicated that PDAs drew constant power to keep the data on the devices populated. If the device was allowed to be completely drained of power, the data would no longer be available. Because the data resided in the device's RAM, the data was volatile, and, as such, when power was lost, the data stored in RAM was also lost. It was a hard lesson to learn but a great example of the limitations of volatile memory.

Moving forward, all devices had to have their power maintained, if possible, and most training courses on mobile forensics began to preach this lesson as well. Instruction on the use of portable charging units during transport as well as during storage began to be included in training courses and digital forensic examination kits.

Mobile Device Data: The Relevance Today

Mobile devices have come a long way since their inception. Data contained on a mobile device can now be compared to the written notes of a daily diary (often with pictures and videos), containing a company's most secure documents, our financial status, and, most importantly, our everyday habits, patterns, and routines. If a picture of an individual were painted with the personal data recovered from a PC, the picture would be a blurry representation with no clear edges. If, however, the data recovered from a mobile device were examined, it would most likely paint a very personal, and potentially embarrassing, picture of the individual.

Mobile Devices in the Media

Every day, the world media reports cases solved using evidence from mobile devices or stories about how a mobile device was involved in a crime—either a text message or chat was sent or received; a social media post was sent or interpreted; or a voicemail was heard, recorded, or hacked. In January 2018, the Pew Research Center reported that 95 percent of U.S. adults owned a cellular phone, which translates to about 309 million cell phone users. It is no wonder that law enforcement digital examiners are looking to these devices as an evidence treasure trove. Mobile device data is today's equivalent to yesterday's DNA evidence.

One day, a mobile device that shows up in the lab or in the field may hold data that has never been seen by the world—the grassy knoll, the shot heard 'round the world, the missing plane's last communication, and the smoking gun. Quite simply, you can fill in the blank to determine what a mobile device might be involved in when contemplating the events of today.

Digital data from a mobile device holds the key. To the forensic examiner, this data is critical in many investigations. The sidebar "Mobile Devices on the Scene" details the relevance of data on a mobile device and how its recovery is critical.

Mobile Devices on the Scene

In 2009, a fight was reported on a congested Bay Area Rapid Transit (BART) train platform. After officers responded to the scene, an officer shot one of the subjects, who later died at a nearby hospital. At the time, there were no news crews on the scene or closed-circuit television cameras in the area. What *was* on the scene were multiple people with cell phones, who captured the event both in still and video formats. These videos were uploaded and disseminated to media outlets and the Web. It is reported that the raw video of the event was downloaded more than 500,000 times in a single week! Videos from the cell phones taken from many different angles were critical to the case. The videos were irrefutable evidence of the transpiring events—a police officer was subduing an individual at the train platform, shots were fired, and the account of what happened was clear. The individual who was being subdued was shot as onlookers watched. What is different about this case is that it was not a witness's perception of the event, but the actual event caught on video in real time. Six videos were known to exist prior to the case and were subsequently used in the court cases. By all accounts, the videos from the mobile devices were critical evidence used by both prosecutors and the defense during testimony at not just the criminal trial but also the civil trial.

The stories of eyewitness accounts captured with a mobile device are endless. In a traditional investigation, investigators would have to rely on an actual person recalling the event—which, as most know, can be susceptible to many different perceptions, views, and beliefs. The always on, always ready, and always filming video camera in today's mobile device has revolutionized and forever changed what an eyewitness is and will ever be.

The Overuse of the Word "Forensic"

NIST describes "forensic science" as "the application of science to the law." Forensics can include the scientific examination of fossils, a crime scene, metals, vehicles, postmortem bodies, and of course, digital data in many different forms. Digital examiners get lost in this meaning and sometimes interpret forensic science to indicate that a software application or a hardware device is forensically sound. But according to the NIST definition, the use of the tools by the examiner in a way conforming to known scientific processes and procedures is the forensic science, not the tool itself. A perfect example is the use of a write blocker when processing digital data.

Write Blockers and Mobile Devices

NIST and its Computer Forensics Tool Testing Program specifically state that the central requirement of a proper forensic procedure is that the examined original evidence must not be changed or modified in any way. One of NIST's listed requirements for a layered defense is to use a hard disk write-blocker tool to intercept any inadvertent disk writes.

Note

Many different kinds of write-blocking equipment have been created by many different companies. Any mention of a specific unit here is not an endorsement of the tool; it simply indicates that the tool was used in the case or test.

A *write blocker* is a software or hardware device that stops specific communication from a computer to a mass storage device. Write blockers come in many different types. Software-based write blockers can use a simple Windows Registry change; hardware units are sophisticated boxes that are coupled to the examination computer via cables, with the device to be examined attached to the other side. Some allow a connection directly to the pins located on the actual hard drive and then to the computer conducting the forensic analysis, while others have USB connections to plug a removable USB hard drive or flash drive into an available port.

Hardware tools can be used to protect disk access through the interrupt 13 BIOS interface of a PC. Because the mass storage device is attached to the hardware write blocker, all input and output (I/O) commands sent from the PC are monitored. Any commands that could modify the hard drive are not passed onto the hard drive, but intercepted by the write-blocking device. Software write-blocking tools also block the writing to attached drives plugged into the USB drive, mounted drives, and by device classes, if needed. This can be changed by editing the Registry of the Windows PC or using many of the available software tools.

Essentially, the write blocker acts as a traffic signal to data requests made by the computer. The computer makes requests to receive data from the connected device, which are accepted and processed; if a request to write data to the protected device on the other side of the write protector is made, these requests are stopped and not allowed to reach the device. The write-blocking hardware or software tool is not considered forensically sound, but the employment of the device and methods as part of the process is. To test this theory, obtain any write-protecting hardware device and attach a mobile device; then follow along in the upcoming sidebar, "Real-World Example: Testing the Theory," in the next section.

Field Note: From Commercial Tool to Forensic Tool

This story was passed along to me when I first started training examiners in the mobile forensics world as I began researching emerging software.

While I was training law enforcement officers in early 2007 on tools such as Cell Seizure and Data Pilot, and free programs such as BitPim, Oxygen Nokia Phone Suite, and SIMCon, a student retold a story of two people who worked at a nonprofit company who ventured into a cellular telco store and saw a black box that was communicating with a phone and transferring the data to an SD card. They had previously seen this device at a trade show for electronics and law enforcement and asked the company if they could try it out. The nonprofit had trained law enforcement officers on digital forensic examinations and conducted mobile device collection research, and the employees believed this tool might assist law enforcement. But the nonprofit employees had been turned away, so they purchased the same unit that they had seen used in the store.

This is how the store device worked: The customer's old mobile phone data needed to be transferred to the newly purchased mobile device. So the old phone was connected and the data was copied to an SD card. The new phone was then connected and the data from the SD card was transferred to the new phone. It later progressed to attaching one phone to the left and another to the right and moving the data from one to the other. This enabled a user to come into the store with an old device and transfer contacts, call logs, and anything else to the new phone. Law enforcement officers were interested in the technology because they had been conducting data recovery from mobile devices, and the ease of use was captivating. By word of mouth, more law enforcement officers started purchasing the data transfer version as used in the store. Soon the device manufacturer realized the demand and interest, and using new ideas from the officers, began to manufacture a new forensic model of the tool. This model was nearly the same tool used by the telco to transfer data between phones, but now the device transferred data from a device to a removable USB stick and the term *forensic* was assigned.

Shortly thereafter, many other companies began designing forensic versions of their data transfer software. Companies such as Cellebrite, Compelson, Oxygen, and Susteen all migrated to the forensic software business from their initial start in the phone data transfer market. It has been said that Compelson began to manufacture MOBILedit Forensic based on its data transfer software MOBILedit before any of the other companies produced similar software.

Mobile Device Technology and Mobile Forensics

Mobile device forensics is likely the most rapidly advancing discipline that digital forensics has ever seen or ever will see, primarily because of the rapidly changing environment of the actual devices. Device operating systems have become more advanced, and the storage capacity on the current devices is astronomical. Today's devices are mobile computing platforms, but accessing the data contained on these devices is much more difficult than accessing data from any other digital device. Communication to these devices with software is much different from communicating with other media containing digital data. Users must use tools specifically designed for mobile devices, and they use many different software tools on one device. This brings a new wrinkle in the examination process. A lot of seasoned digital forensic examiners are disappointed and turned off by the limitations imposed by software to these mobile devices, believing that mobile devices should behave like a computer's hard drive. Also, many mobile device forensic software products have their roots in the sync software world and not in the forensics world. This does not help their credibility with examiners coming from the computer forensics lineage.

From Data Transfer to Data Forensics

Mobile sync software enables the user to add or delete items such as contacts and calendars from a mobile device. This is extremely satisfying to the standard mobile device owner, because it allows the easy addition of hundreds of contacts and contact details. No longer does the user have to key in every contact via the keypad; now, with the assistance of PC software and a cable, this information can be quickly added to the device.

Adding and deleting user data would never be something a forensic examiner would desire, a fact that was realized by companies developing sync/data transfer software. So these companies began to create models and versions that were intended for the forensic examiner. The software and hardware were no different from the sync software, with one exception: The write/transfer button or selection was not enabled. These devices and applications simply allowed the data to be read from the device, not written to it. Sometimes the forensics updates involved a removal of a button, and sometimes it was merely adding new instructions in the manual. The underlying code base in the equipment or software was the same; only the implementation was different —along with the manufacturer's suggested retail price (MSRP).

Real-World Example: Testing the Theory

This research example was conducted while I was preparing for testimony on a mobile device case in which a computer forensics expert was challenging the lack of usage of a write blocker when conducting an examination of a mobile device.

Using a Tableau USB write blocker and a Motorola Razor V3 mobile device, I began the experiment. The object of the research was to test my theory on write-protection devices and mobile devices. My theory was that since a mobile device is not observed by a PC's operating system as a mass storage device, the Tableau USB write blocker would not stop writes targeting the mobile device. (Note that this is not a limitation of the Tableau USB write-blocker product. This product, as well as all others, will operate the same as outlined in this example.)

I plugged the mobile device into the USB port of the Tableau write-blocking device. I then plugged the USB write blocker into a USB port located in the PC running Microsoft Windows.

The Tableau device immediately identified the device in the digital readout: "Motorola." Using P2K Commander, a free tool for browsing a Motorola PK device, the device's file system was read into the software. The entire device's file system was available. With the options in P2K Commander, I created a new folder within the device's file system. Since a write protector was in place, it would seem plausible that the folder would not be created. Unfortunately, however, the folder was created in the device's file system.

With the write-protector device still in place, I copied a file called WriteProtector.txt into the device file system and into the folder that I had created. The file showed that it was successfully transferred. I then shut down the software and collected data from the device using BitPim. The file system, as expected, contained the folder and the file that I had previously moved onto the phone with P2K Commander while the write protector was in place.

What did this mean for the write blocker used in this test? Did this mean that the write blocker did not operate as it should? Did this mean the test was not forensically sound? Of course not, but you must understand that these devices protect for writes only to mass storage devices, not devices viewed as something else by the computer's hardware. Because the Motorola was viewed as a modem, the connected computer could write any data it wanted to the attached media because writes to the modem were not halted.

As an examiner entering this field, you need to understand that the process you use to examine the mobile device and comprehend what has changed on the device is what makes the examination forensic. It is neither the software nor the hardware you use, but solely the process that you use during examination that makes it truly forensic. This concept applies not just to mobile device examinations but also to the examination of all digital data in today's electronically driven world.

Processes and Procedures

If automated tools will be used in any forensic examination of digital data, examiners must have a set of processes and procedures in place; using an automated tool without such direction could be a detriment to the entire process. An understanding of the processes and procedures of mobile device forensics is required in this age of mobile computing because of the many automated tools, variances among them, lack of complete coverage, and diverse training levels. Ultimately, understanding the processes and the steps needed, interpreting results, analyzing output, and documenting the data, all while maintaining case integrity, should be the goals.

People use mobile devices every day to buy groceries, send birthday wishes, break up with a lover, transfer millions of dollars, and unfortunately commit crimes. Gaining an intimate knowledge of how to collect valuable data from these devices, uncover valuable data, and present the data will put you ahead of the rest. Chapter 2 will discuss how to compile standard operating procedures for collecting the physical mobile devices and accessing data on those devices. Your understanding of the device and the progression into the actual digital investigation is critical for a successful outcome.

Examination Awareness and Progression

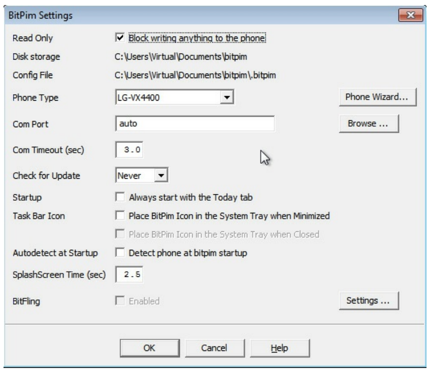
When mobile devices contained volatile data storage, the only examination of the mobile device consisted of removing the SIM card and reading the information stored on it. This made for a quick examination of the smart chip, and the device was seldom examined forensically. So essentially the first introduction anyone had to the examination of a mobile device started at the SIM card. The first software for mobile device collection also read the data from a SIM card. (In Chapter 8, you will be shown the various collection methods and software used for SIM cards.) As mobile devices advanced in terms of how data was stored, the methods used to collect the data from the internal memory, or nonvolatile area, did not progress in tandem.

The awareness of the data was there, but collecting it by means other than scrolling through the device and documenting the information was extremely limited. The first examinations of data by law enforcement on a Motorola StarTAC or Nokia 1610 Plus involved simply turning on the device after reading the data from the SIM, scrolling through the menus, and writing by hand the calls made and received and the contact information. This information was then transferred to a report, and the first mobile device examination was out the door. This worked well because the devices held a limited number of calls and contacts, but imagine using this process with a mobile device of today! Imagine documenting 5000 contacts into a spreadsheet by hand, including all the associated e-mail addresses, phone numbers, and addresses, not to mention scrolling through the selfies.

Cameras eventually began to play a role in the documentation of data for mobile devices. Important SMS messages, images, or contacts could be documented using a webcam or digital camera. These images were then placed into a report for review. Several companies built systems to enable the examiner to lay out the evidence, document the size, and more.

Soon the examination progressed to using software as a few software applications became available. The early mobile device software applications used in the examinations of cell phones were developed by engineers to deal with their own personal mobile device problems—such as how a phone book could be saved, edited, changed, modified, and exported without having to use the phone's number pad to reenter information.

Software applications such as BitPim, initially developed by Roger Binns, is a perfect example. Binns created an application that would enable him to transfer data, primarily his contacts, from an old phone to a new phone. He explained that it had been extremely tedious to add contacts, get pictures, and move information on and off an LG phone, so he decided to write an application to make it easier, and BitPim was born. At first, BitPim enabled data to be written to and read from the cell phones supported, which was forensically sound. At the time, Binns had not seen the need to allow only data to be read, but soon he was contacted by a mobile forensic examiner and asked if he would allow users to turn off the ability to write data to the phone. BitPim software was being used in Mobile Forensic, Inc., courses and primarily for law enforcement officers, so Binns kindly accepted the reasoning and soon he added a Block Writing Anything To The Phone checkbox to the settings, as shown in the following illustration. BitPim was limited mostly to CDMA devices, but because protocols were based upon the Qualcomm chipset, some Samsung GSM devices could be supported as well. GSM devices, however, were never officially supported by BitPim, and the software's last official build was on January 24, 2010.


BitPim Settings

Read Only	[✓] Block writing anything to the phone
Disk storage	C:\Users\Virtual\Documents\bitpim
Config File	C:\Users\Virtual\Documents\bitpim\.bitpim
Phone Type	LG-VX4400
Com Port	auto
Com Timeout (sec)	3.0
Check for Update	Never
Startup	[ ] Always start with the Today tab
Task Bar Icon	[ ] Place BitPim Icon in the System Tray when Minimized
	[ ] Place BitPim Icon in the System Tray when Closed
Autodetect at Startup	[ ] Detect phone at bitpim startup
SplashScreen Time (sec)	2.5
BitFling	[ ] Enabled

More and more developers began to contribute, and more and more devices were added. More applications from actual cell phone manufacturers such as Nokia and Samsung were developed for the transfer of user contacts, SMS, and more. These proprietary software suites began to be used regularly by law enforcement officers to perform extractions and collections so they did not have to go "commando style" on a cell phone to get the information manually. Applications such as Paraben Cell Seizure, TULP2G, SIMCon, and MOBILedit began to be used in training courses and forensic labs around the world. These applications gathered data from mobile devices by using the same protocols reverse-engineered from tools such as Nokia Suite and Samsung PC Suite. Today, as in the beginning, the same problem exists: not all mobile devices are supported by a single solution. Now, bring in drones, IoT (Internet of Things) devices, wearables, infotainment, and other mobile-to-mobile (M2M) devices, and the extraction and examination issues are only exacerbated. There is not and will not be a single solution to extract and collect data from all digital mobile devices, simply because there are too many kinds of devices, and new devices are being added and changed continuously.

Tip

Using multiple tools to examine digital data from a mobile device should never be observed as poor practice. Quite the opposite is true: using multiple tools enables not only verification of data, but more importantly, it enables you to uncover additional data the initial tool either did not support or simply missed.

Examiners in the beginning of the mobile forensic time line had two types of evidence to process: a SIM card and the mobile device. Sometimes the only data that could be collected from a mobile device came from a SIM card. There were, and still are, many variables to be considered: Did the actual device even have a connection to communicate with the software? Did the device store data internally? More importantly, was the device even supported? At this time, a new trend emerged: devices were now capable of holding data on a removable card, known as a memory card. Yes, the first cards were not microSD cards, but large PCMIA (Personal Computer Memory Card International Association) cards primarily used for PCs. These obviously were very large, so transitioning to CompactFlash (CF) cards moved them from the PC to the camera and into the many PDA devices for storage.

With the addition of removable media, mobile device forensics expanded to three pieces of evidence to collect during any examination. CF cards are still available, but the Secure Digital (SD) cards are now most prevalent in mobile devices. These small SD cards range in storage capacity, but the devices of today are no longer limited to reading 128GB, 256GB, or even 400GB—some devices tout the ability to read 2TB cards! Obviously, devices can contain multiple SD cards or SIM cards, but the primary items to be examined and collected will remain the same: device, SIM, removable media, and sometimes a backup of the mobile device that is located on another type of media, such as a PC. Today's mobile devices primarily use the microSD card as the de facto external storage medium.

With the ability to extract and decode data from the cloud, today's examiner now has an additional repository for mobile device evidence. In Chapter 4, the discussion turns to the storage of today's devices and another treasure trove of evidence. Digital devices for years have been storing data on servers located around the globe. This fact is easily observable every time a user upgrades to a new mobile device. The user simply downloads the app to the new device, signs in, and all his or her history magically appears within the app, kindly planting the server's data contents on the new device. Only recently have mobile device examiners had easy access to these endpoint storage cabinets with automated software. So now add another evidence area the savvy examiner must become familiar with.

To help you understand the multiple points of examination of a mobile device, let's take a general look at these data storage points and what information might be stored on each.

Data Storage Points

With multiple places to store data, mobile devices and cellular technologies, along with their forensic examinations, have become more advanced as the years have progressed (see Figure 1-2).

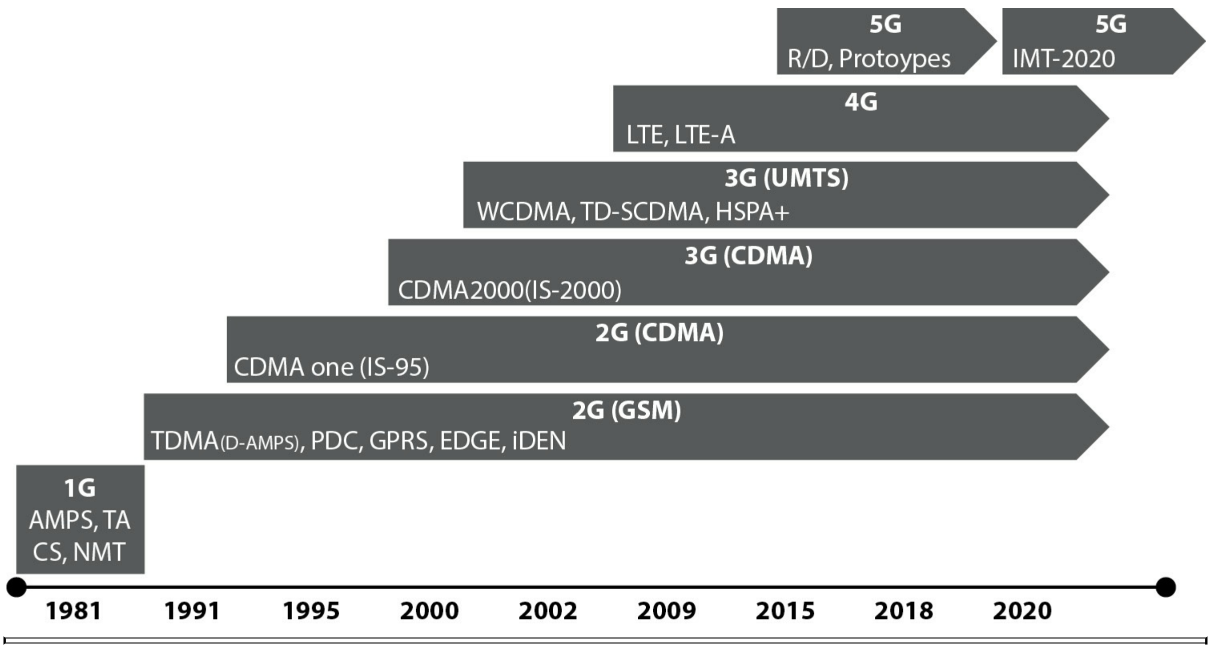


FIGURE 1-2 **The cellular generational time line represents the various transitions in technology related to mobile devices from 1G to our current LTE-A, and soon 5G.**

Mobile Technology Terminology and Acronyms

It is important that you understand the current terminology used in the industry before you read more about using storage points for evidence. Acronyms are used throughout the book and also in most digital forensic reference materials. Technological terms, reference documents, and government agencies all use acronyms, and the following list includes the most prevalent terms in the industry. Some terms describe cellular network functions and transmission techniques.

Note

The following provides an overview, rather than a complete, detailed description. There are entire books describing these transmission methods in great detail. This book is intended to expose these terms generally, so, if needed, you can conduct further research on your own.

AMPS/NMT/TACS

In the 1980s, the Advanced Mobile Phone System (AMPS) was the analog standard used in the United States, Nordic Mobile Telephone (NMT) was the analog standard in the Nordic countries, and Total Access Communications Systems (TACS) was the analog standard in the United Kingdom. Analog transmissions operated in the 800 MHz band between the frequencies of 824 MHz and 894 MHz in the United States, 450 MHz and 900 MHz in the Nordic countries, and in the 900 MHz range for Europe. The transmission and the reception of the data ranged from 20 kHz to 50 kHz so transmissions did not collide. Obviously, with the limited number of devices that could operate simultaneously, it was easy to listen to conversations on radio devices tuned to the same frequency. This technology was later termed 1G, for first-generation, cell phone transmission technology. AMPS expired February 18, 2008.

Governing bodies for mobile device transmission understood that additional data should be included in the transmission, such as text messages, while also allowing for more mobile devices to use the same frequency channel. For this reason, analog would need to transition to digital data. Mobile devices purchased for use that were labeled single-band indicated that the device could use only a single cellular band. Because of the varying frequency differences among countries, a device might not work in another country if the cellular band was not the frequency band within which the device was capable of operating. Because of this limitation, device manufacturers began to add hardware capabilities to allow for multiband frequency support, moving first to dual-band, then tri-band, and eventually quad-band support.

TDMA/GSM/CDMA

Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA) are 2G (second-generation) technologies, and these devices entered the handheld wireless scene in the 1990s. The main standards for 2G are GSM, Interim Standard 95 (IS-95) (CDMA), IS-136 (digital amps), and iDEN (Integrated Digital Enhanced Network). The biggest change was from analog to digital with the advent of 2G systems, along with digital encryption of the transmission. The first commercial 2G network on the GSM network was first made available by Radiolinja, a Finnish operator.

2G also brought the ability to send data other than voice over the wireless network; SMS was born, with the first text reportedly sent from a computer to a mobile device on December 3, 1992, from Neil Papworth to his colleagues at Vodafone, that said, "MERRY CHRISTMAS." 2G technologies slowly improved, and the progression toward 3G was slow. 2.5G, or GPRS (General Packet Radio Service), improved network stability, and 2.75G, or EDGE (Enhanced Data rates for GSM Evolution), improved the speed of transmission and was a large leap from 2G technologies.

Note

It used to be extremely easy to distinguish the various GSM or CDMA wireless companies simply by looking under the device battery. If the device had a SIM card, the device was a GSM device; if it did not have a SIM card, it was a CDMA device. In today's world with multiple-band devices, it has become common to see SIM card slots in CDMA devices.

UMTS/CDMA2000

Universal Mobile Telecommunications System (UMTS) and IS-2000 (CDMA2000) are the 3G, or third-generation, cellular systems established by specifications outlined in IMT-2000 (International Mobile Telecommunications), which introduced large gains with Internet access and video and data streaming in the early 2000s. The Third Generation Partnership Project (3GPP) standardized UMTS, and 3GPP2 (Third

Generation Partnership Project 2) standardized CDMA2000. UMTS uses W-CDMA (wide-band CDMA) for transmission and CDMA2000 using EV-DO (Evolution–Data Optimized). UMTS was used with GSM devices in Europe, China, Japan, and most other areas of the world, while CDMA2000 and CDMA devices were used primarily in North America and South Korea.

UMTS upgraded to HSDPA (High-Speed Downlink Packet Access) and was combined with HSUPA (High-Speed Uplink Packet Access) to form HSPA (High-Speed Packet Access), which is still the most widely deployed technology globally. What might be referred to as 3.5G is debated as HSPA+ and also LTE (Long Term Evolution). LTE is often mentioned with regard to 4G technologies, but the data upload and download rates do not meet the standards set forth by the ITU (International Telecommunication Union) to call it 4G. The first commercial LTE networks were launched in Norway and Sweden in 2009.

LTE-Advanced

LTE-Advanced (LTE-A) is also defined by the ITU, and it is a true 4G system. LTE-A was standardized in 2010 by the 3GPP and is dependent not only on the infrastructure of the underlying cellular network but also on the processors within the mobile devices.

IMT-2020 (5G) International Mobile Telecommunications (IMT) systems 2020 is also defined by the ITU, and it is currently a theoretical 5G system. As of this writing, IMT-2020 is being debated and developed, and is said to be a single unified standard for mobile communication and data. The current specifications from ITU indicate a 20Gbps (gigabits per second) download capacity for a single cell as a minimum, with support for one million connected devices per one square kilometer (0.39 square miles). Compared to the current LTE download capacity of 1Gbps, the infrastructure and demand on the telecommunication companies around the world will be significant. Features will include simultaneous wireless connections to a single device using a new Wireless World Wide Web (WWWW). This type of wireless connection will significantly impact the use of wearables and other IoT devices.

Mobile Device

The mobile device evolved from a simple handheld device to communicate via voice into a mobile computing device capable of transmitting large amounts of data instantly. The movement in technologies and the air-side of the network was never intended to create a clearer conversation, but to increase how much information one could send or receive in the shortest amount of time.

A mobile device can contain many pieces of information, from contacts, SMS, and call logs in early devices, to complex documents, applications, media, SMS, Multimedia Messaging Service (MMS), call logs, e-mail, calendar, notes, and contacts in today's devices, which are actually small computing devices. Built-in applications, or apps, can store additional data types, communicate with the Internet, launch rockets, play music, map a house, direct a vehicle to a destination, determine internal body vitals, and conduct bank transactions. There is almost nothing that a person cannot do with these smart devices—including activities that only ten years ago required devices thousands of times their size. Today's current smart devices come with standard apps, but a user can also install one of approximately seven million apps available on various mobile phone app distribution points, which makes a "standard examination" of the contents of mobile devices a nearly impossible exercise.

Mobile devices come in all shapes and sizes and have progressed from holding circuits to more than 256GB of storage space. What is most important for the mobile device forensic examiner is the underlying operating system. Knowing the type of operating system will help you determine the course of action you should take in the examination. Knowing this will also determine the actual software that you should use to complete the collection. As you read through this book, you'll learn more about these devices, and it will become apparent that your examination should start with an understanding of various types of evidence.

SIM and UICC

Subscriber identity modules, or SIM cards, were developed to enable portability and, more importantly, to store information to enable authentication on the cellular network. *Authentication* in this sense means that the device can register and enable users to make and receive calls. With the SIM card acting as the key to the network, a user can switch equipment by removing the small smart chip and inserting it in a different GSM device. As SIM cards began to enable the storage of user data such as contacts, SIM removal also meant that users could take their contacts with them to the new device. When a SIM card is inserted into the new device, the user can access contacts and make and receive calls. Storage on the SIM card makes it easy to move from one device to another. Along with contacts, SMS are also stored, as are the last numbers dialed.

All data types on a SIM card are not infinite, and the amount of data that can be stored is a static value determined by GSM standards. As outlined in the initial documents by ETSI (European Telecommunications Standards Institute) and later adopted by 3GPP, the "Technical Specification Group Terminals Specification of the Subscriber Identity Module" outlines the type of data along with how many records can be stored for each type of data. The SIM cards in today's GSM and CDMA devices are called UICCs (Universal Integrated Circuit Cards) and assist in the operation of the UMTS, HSPA, and LTE systems. It can be said that the UICC will work on all mobile telecommunication networks.

Some say, incorrectly, that a SIM card is the same as a USIM (Universal Subscriber Identity Module) card. Although technically SIM cards are physical cards that operate only on 2G GSM systems, a USIM is an application on the UICC. The USIM application enables the mobile device to be identified on the UMTS, HSPA, and LTE systems. UICCs also contain a SIM application that allows for backward-compatibility to the 2G networks and may also contain another application called the CSIM (CDMA SIM), which allows access to CDMA networks, and an application called ISIM (IP-multimedia Subsystem Subscriber Identity Module). The ISIM allows for secure use of the Internet Protocol multimedia subsystem and is the backbone of the LTE network. This means support for voice over IP (VoIP), SMS, and emergency calls without a network can occur securely. One of the biggest advantages of the UICC is this ability to store multiple applications on it.

Note

Earlier CDMA SIM cards were called R-UIM (removable user identify module) cards and contained a primitive version of the CSIM application and a SIM application for GSM.

Like the SIM, the UICC also allows for the storage of contacts, SMS, and other user histories. What has not changed from the SIM to the UICC or USIM application is the ability to store more SMS messages or more call detail data. The reason behind that has to do with the understanding that today's devices default to storing data to the handset, not the SIM card. Along with the velocity and volume of this data, this is not likely to change. In Chapter 11, you can read about the actual structure and the evidence locations for specific SIM and UICC data.

Media Storage Cards

SD cards were created in an effort to expand the available storage for mobile devices. SD cards are of four different types: Standard-Capacity (SDSC), High-Capacity (SDHC), Extended-Capacity (SDXC), and a newer type not used currently with mobile devices, called Input/Output (SDIO). These four types come in three different form factors: the original, which measures 32-by-24 mm; the mini, which measures 21.5-by-20.0 mm; and the micro, which measures 11-by-15 mm.

Tip

External micro-sized SD cards typically come with an adapter that enables the cards to be read using a standard-size SD card reader. Important to remember is that most SD card full-size adapters have a "lock" that disables writes to the SD card media. When processing these storage mediums, you must set the card to locked to enable the write-protection feature built into the card.

Today, mobile device equipment often contains a slot to accommodate a small storage card that stores saved images, videos, or other media, but can also store documents, applications, and more. Important to remember is the fact that external media cards are another form of evidence. In the early years of mobile devices, external media cards were sized at 256 to 512MB. The most prevalent type and form factor used in today's devices are the original and microSDHC cards. The SDHC standard, released in 2006, supports a total capacity of 32GB, and the micro size is the most prevalent type of card with mobile devices. The SDXC, released in 2009, supports up to 2TB of data by using Microsoft's exFAT (Extended File Allocation Table) file system, and some current mobile devices will support and recognize cards in the micro size up to 2TB. However, the purchase of legitimate operational 400GB SDXC for a mobile device is now possible.

Note

To help you understand just how much data can be stored in a gigabyte, let's discuss what a gigabyte is. A gigabyte is equivalent to a device's ability to store 10,000 one-page documents in the Times New Roman font, 12-point, single-spaced; 520,000 text messages; 1074 regular sized photos; and approximately 250 songs that are each four minutes in length. With the sheer volume of data that can be stored on media smaller than a postage stamp, it becomes apparent that omitting this type of evidence from your mobile forensic exam could be catastrophic.

Many new devices, especially those running the Android operating system, also have internal microSDHC and SDXC cards soldered to the internal circuitry to add internal storage capacity. This is an inexpensive way to expand the capacity of a device. Understanding and recognizing that a device has internal microSD storage is very important during your digital exam. This internal storage is in the range of 4 to 64GB, which can hold an astronomical amount of data. More devices today contain both internal and external microSD cards. During a proper examination, if possible, you should collect and analyze both storage locations.

Tip

To determine whether a device contains both internal and external SD card components, research the device before starting your examination. This information can be obtained from web sites discussed in the coming chapters.

Mobile Device Backups

The last type of associated data for mobile devices is the backup that some devices create on media such as PCs, Macs, and other operating systems. Today's devices also use the Cloud to store backups. Devices such as Apple iPhone and iPad and the new BlackBerry smart phones create backups that you can later analyze. Apple iOS devices create a backup using the Apple iTunes software. You'll find these backups in different locations depending on the type of computer operating system and version.

For Windows computers, the backup default location for an Apple iOS device will depend on the operating system version:

- **Windows XP** \Documents and Settings\<*username*>\Application Data\Apple Computer\MobileSync\Backup\
- **Windows Vista, Windows 7, 8, 10** \Users\<*username*>\AppData\Roaming\Apple Computer\MobileSync\Backup\
- **macOS** ~/Library/Application Support/MobileSync/Backup/

These areas contain valuable data and can also contain multiple backups of the same device. For an iOS device, the data that is typically backed up as outlined by Apple in the support wiki (http://support.apple.com/en-us/HT4946) is listed here:

- Camera Roll photos, screenshots, images saved, and videos; for devices without a camera, Camera Roll is called Saved Photos
- Contacts and Contact Favorites
- Calendar accounts and subscribed calendars
- Calendar events
- Safari bookmarks, cookies, history, offline data, and currently open pages
- Autofill for web pages
- Offline web app cache/database
- Notes
- Mail accounts (mail messages aren't backed up)
- Microsoft Exchange account configurations
- Call history
- Messages (iMessage and carrier SMS or MMS pictures and videos)

- Voicemail token (not the voicemail password; the token is used for validation when connecting and is restored only to a phone with the same phone number on the SIM card)
- Voice memos
- Network settings (saved Wi-Fi hotspots, VPN settings, and network preferences)
- Keychain (includes e-mail account passwords, Wi-Fi passwords, and passwords entered into web sites and some apps—can be recovered in an iTunes backup only if the backup has been encrypted)
- App Store app data (except the app itself, its .tmp file, and Caches folder)
- App settings, preferences, and data, including documents
- In-app purchases
- Game Center account
- Wallpapers
- Location service preferences for apps and web sites you've allowed to use your location
- Home screen arrangement
- Installed profiles
- Map bookmarks, recent searches, and the current location displayed in Maps
- Activity and Health (can be recovered in an iTunes backup only if the backup has been encrypted)
- Paired Bluetooth devices (only if restored to the same phone that did the backup)
- Keyboard shortcuts and saved suggestion corrections
- Trusted hosts that have certificates that cannot be verified
- Web clips

Some items are not contained within an iTunes backup reported by Apple (https://support.apple.com/en-us/HT204136), including the following:

- Content from iTunes and App Stores, or PDFs downloaded directly to iBooks
- Content synced from iTunes, such as imported MP3s or CDs, videos, books, and photos
- Photos already stored in the cloud, such as My Photo Stream, and iCloud Photo Library
- Touch ID settings
- Apple Pay information and settings

Note

A series of files and folders are created for the iOS device backups at these storage positions. If the user has elected to encrypt backups via the iTunes interface, the backups will not be readable unless the iTunes password is known.

BlackBerry also creates backups using the BlackBerry Desktop Management software, or Link as it is known today for some new BlackBerry devices. These backups are also located in a default location:

- **Windows** My Documents\BlackBerry\Backup

- **Mac** /Users/<*name*>/Documents/BlackBerry Backups

Note

A user can always change the backup location using the BlackBerry Backup Software.

Depending on the version of BlackBerry software that was used to create the backup, the file will either have an .ipd or a .bbb extension. Files with an .ipd extension will have been created with earlier versions of BlackBerry Desktop Manager (versions up to 6), while .bbb files are created by BlackBerry Desktop Manager version 7 and the new Link software. The .bbb files that are created by Link are fully encrypted, and known BlackBerry user information must be used to decrypt the data.

Tip

Both iOS and BlackBerry backups contain very valuable data that you can analyze in commercial tools such as Oxygen Forensic Detective and Analyst, MSAB XRY, and Cellebrite Physical Analyzer, along with free tools such as iPhone Backup Browser and MagicBerry. The collection and analysis of backups can assist your investigation if the device cannot be located or accessed, but examinations of the backups are also a way to compare the data from the backup to the actual device for a historical perspective.

BlackBerry Enterprise Server (BES) technology was the first platform that allowed an enterprise to store data from a mobile device in a central location on the company premises. This added great security to a company's critical and confidential data by enabling local storage of e-mail, documents, messages, event data, and company materials. This information could be accessed by company enterprise security teams to ensure compliance and gain visibility into the device if law enforcement requested data not stored on a device or data that had been deleted. Security administers of the BES can also remotely unlock a device if it is under control of the enterprise, which is also of great benefit to law enforcement. BES12 is an endpoint mobility management (EMM) tool and can contain data that is not accessible on the physical device, so don't overlook this information while conducting an investigation of a BlackBerry device.

As mentioned, devices can also create backups in the Cloud. iOS devices use a service called iCloud, which launched in October 2011, with almost 800 million users currently. The iCloud service enables the user to store user data, applications, documents, pictures, and more on a server maintained and owned by Apple. This information can be retrieved using tools such as Oxygen Forensics Detective Cloud Extractor or Elcomsoft's Phone Breaker with the user's Apple ID and password.

Note

For iCloud backups, a user can obtain most of the data from a standard iTunes backup with a few exceptions. The following items, as indicated by Apple's support site (https://support.apple.com/en-us/HT204136), are not included in an iCloud backup:

—Data that's already stored in iCloud, such as Contacts, Calendars, Notes, My Photo Stream, and iCloud Photo Library

—Data stored in other cloud services, such as Gmail and Exchange mail

—Apple Pay information and settings

—Touch ID settings

—iCloud Music Library and App Store content (if it's still available in the iTunes, App, or iBooks Store, you can tap to re-download your already purchased content)

Because of many security concerns with unauthorized access to iCloud backups, Apple utilizes end-to-end encryption as long as the device owner has turned on two-factor authentication. This security update has added another hurdle for law enforcement when legally obtaining information stored using the iCloud service and tools.

Storing backups on the Cloud has moved to Android to include Samsung Cloud, Google Cloud, Huawei Cloud, OPPO Cloud, and many others with onboard applications downloadable from the Google Play Store; BlackBerry has moved new devices to BlackBerry Protect. With many new devices moving to the Cloud, investigators must be cognizant of these locations as well. These cloud services, as well as methods to extract and analyze this data, will be covered in Chapter 4.

Educational Resources

You should always begin any investigation with research of the device to be examined. Knowing as much information about a device as possible will ensure that you won't miss internal and external components. Knowing, for example, that a device has a built-in 64GB internal SD card is very good information to know before you start an examination. Several resources online can assist you in identifying the many nuances of the millions of devices available.

Phone Scoop

Phone Scoop (www.phonescoop.com/phones/finder.php) contains valuable information to assist in the identification and research of the device to be examined. You can search by device model, make, or operating system to see information such as the current operating system and version, processor, memory (both internal and external), whether or not there is a SIM card, and the type, memory slots, and the FCC ID.

Phone Scoop has images of the device to help the investigator quickly identify the device visually and also contains links to query the FCC database. Querying the FCC database provides detailed summaries of the internal components, what the original equipment looks like, and where the label should be located and what it should look like.

Phone Scoop stores information on devices for North America only. Using this resource will help you as you research to gain a better understanding of the device, its features, and how to begin an examination.

GSMArena

GSMArena (www.gsmarena.com/search.php3) is much like Phone Scoop, but this site contains only GSM devices and is primarily for devices outside of North America. GSMArena is an information repository for reviews of the devices; phones searchable by model number, manufacturer name, and common name; and mobile device specifics. Mobile phone manuals are available for most devices, along with all the specifications, including but not limited to, size, memory, SIM type, operating system, hardware specs, USB port, and detailed images of the device.

Forums

Although Phone Scoop and GSMArena are valuable resources, obtaining information about what software might have been used on a specific device, tricks that you should know, and processing ideas all come from real experience. Forums are a way to connect with examiners and ask specific questions, search large databases for a device to see what software works best with that particular device, and so much more. The key to gaining experience and knowledge is by gathering as much intelligence prior to starting the actual examination of the device.

Here is a list of several forums that are good sources for information.

Forums with Databases

This forum contains a database that you can query in search of a device and the software to use.

- **Mobile Forensics Central** (www.mobileforensicscentral.com/mfc/) Use Mobile Forensics Central by searching its database to locate a phone report that lists the mobile device software and additional information needed for processing. MFC is a free site but requires registration.

Technical Information Forums

Use these forums to ask and search for mobile forensic technical questions.

- **Forensic Focus** (www.forensicfocus.com/computer-forensics-forums) This forum is for general forensic comments and questions based in Europe. This forum has a subforum for mobile forensics and contains tips, tricks, software information, and more.
- **Android Developers Forum** (http://developer.android.com/index.html) In this forum you'll find developer information on the Android operating system; you can also obtain all drivers for Android devices and download the Android Emulator along with prereleases of the Android operating system.
- **XDA Developers Forum** (www.xda-developers.com/) This site contains a forum along with development resources for smart devices, primarily Windows and Android. In 2010, a sister site, iPhone-Developers.com, was launched. The XDA name represents the first device, a Windows PDA, for which the site was developed.

Preparing for Your Journey

Mobile device forensics is no different from any other technical discipline. All demand a respect of the details along with an effort to understand every variable and contingency that you might encounter. There will be many bumps in the road with respect to tools and devices and, more times than not, a mixture of both. Working through each and every problem for a solution will arm you with knowledge that many others in today's mobile forensic world relegate to a tool.

Mobile devices will continue to be involved in the lives of nearly every human being on this Earth. They will continue to be used in some way in crimes, conversations, transactions, meetings, and Internet access for years to come and beyond. Mobile device examinations are not like any other digital examination discipline. To conduct thorough mobile device examinations, you need a constant appetite for research and training with this rapidly evolving discipline. Be patient, but more importantly, question the tools, understand what is under the hood, and prepare for anything!

Chapter Summary

Mobile device forensics is nothing like any discipline in the range of digital data examination because of the many different types of media associated with the single device (SIM, device, SD card, the Cloud, and even backups). Furthermore, the ways of acquiring the data from the devices are atypical to the standard digital forensic exam. The standard device collection is not the "bit-by-bit" image you would expect in other disciplines. Having a firm grasp on the basics will give you a strong foundation that will withstand the staunchest critique and challenge. Without having an underlying foundation, your examination will surely crumble under the scrutiny of an opponent.

2 Mobile Devices vs. Computer Devices in the World of Forensics

Mobile device forensics is a very specialized, and at times frustrating, discipline, especially for a new mobile forensic examiner crossing over to the discipline after years of service as a computer forensic examiner. With typical file system formats and well-documented operating systems and system processes, understanding a computer and its relevant artifacts is straightforward. With a legacy Nokia Series 40 (S40) mobile device, however, with its S40 operating system and no official documentation outside of Nokia, uncovering structure and artifacts may not be so straightforward. Mobile device storage media or internal flash is similar to the solid state drives (SSDs) in today's computers, but the file formats and, most importantly, the way to access this data as an examiner is much different. Additionally, the ability to create and obtain a physical image of a mobile device is much different from, and at times extremely more difficult than, creating a physical image of a mass storage device or traditional hard drive.

Note

When speaking about a physical image of a hard drive or mass storage device, most practitioners refer to obtaining every bit and byte from the first sector (or start of the hard drive) to the last sector (or end of the hard drive). Obtaining an exact copy of the media is the truest form for a computer forensic examiner.

Most seasoned computer forensic examiners struggle with the limitation of obtaining an exact, unaltered form of a mobile device's internal storage media. For this reason, many examiners say they will not conduct a collection or an examination of a mobile device and cannot refer to it as a "forensic exam." Some have even refused to refer to this as "mobile device forensics," instead calling it "mobile device collection or extraction." This mentality will last only so long, however, especially when an examiner conducting a mobile forensic exam becomes an expert. As soon as an examiner is held to be an expert in the field, his or her testimony will be weighed against the *Daubert* or *Frye* standard, depending upon the current jurisdiction.

Note

Frye is based upon *Frye v. United States*, 293 F. 1013 (www.law.ufl.edu/_pdf/faculty/little/topic8.pdf), which states that expert testimony must be based upon scientific methods that are sufficiently established and accepted. *Daubert* is based upon *Daubert v. Merrell Dow Pharmaceuticals*, 509 U.S. 579 (https://supreme.justia.com/cases/federal/us/509/579/case.html), which states, among other things, that scientific knowledge can be established if it can demonstrate that the conclusion is the product of sound "scientific methodology" derived from the scientific method, and the decision as to scientific knowledge laid out by Federal Rule of Evidence 702 will rest on the shoulders of the trial judge.

The world is quickly moving to mobility, and mobile devices are becoming more like small computers, with no end in sight. Examiners must begin to understand that the forensic process is more important for the examination of a mobile device than for traditional computer forensics, based upon the many differences and hurdles that currently exist with these devices.

The forensic process is important in the examination of a mobile device for several reasons. The greatest impedance to the forensic process is overcoming and recognizing that write-protected devices are ineffective in protecting the integrity of mobile device evidence. When corners are cut, training is insufficient, and a full understanding of the process is relegated to pushing a button, case law and precedence are often set. If the examiner has a concrete understanding of the process and follows it and understands the theory behind mobile device communication via software, then the evidence recovered will stand up to challenge. The flip side is uncovering a significant amount of useful information by not following or understanding the process and then being informed in court that recovered data cannot be used. What could be worse is that the evidence is used in the proceeding, but during the proceeding the validity is disputed because the practitioner used improper collection methods or did not follow a formulated scientific process. This concept is difficult to accept for examiners with many years in the computer forensic discipline.

Computer Forensics Defined

The examination of digital data from a computer's storage medium, either the traditional hard drive or the SSD, is a discipline familiar not only to law enforcement but also to enterprise and academia. In every vocation, digital data from computers has been examined to verify, confirm, and attempt to determine users' intentions by investigating their actions and history. Processes and procedures govern the way a computer forensic examination and, more importantly, an examiner should proceed throughout the investigation. When you understand how prominent organizations define computer forensics and know the minimum requirements for a successful digital investigation, you can establish proper processes and procedures, implement plans, and establish a structure to facilitate mobile forensics.

International Association of Computer Investigative Specialists (IACIS)

IACIS is an international volunteer-based nonprofit corporation that provides training and education in forensic computer science and grants the Certified Forensic Computer Examiner (CFCE) certification. The organization membership was originally limited to law enforcement but is now open to computer forensic practitioners who qualify. The CFCE core competencies document describes computer forensics as the acquisition, reconstruction, examination, and analysis of data stored on electronic media. To become certified, the examiner must address seven competency areas in the CFCE program, obtain peer review, conduct practical examinations, and pass a written examination. The seven areas are pre-examination and legal issues, computer fundamentals, partitioning schemes, Windows file systems, data recovery, Windows artifacts, and the presentation of findings.

International Society of Forensic Computer Examiners (ISFCE)

ISFCE is a private organization that conducts research and development of new and emerging technologies in the science of computer forensics. ISFCE grants the Certified Computer Examiner (CCE) certification, which also requires that core competencies be met in the certification process. These competencies include ethics, law, software, hardware identification, networks, operating systems, seizure process, forensic examination procedures, file systems, other media, media geometry, preparing media for imaging, forensic boot disks, low-level analysis, processing issues, and practical examination skills. To be accepted into the CCE testing process, a candidate must have completed at least one of three requirements: attended a CCE Boot Camp from an authorized training outlet, possess at least 18 months of verifiable digital forensic work, or have documented self-study in the field of digital forensics deemed appropriate by the CCE board. Once this requirement is fulfilled, the candidate can apply to the CCE testing process and must pass both an online written assessment and practical examinations.

Applying Forensic Processes and Procedures

Both IACIS and ISFCE follow similar processes and procedures in their competency requirements before a candidate can become certified to examine computer digital evidence. The following sections cover specific parts of both organizations' competencies along with supplemental information obtained from the U.S. Department of Justice (DOJ) on electronic evidence. By understanding the following areas of concentration as outlined for computer forensics methods, you can formulate strategies to build the best approach for a mobile forensic examination. Having a competent examination process that is repeatable and that subscribes to the scientific method at the onset of the journey will help you create a firm foundation and reliable process. This foundation and process must hold the weight of the facts when you testify to the results of any examination. Many certifications in today's mobile forensic community are from tool vendors. Having a tool-specific vendor certification may be important for testimony that competency in the tools is vetted. However, having an overall understanding, competence, and firm foundation in general and advanced mobile forensics is essential, and this is seldom offered by a tool vendor.

Seizure

Any investigation into electronic evidence must start with the legal seizure of the device that is holding, was holding, received, or transmitted the electronically stored information (ESI). The proper legal steps will be determined by the situation. Is the ESI in a place that requires a search warrant to be obtained, that is corporately owned, or that requires permission be given by the owner? If the seizure of the device and subsequent data are tainted by questions on the legality of how the data was obtained, the information collected will be dismissed in subsequent proceedings. You must exercise extreme care at the onset.

Collection

You must extract data from a digital device in a manner that enables you to show that the ESI did not change, was not altered, and is the same as when it was collected. The collected data typically contains an electronic fingerprint. If any data is changed, added, or removed from the collected data container, the electronic fingerprint will change. The integrity of the collection of digital data and the validation and verification of the software that collected the device are the responsibility of the person conducting the collection.

Analysis/Examination

Analyzing the data that has been collected from the seized device is often the most tedious part of an examination. Looking into many gigabytes of information is labor intensive even with automated tools. Because of the volume of data, typical digital examinations are predicated by the type of investigation the data will be used to support. Using the DOJ's "Electronic Crime Scene Investigation" document as a guide, the examiner can steer the examination of digital evidence by the crime or incident being investigated. By using this type of direction, an examiner's time can be better used in the analysis of pertinent information on a case-by-case basis. For example, the guide states that for crimes against persons, investigations and electronic data examinations should focus on the recovery and analysis of images (pictures), Internet activity logs, legal documents, and wills, along with any research the suspect conducted on the victim. The analysis and examination of the digital data can take significant time, but if the examiner has a clear picture of what information might be required for a particular investigation, the time can be minimized.

Presentation

Once the analysis is completed on the seized and collected data, the examiner must present the information, typically via a written report. The presentation is the most important piece for those who will review and most likely act on the information that has been recovered and analyzed. The presentation must outline the entire process, including any problems encountered, from the seizure to the analysis. The analysis portion must be clearly documented as to the request, the methodology, and findings. The examiner may have the most difficulty at this stage because technical examiners often do not communicate well with nontechnical people. The presentation stage defines the entire examination, however, since those who review the process will use this information to determine the validity of the examination and its results.

Approach to Mobile Device Forensics

Using well-tested and clearly defined forensic procedures can be a great start to creating a process and procedure for mobile device forensics. First, however, you need to understand several items that are at the heart of the scientific debate over the validity of mobile forensics.

One of the major debates by seasoned computer forensic examiners has to do with the integrity of image creation. Because a write-protect device cannot be used in the examination of a mobile device, the image cannot be substantiated as a true representative of best evidence—or can it? Most computer forensic examiners consider mobile device forensics *nonscientific* because of this single limitation. As discussed previously, a write blocker stops writes to a mass storage device, thus maintaining the integrity of the device from which the image is being created. The hardware device or software switch inhibits writes to ensure that data is not overwritten and allows for a duplicate image of the storage device. Thus, the examiner can obtain a *hash*, or mathematical fingerprint, of all the data on the device. Because a mobile device is not recognized as a mass storage device and a write blocker cannot be used, some examiners believe the image must be labeled unreliable. However, when you fully understand the nature of the software used to connect to and collect data from the mobile device, you realize that the image can be reliably captured, even without involving a write-protect device.

When plugged into a computer, mobile devices initiate a change in the computer's operating system, which recognizes that a mobile device has been plugged into the system. Furthermore, the mobile device also makes changes to its operation to allow for communication with the computer. Mobile devices can be tethered to a computer using several means: infrared (IR), Bluetooth, Wi-Fi, serial cable, and USB cable. Connection with the device will always need a way to communicate, and this requires a driver—a conduit, command set, or program for a particular device or device set that allows communication between the device and the computer. In essence, a driver "bridges the gap" between the devices. Drivers are used not only for mobile devices but also for any hardware attached to a computer system. If a driver is not installed properly, communication cannot occur. Drivers are the primary "pain points" when it comes to processing a mobile device. The correct driver must be installed correctly for communication to work while processing a mobile device. The communication between a mobile device and a computer system could involve the transfer of data to and from the device to the computer, creating an Internet hotspot, installation of applications, and many other things.

The word *communication* is important. For the mobile device to be recognized and communicate with the system via the driver, the mobile device typically must be powered on, and this fact brings up another point of contention with computer forensic examiners. If the device is powered on, then it is possible that data is constantly changing on the device from the cellular network or Wi-Fi network to which it is connected. Would not the clock on the device continue to update along with various other running processes on the mobile device? Invariably, the data is in constant flux, but ultimately it is the responsibility of the examiner to determine what data has changed, if any. Isolation techniques will be covered in later chapters, but for now you can assume that data will change on the device when it is powered on. This is an incredibly difficult concept to comprehend for a computer forensic examiner coming into mobile forensics. Both IACIS and ISFCE specifically state that nothing should change on a device during the collection process, and if it does, the recovered data cannot possibly be used as best evidence.

In today's modern world, however, we have software tools that recover volatile memory from a running computer. *Volatile memory*, memory that goes away if a computer is powered off, includes Random Access Memory (RAM). RAM can contain very valuable information such as passwords, keywords, media, and files, along with many other great forensic items. Because RAM is volatile, if the computer is shut down, the data would cease to exist, would be purged from the system, and recovery would be impossible. So before you could collect this data, the computer must be live and powered on, and you would run a forensic application to target the machine, capturing the volatile data.

This live technique is in fact much like mobile device data collection. Files are changed on the mobile device just as the files on the computer change during a live collection. It is up to the examiner to understand and comprehend what is taking place from start to finish, and most importantly, to ensure that no user data is altered by the software during the collection.

Communication between the mobile device and mobile forensic software occurs via the driver connection using the device's protocols, and the communication must first be in the language (protocol) accepted by the device. Different protocols are used for different devices, and sometimes multiple protocols are used, depending upon the access needed to the device. Some devices must even have small applications installed via their operating system onto their internal storage to collect the required data during the forensic examination. When you consider the many different devices available to the consumer and the various protocols used, it is apparent why some mobile devices cannot be examined using communication via a USB cable, serial cable, Bluetooth, or Wi-Fi. It is important that the examiner develop a thorough understanding as to what communication is occurring between the software and the mobile device so that he or she can maintain the integrity of the image and the mobile device's contents can be verified as best evidence and untainted during the mobile forensic examination.

In 2008, NIST published a document entitled "Forensic Filtering of Cell Phone Protocols" (NISTIR 7516), which describes ways to use phone manager software tools believed to be non-forensic in forensic examinations. The document describes a protocol filter that can be applied to the software to intercept communication that poses a risk to the integrity of the investigation. The document, although dated, contains valuable information as to how forensic tools of today combat the limitation imposed when a write-protection feature cannot be used. The document explains that the underlying functionality of forensic mobile device tools is based upon the same protocols used by the manufacturer's phone management tools. Furthermore, the forensic software tools inhibit or restrict the protocols used to issue commands or instructions that will read data from the device, along with other functions that impose little risk to the integrity of the evidence. This is done, as NIST explains, using a filter between the software and the device—much like how hardware and software write blockers work.

Validating the mobile forensic software and verifying that it uses forensic filtering of common mobile device protocols are the responsibility of the examiner. The ability to add, remove, or change records on a mobile device during or after a collection and subsequent analysis of the device should not be available to the examiner or in the software available today. Of course, as will be explained later, there are certain instances in which writes do occur to the connected device. These writes are not indiscriminate to the device, but targeted writes to temporary storage in the form of an application that assists with the extraction of the user data.

After the examiner collects the data from the mobile device, the analysis of the data, much like in computer forensics, is the most time-consuming part of the process. At this phase, some examiners and examinations falter. A lack of knowledge on the structure of a device's internal file system, critical artifact locations, system values, and critical file types means most examinations target the "lowest hanging fruit." WYSIWYG data typically satisfies the majority of examiners, not because of lack of motivation, but because of a misunderstanding of the device's storage areas and operating system characteristics. Know the device, its capabilities, and its file system, and you can substantiate any challenge to the examination.

Unlike computer forensics examiners, those who conduct mobile device forensic investigations have no globally recognized standards, as outlined earlier in the chapter. Simply put, as long as the data from the mobile device is included in a report, there is no regard as to how that data was collected from the mobile device. It is because of this "Wild West" approach that most computer forensic examiners see mobile device forensics as mobile device *extraction*, which is inherently wrong if you understand what it really means to be *forensic*. Mobile forensics do, indeed, have processes, like computer forensics, but typically they are seldom adhered to by those conducting examinations.

This disparity is one of the main reasons the Mobile Forensic Certified Examiner (MFCE) curriculum was created and formulated by the training company Mobile Forensic, Inc. (MFI). The MFCE requires that a candidate receive training on mobile forensic practices or have verifiable work in mobile forensics to apply for certification. If accepted, the candidate then undergoes six practical exercises and a written test to receive certification. The MFCE has not been widely accepted, however, primarily due to its limited reach and resources. For that reason, in late 2014, the MFCE was incorporated into the IACIS advanced mobile device training as a certification process, where it will be maintained and governed by a board of examiners.

Most software vendors also provide training on the software that they develop and sell. This type of training is important because the examiner must fully understand the tool to use it correctly during mobile forensic examinations. Training also provides credibility when examiners testify in court. These vendors also provide certification, which typically involves showing competence via written examinations, practical experience, or both.

Note

If you are unsure about whether to receive a mobile forensic certification on your overall understanding of the process or based on a particular tool, my best recommendation is to work toward a cumulative certification on overall processes and procedures.

As shown in Figure 2-1, the current approach to mobile forensics and the analysis of a mobile device is a bit upside-down. The foundation shown in the figure is the tool, which is supporting the weight of the procedure, process, and training. But having only a tool-based foundation in mobile forensics will never support your conclusions. A foundation based on training, however, which represents the largest area of the pyramid, along with a core centered on processes and procedures, will support a tool-based approach. To be successful in the mobile forensic field, you will invert the current examination pyramid and rest your knowledge foundation on what will support the approach covered throughout this book.

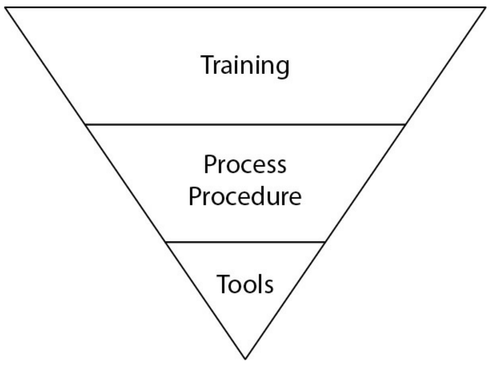
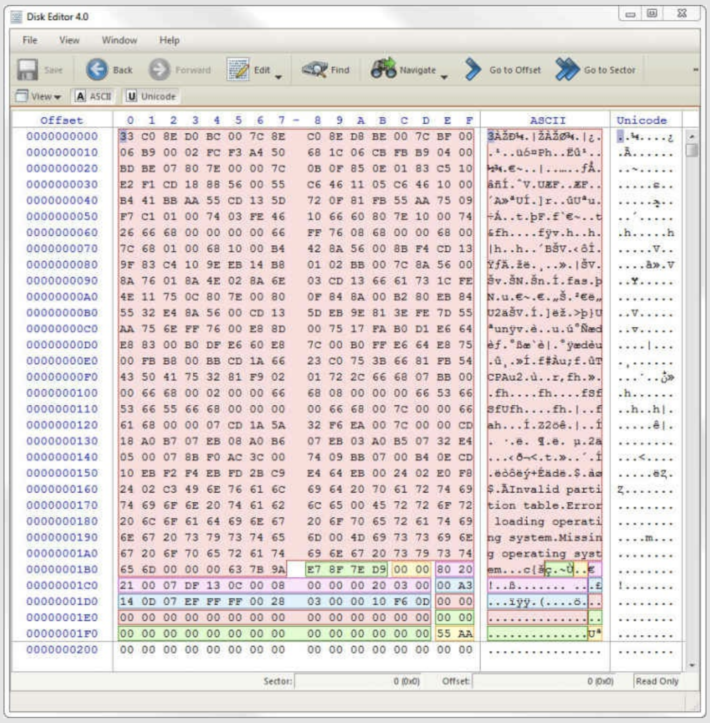


FIGURE 2-1 **The backward approach to mobile forensics**

Tool Understanding: Should It Be Painful?

When I first began in forensics, I attended several training courses. The most prominent other than my two-week IACIS training course was Basic Data Recovery and Analysis (BDRA) conducted by the National White Collar Crime Center (NW3C). In the week-long BDRA course, students started at the hard drive disk level. We learned about disk geometry, how the aperture arm worked, how the platters are configured, how the 30-pin ribbon cable worked, and the jumpers used to make one a master or a slave. We also learned how to create an image of a 3.5-inch floppy drive containing possible evidence. We then examined the image using a tool called a *disk editor*. The disk editor enabled students to examine the information at the data level, which helped us identify files by the file header and the file footer to determine the actual file type. The following screenshot shows what I was exposed to during the initial training phase and using a disk editor to examine the data on digital media.



```
Offset       0  1  2  3  4  5  6  7 -  8  9  A  B  C  D  E  F   ASCII             Unicode
0000000000  33 C0 8E D0 BC 00 7C 8E   C0 8E D8 BE 00 7C BF 00   3ÀŽĐ¼.|ŽÀŽØ¾.|¿.  ........
0000000010  06 B9 00 02 FC F3 A4 50   68 1C 06 CB FB B9 04 00   .¹..üó¤Ph..Ëû¹..  .Ā......
0000000020  BD BE 07 80 7E 00 00 7C   0B 0F 85 0E 01 83 C5 10   ½¾.€~..|..…..ƒÅ.  ........
0000000030  E2 F1 CD 18 88 56 00 55   C6 46 11 05 C6 46 10 00   âñÍ.ˆV.UÆF..ÆF..  ........
0000000040  B4 41 BB AA 55 CD 13 5D   72 0F 81 FB 55 AA 75 09   ´A»ªUÍ.]r..ûUªu.  ........
0000000050  F7 C1 01 00 74 03 FE 46   10 66 60 80 7E 10 00 74   ÷Á..t.þF.f`€~..t  ........
0000000060  26 66 68 00 00 00 00 66   FF 76 08 68 00 00 68 00   &fh....fÿv.h..h.  .h.....h
0000000070  7C 68 01 00 68 10 00 B4   42 8A 56 00 8B F4 CD 13   |h..h..´BŠV.‹ôÍ.  .....V..
0000000080  9F 83 C4 10 9E EB 14 B8   01 02 BB 00 7C 8A 56 00   ŸƒÄ.žë.¸..».|ŠV.  ....ä».V
0000000090  8A 76 01 8A 4E 02 8A 6E   03 CD 13 66 61 73 1C FE   Šv.ŠN.Šn.Í.fas.þ  ........
00000000A0  4E 11 75 0C 80 7E 00 80   0F 84 8A 00 B2 80 EB 84   N.u.€~.€.„Š.²€ë„  ........
00000000B0  55 32 E4 8A 56 00 CD 13   5D EB 9E 81 3E FE 7D 55   U2äŠV.Í.]ëž.>þ}U  ..V.....
00000000C0  AA 75 6E FF 76 00 E8 8D   00 75 17 FA B0 D1 E6 64   ªunÿv.è..u.ú°Ñæd  ..v.....
00000000D0  E8 83 00 B0 DF E6 60 E8   7C 00 B0 FF E6 64 E8 75   èƒ.°ßæ`è|.°ÿædèu  ....|...
00000000E0  00 FB B8 00 BB CD 1A 66   23 C0 75 3B 66 81 FB 54   .û¸.»Í.f#Àu;f.ûT  ........
00000000F0  43 50 41 75 32 81 F9 02   01 72 2C 66 68 07 BB 00   CPAu2.ù..r,fh.».  ........
0000000100  00 66 68 00 02 00 00 66   68 08 00 00 00 66 53 66   .fh....fh....fSf  .h......
0000000110  53 66 55 66 68 00 00 00   00 66 68 00 7C 00 00 66   SfUfh....fh.|..f  ..h..h|.
0000000120  61 68 00 00 07 CD 1A 5A   32 F6 EA 00 7C 00 00 CD   ah...Í.Z2öê.|..Í  .....ê|.
0000000130  18 A0 B7 07 EB 08 A0 B6   07 EB 03 A0 B5 07 32 E4   . ·.ë. ¶.ë. µ.2ä  ........
0000000140  05 00 07 8B F0 AC 3C 00   74 09 BB 07 00 B4 0E CD   ...‹ð¬<.t.»..´.Í  ...<....
0000000150  10 EB F2 F4 EB FD 2B C9   E4 64 EB 00 24 02 E0 F8   .ëòôëý+ÉädëÞ$.àø  ........
0000000160  24 02 C3 49 6E 76 61 6C   69 64 20 70 61 72 74 69   $.ÃInvalid parti  ........
0000000170  74 69 6F 6E 20 74 61 62   6C 65 00 45 72 72 6F 72   tion table.Error  ........
0000000180  20 6C 6F 61 64 69 6E 67   20 6F 70 65 72 61 74 69    loading operati  ........
0000000190  6E 67 20 73 79 73 74 65   6D 00 4D 69 73 73 69 6E   ng system.Missin  ....m...
00000001A0  67 20 6F 70 65 72 61 74   69 6E 67 20 73 79 73 74   g operating syst  ........
00000001B0  65 6D 00 00 00 63 7B 9A   E7 8F 7E D9 00 00 80 20   em...c{šç.~Ù..€   ........
00000001C0  21 00 07 DF 13 0C 00 08   00 00 00 20 03 00 00 A3   !..ß....... ...£  !.......
00000001D0  14 0D 07 EF FF FF 00 28   03 00 00 10 F6 0D 00 00   ...ïÿÿ.(....ö...  ........
00000001E0  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................  ........
00000001F0  00 00 00 00 00 00 00 00   00 00 00 00 00 00 55 AA   ..............Uª  ........
0000000200  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................  ........

Sector: 0 (0x0)   Offset: 0 (0x0)   Read Only
```

Using the disk editor, we were instructed to find a JPG by identifying its header and footer. We would then take the first byte and move to the last byte, copying the data from the disk to its own file, assigning it a .jpg extension. Clicking the file we just created, we immediately saw the product of our labors—a picture of a cat. It was amazing, carving an image from a blob of hexadecimal data, to produce an actual file! It was not until the last day of the course that we were allowed to use automated tools again to carve the data from the floppy disk. What!? Automation? Why was this tool not given to us the first day of the course? This would have saved a tremendous amount of time and shortened the five-day course into two. I was beginning to think that the instructor was a masochist and was questioning his intentions, but then the epiphany came.

The countless hours of explaining the architecture of the media, the structure of the file system, and formation of files finally made sense to me. We, in simple terms, were the tool—first finding the header of the file and carving from the header to the footer—just as an automated tool was doing. So when we are requested during testimony to qualify what *carving* is and how a tool can find and extract the forensic file data, we fully comprehend and can explain how this process takes place within a software solution.

The foundation of a mobile forensic exam should be a sound understanding of the operation and output of any automated tool used. Organizations and academia have now changed the “computer forensic” label to include digital forensics in an effort to cover small-scale devices such as mobile phones and tablets. As discussed, there are inherent differences in the way information is obtained from a mobile device, but the examination of the collected data should be complementary.

NIST and Mobile Forensics

In its executive summary on mobile forensics (NIST Special Publication 800-101 Revision 1, May 2014: "Guidelines on Mobile Device Forensics"), NIST clearly explains that the digital forensic community faces the biggest challenges when it comes to mobile devices and investigations. Mobile devices are constantly changing to improve technologies, and investigative techniques must evolve with each new introduction. As stated, the key to understanding and answering the questions of today's investigator is having a firm understanding of the mobile device's software and hardware. A clear understanding of the limitations and the functions of forensic tools is important, but a clear and documented process and procedure are paramount.

Process and Procedure

Every job or discipline requires a set of processes and procedures that clearly define what is expected of the work and that aid in the workflow and ultimately a successful outcome. Procedures in forensics should never comprise a mere list of items to accomplish, starting at A and ending at Z. Adhering to a list and simply missing a step between A and Z would mean the entire process would fail. In forensics, and in technology investigations in general, the process-and-procedure document should be used only as a guide. There will be many different variables to consider, ranging from the actual physical device, to the type of software used, to the type of investigation the mobile device evidence will support. Processes used for one device might not be relevant to another or to a particular type of examination. What is important is that a clear set of procedures for conducting a mobile forensic collection and examination be established prior to the first examination to be used as a guide.

Even before a set of processes and procedures can be created, however, you must consider several objections to the notion of a process in conducting mobile device collections and thorough examinations. Following are two of the more prominent objections.

Lack of Time

To complicate the process further, examiners began to incorporate a four-letter word in forensics: T-I-M-E. Time is the excuse that is typically used in the mobile forensic community when it comes to undergoing a full examination of the extracted data from a mobile device. "We do not have enough time in our job to look at every application or file system file." The person conducting the collection has to move on to another case, another device, or another responsibility. But conducting the examination with this attitude is a critical deficiency in mobile device forensics, especially in the age of the smart phone, and can prove to be detrimental and costly in the long run. A hasty examination of digital data is like reading the first and last chapters of a book and then trying to write a review. The overall premise and how it might fit together could be garnered from the pages read, but without reading the entire book, the opinion is based upon 20 percent of the overall information contained therein. The examination is without substance. Although mission-critical data is often needed in the shortest time possible, the examiner should never ignore an opportunity to examine every byte, if given the chance.

The Importance of Time and Training

Back in the day when the first floppy drives were seized at an electronic crime scene, they were stored to be analyzed later because they had something to do with a computer's data—and the same goes for a laptop, PC, or any other storage/processing device found at the scene. It was understood, and still is, that these devices may contain data relevant to the case. First responders were not trained in the examination of the data, so they collected and held the devices until the computer forensic examiner could conduct a proper investigation. Floppy drives held a maximum of 1.44MB of data, and the devices of the day could have contained only a 528MB hard drive for storage. Today's mobile device on-board storage could be 16, 32, 64, and 128GB of data. They can be even extended to add 400GB of storage via removable media cards! (To put this into perspective on sheer storage capacity, a single gigabyte can hold data from 711, 1.44MB floppy disks and the data from almost two of the 528MB hard drives.) First responders, patrol officers, narcotics officers, and school resource officers are collecting and holding mobile devices every day because they need actionable intelligence quickly, but they still consider a mobile device just a phone, not a computer.

Collections by first responders should be supported, however, as long as the collections adhere to a predefined and vetted process. The examination of the data must be conducted by a person who has both the *time* and the *training* to peel the skin from the onion and peer into the many subcutaneous layers.

Simplicity of the Tool Equates to No Training Needed

Mobile forensic software and hardware tools have been designed and marketed to express to the purchaser that little to no training is needed to conduct a mobile forensic examination. Simply push this button and out comes the evidence; no need to add training or comprehend what is actually occurring when the button is pushed. The inverse is true, however, because the simpler the tool, the more training the examiner will need in order to testify about what is really occurring. Did the software on the device, once the button was pushed, query a database to retrieve the contacts and SMS? If that is the case, what database did it query? If the examiner can answer these questions, then using an automated tool for mobile device collections will stand up to scrutiny.

Standard Operating Procedure Document

What should be a part of the mobile forensic process? Much like the computer forensic process, it must start with a written document outlining the mission—a standard operating procedure (SOP) document. An SOP covers not only the person or persons conducting the examination of data but also those who are collecting and seizing the device or devices holding the digital data. The SOP should outline the process and procedures to be followed, from the seizure to the reporting of the data.

When you're starting a forensic lab or beginning to examine mobile devices, creating this document should be your first step. SOP processes and procedures for mobile forensics are much like those for computer forensics. The Scientific Working Group on Digital Evidence (SWGDE) maintains that a proper SOP (such as its template, version 3.1) will assist you in maintaining the best practices for collecting, acquiring, analyzing, and documenting data in digital forensic examinations. Using the outline, suggestions, and information in the following sections, you can develop an SOP that will be specific to the enterprise, agency, or company.

Purpose and Scope

A mobile device forensic SOP should outline the purpose and scope of each possible location at which a mobile forensic device or collection could occur. The *purpose* explains why the section or SOP is being used and identifies the goal of the SOP or section, and it should be detailed enough so that the reader will recognize what the document or section will cover. Several purpose statements can exist in a single SOP, with only one purpose statement per SOP section. The *scope* identifies who will be following this procedure and what the procedure covers. The scope can also explain what will be needed or covered in the SOP or the section. The scope should be very clear and identify areas that are and are not going to be covered or that are outside the scope of the document. The writer of the SOP should not assume that the reader will understand that something is or is not part of the document or procedure; this must be implicitly stated. Here are examples of both purpose and scope for a mobile forensic off-site request.

> *Purpose:* The purpose of this procedure is to seize, secure, and collect digital data from a mobile device at an off-site location to maintain the integrity of the device and contents for further analysis and processing.
>
> *Scope:* This SOP outlines the process and procedures to follow when you are conducting mobile forensic assistance at an off-site location. This SOP is not a training document, but a set of procedures to follow at an off-site location.

After listing the purpose and scope, the SOP will continue with the actual equipment, specific knowledge data, and procedural items. Recommended areas to cover include definitions, materials and equipment, general information, procedures, and references and related documents.

Definitions

The definitions paragraph should list and define all acronyms and technical words included in the procedural part of the SOP. When creating an SOP, the writer must understand that the reader might not understand the technology—for example, assuming the reader knows what WCDMA or UICC means could cause on-site problems if these are unfamiliar terms. Defining any term that is related to the technology is recommended. Here is a sample definitions paragraph that could be used for an on-site or lab SOP.

> *Definitions:*
>
> **Mobile device:** Portable devices that use network communication (cellular or Wi-Fi) and have digital storage capabilities. Examples can include a cellular phone, tablet, or wearable.
> **Mobile device external media:** Digital storage media. Examples include microSD cards and SD cards.
> **Mobile device internal media:** Digital storage media that is part of the actual mobile device, typically soldered to the internal components of the device and not removable.
> **Cloud mobile device data:** Digital data that is stored on remote servers and accessed by the mobile device transactionally.

Equipment/Materials

The equipment statement should include all the items that are needed to accomplish the listed procedure successfully. This should cover every contingency and should be broad enough so that the reader will not have to return to pick up additional items not listed in the SOP. An example of an equipment statement is listed next.

Equipment: The equipment that will be needed includes the following items:

- Digital camera
- Sterilized portable USB hard drive
- Media card write-blocking tools
- Radiofrequency (RF) shielding device
- Mobile device collection tools
- Mobile device cables and SIM card readers
- Evidence packaging materials

General Information

Include general information to define limitations or background information regarding performing the duties outlined in the SOP. Important limitations should be explained directly and clearly to the reader. In a mobile forensic examination, limitations could include information regarding on-site and off-site seizures and collections. This paragraph should set the stage for any contingency the reader may encounter. This area of the SOP can often continue to grow because of constant change in both mobile device and software technology. An example of a section outlining limitations on an off-site seizure of mobile device data is shown here.

General Information – Limitations:

- If a mobile device has network access, data destruction can occur.
- If a mobile device is shut down or loses power, it may lock, essentially eliminating further access.
- If a mobile device is locked upon seizure, further access might not be possible unless a passcode is obtained from the owner or computer with which the device was last synched.
- Some software tools do not collect all of the data on a device.

Procedure

In the procedure portion of the SOP, the reader is walked through the performance of the task. This should include enough detail to enable the reader to perform the duty, but it should not be so detailed that the reader feels compelled to perform each step for success. An SOP is not the exact process that must be used in every instance, but a guide as to the best practices to conduct a mobile forensic examination from start to finish. For example, the following procedure is too detailed to be used in every case:

Place the device into airplane mode by navigating to Settings > Tools > Network > Airplane mode.

Clearly, this would not work in every instance and would cause confusion. Simply stating, "Place the device into Airplane mode," would suffice. An example of a procedure you could use for an off-site response to a mobile forensic scene is shown next.

General:

- The scene should be secure and safe for all individuals.
- Protect devices and the evidence contained on the devices.
- Identify the areas of the scene to be searched.
- Photograph the area and each potential item of evidentiary value.

Mobile Devices:

- Photograph the device and any data on the screen.
- Block the mobile device from receiving RF signals either by placing the device in airplane mode or using an RF-shielding device.
- If the device cannot be shielded from RF, the device should be turned off.
- The device should be packaged and submitted for processing as soon as possible.

References/Documents

The final category included in an SOP will be references or related documents. The reference area should include other SOPs that are related to the current SOP, such as the SOP for the "Collection of Mobile Device Data On-Site" included within the SOP for the "Seizure of Mobile Device Evidence On-Site." This enables the reader to transition immediately to the SOP to learn how to handle an actual collection of the data if needed. Also, any documents that might aid the reader in performing a task should be included, such as published documents, web sites, or manuals.

Successful SOP Creation and Execution

For mobile forensic SOPs to be successful, they must be categorized in sections or modules according to specific operations. The following sections discuss the recommended modules and content that should be contained within a mobile forensics SOP document. You can add more specifics to these modules after establishing a clear understanding of proper procedures for conducting the seizure, image creation, and analysis. These processes are covered in depth in later chapters.

Mobile Forensic Seizure On-Site Procedures

This module should cover the procedures users will take when preparing for and arriving to a site or scene where mobile devices and evidence relating to a mobile device will be encountered. This module should cover equipment needed, scene safety, identification of a mobile device, SIM cards, external storage media (SD cards and microSD cards), USB cables, manuals, location of passwords or PIN numbers for a mobile device, and devices that may hold backups of mobile devices such as a computer or laptop. The proper methods to inhibit a mobile device's network connections, what to photograph, how to package a mobile device upon removal from the site, and the transportation of mobile device evidence should also be included in this module.

Mobile Forensic Image Collection On-Site Procedures

This module should cover the necessary procedures if a mobile device image is to be collected using mobile forensic tools while the user is on the site or scene. Items covered should include, but are not limited to, equipment needed to create a forensic image of a mobile device, SIM card, or removable media, and procedures for proper isolation of a mobile device from network connections. Proper procedures for the recovery of a mobile device image should include determining the type of mobile device and its functions, determining the correct software for the type of device, and conducting an extraction and subsequently verifying that the data extracted can be manually located on the device.

Mobile Forensic Image Collection at Lab Procedures

This module should cover the procedures users should take for forensically processing and analyzing a mobile device in a lab setting. Included should be the steps required to complete the isolation of a mobile device, depending upon the state in which the device was received, with specifications for how device information should be obtained and how to gain a thorough understanding on the device capabilities prior to beginning the extraction. It should include guidance on what to do if the device supports a SIM card and what software will be needed to conduct a forensic examination of the SIM card. If a media card is also located, the appropriate software should be used to create a forensic image of the media card and should be described here. Any software that will be needed to complete an extraction of the mobile device should be identified. Once the correct software for extraction has been determined, the process should be determined by the state of the device. If the device is powered off, this procedure will instruct the lab examiners to begin with data from the SIM and media cards and conclude with the mobile device. If the device is powered on, the reverse process should occur: handset, media card, and then SIM. This information is covered extensively in the processing section.

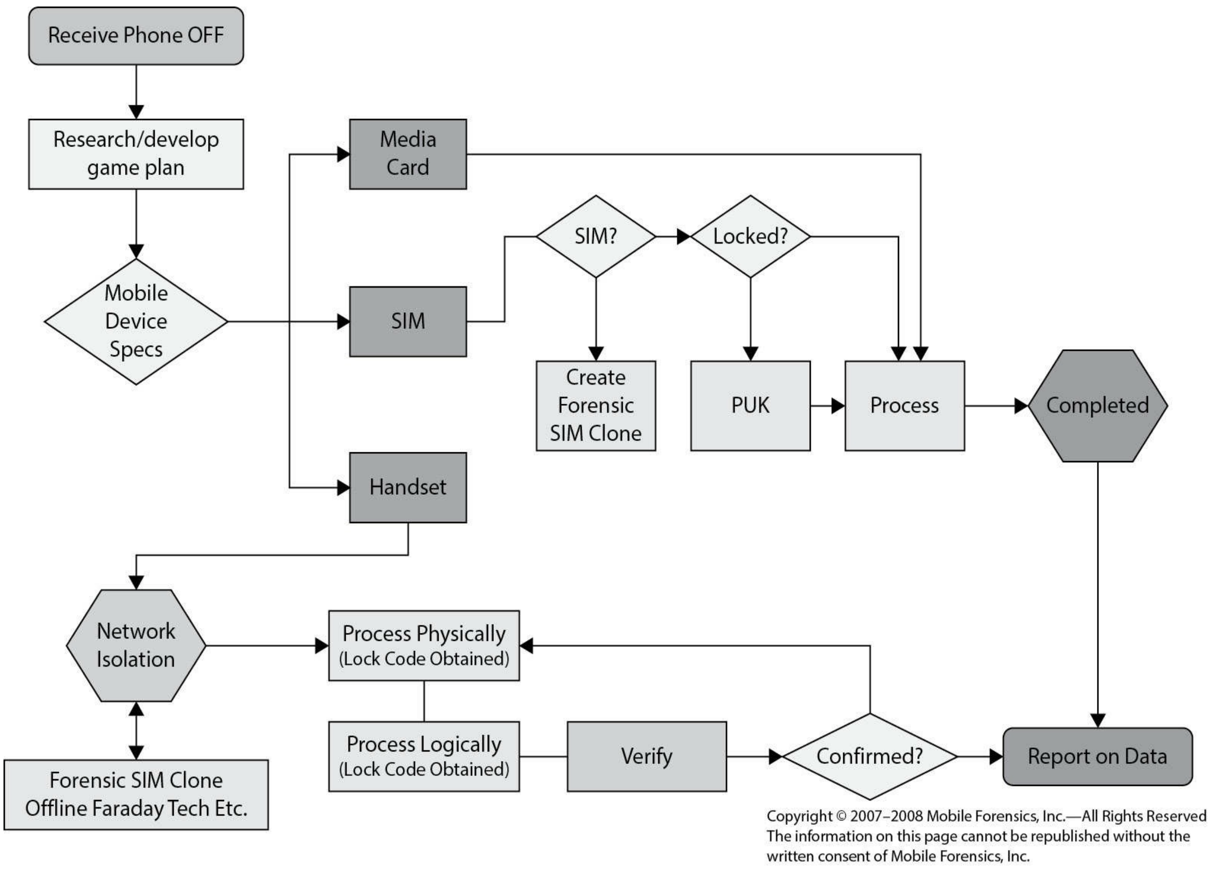
Mobile Forensic Image Collection from Cloud Procedures

This module should cover the procedures users should take for obtaining and analyzing the data from a mobile device that is transactionally stored within a remote server (Cloud). Included should be the steps required to obtain the necessary credentials, which could include a username/password combination or security token. It should reference how to obtain the information legally from the Cloud and should include why it is necessary to obtain the data remotely. Furthermore, the data extracted from a Cloud extraction should be compared, if possible, to the data from the mobile device extraction. This way, a determination can be made on the redundancy of the data, as well as unique properties from the off-device storage medium.

Creation of a Workflow

The creation of a workflow or flowchart can be of great benefit both to the first responder and forensic examiner to guide them from the seizure to the examination of a mobile device. In 2007, MFI produced a flowchart to assist in processing mobile devices when they come into the lab. During Mobile Forensics World

2008, this flowchart was introduced to students in the MFI Crash Course to Mobile Forensics class that was held at the Chicago Police Department's training facility in downtown Chicago. The workflow shown in Figure 2-2 was outlined in 2008 and is still applicable with today's mobile devices. In chapters to come, additional steps will be added to the flowchart to help expand on different scenarios.



Receive Phone OFF
Research/develop game plan
Mobile Device Specs
Media Card
SIM
Handset
SIM?
Locked?
Create Forensic SIM Clone
PUK
Process
Completed
Network Isolation
Forensic SIM Clone Offline Faraday Tech Etc.
Process Physically (Lock Code Obtained)
Process Logically (Lock Code Obtained)
Verify
Confirmed?
Report on Data
Copyright © 2007–2008 Mobile Forensics, Inc.—All Rights Reserved The information on this page cannot be republished without the written consent of Mobile Forensics, Inc.

FIGURE 2-2 **Processing a mobile device**

Specialty Mobile Forensic Units

Mobile device forensics can be much more involved than traditional computer forensics. Successful mobile device forensics can be a time-consuming process. Because of the many applications, files, logs, and constantly changing technologies involved, a special unit or a specially trained individual should be in charge of conducting mobile device investigations. With the proliferation of mobile devices and their involvement in every aspect of today's world, the challenge of finding someone who can juggle multiple disciplines could encumber the ability to conduct a thorough examination.

Mobile Device Examination Priority

When I began working in forensics, I started with a foundation in computer forensics, with proper training in both theory and application. Computers would arrive in the lab and would be categorized and examined based upon procedures that had been established previously. I then moved into the examination of mobile devices, and my world changed—not because I really enjoyed processing mobile devices, but because the onslaught of mobile devices was like a broken faucet, releasing 100 gallons per second, as devices began arriving from all over the state. With both computers and 20 to 25 mobile devices per week to examine, I found it almost impossible to juggle both disciplines. To examine both computers and mobile devices effectively and according to the required standards, something would have to give, or the quality of the investigations would degrade. As the number of devices continued to increase, the variety of software continued to change, and the time involved in a proper examination increased, I decided to make mobile devices my priority and focus.

Forensic Software

The U.S. Computer Emergency Readiness Team (US-CERT) defines computer forensics as "the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law." So mobile forensic software, to somewhat paraphrase US-CERT's definition, would be software that is used to collect and analyze data from a mobile device in a way that is admissible as evidence in a court of law.

A software application is a generic set of instructions for a computer. Software used by practitioners is defined by two classes: system software and operational or application software. System software is used by the computer system itself and does not involve the user; it is designed to accomplish specific tasks that keep the computer or system running as expected—such as writing of data to a disk drive or displaying a graphic on a computer screen. Application software facilitates tasks the user needs to perform his or her work—such as word processing, image creation, and mobile forensic examinations. All mobile forensics tools are application software. They are not magical things that conduct massive investigation feats with the click of a mouse or that autonomously process a device and output a report. You must understand your application software and interact with it in a way that is admissible in a court of law and to produce a product that can be used to substantiate the entire investigation.

Many different types of mobile forensic software can be used for today's examinations. Mobile forensic software typically differs in functionality and complexity, but it also differs as to the types of devices that are supported. With the many different types of mobile devices that vary not only by manufacturer but also by communication protocols in use, no single mobile forensic software tool supports all mobile devices. What it really comes down to is the type of examination, the goal of the examination, and how the produced evidence might be later used.

Common Misconceptions

With mobile forensics and examiners, many misconceptions must be dealt with—not only when explaining the produced work to the requestor but also when answering questions during testimony. Being prepared to combat common misconceptions and head them off before they proliferate will be important to the ultimate success of the examiner.

Seasoned Computer Forensics Examiners' Misconceptions

Examiners who have been conducting only computer forensic exams for many years make up the largest group that misunderstands the validity and value of the scientific examination of a mobile device. These examiners have undergone training that dictates that they must obtain a full duplicate copy of the digital media using a write blocker and ensure that the operating system obtaining the image did not perform any inadvertent writes to the evidence. This produces a copy of the evidence that they later analyze, and they must be able to conclude unequivocally that the data was not tampered with during the collection and analysis.

On mobile devices, writes do occur, simply because the devices are powered on during extraction, with the simple change of the clock or other internal processes. Concurrently, the extraction takes place without a write blocker, and because of this, a mobile device is not viewed as a mass storage device. Also, because a mobile device cannot be imaged as a hard drive, computer forensic examiners have relegated mobile device forensics as merely mobile device extraction.

To combat this misconception, the mobile device examiner must recognize that although data does change on the device during the investigation, user data should never change or be altered by software. This can be shown by locating the actual files from which the data was collected and by verifying the software's results. You must understand that the protocols to communicate with a mobile device limit the collection of the device's internal memory, and that proprietary hardware and software limitations imposed by the hardware manufacturers make it difficult to capture a full image. What makes the examination forensic, however, is not the software, but the process the examiner takes during the extraction and subsequent analysis of the data.

First Responders' Misconceptions

Unfortunately, the majority of mobile device seizures and extractions are conducted by individuals who have not received formal training on forensic processes, but who have received tool-specific training. Most extractions take place at the scene or on-site. Because the majority of the training involves how to conduct the collection of the mobile device, any deviation from the training course material typically means that the mobile device forensic tool will fail. If first responders do not receive training on the software tool, the percentage of failures of that particular device grows dramatically. This does not just have to do with a software tool failing to connect to a particular device, but with the data that is collected. A common misconception is that the software tool did not work if the data is not immediately presented to the user. This is far from the truth, however. Since the majority of today's tools use database files to store data and use common file types, would it not behoove the user to analyze the data rather than the output? This is the conundrum of today's mobile device examiner.

Sometimes Data Is Not Where You Expect

I was assisting an officer with an investigation using a commercially available mobile forensic solution. The officer was extremely frustrated because he had obtained an image of an Apple iPhone and was unable to locate a specific text message from the device's image. Because the device had been returned to the owner, the officer was upset that this evidence was now gone. I asked to see the created image and asked about the case.

The case was simple: a text message was sent to the person requesting some information about a package. The officer pointed to the report and the interface showing the date and the time when it should have been listed and said, "It is not there." I was shown the initial report outlining the SMS and

MMS data and also was told it was not there. He complained, “The tool doesn’t work!”

I immediately asked if he had looked into the device file system to determine whether the message was sent from a third-party application. He said that he looked only at the applications that had been automatically parsed by the tool and did not see the message. I explained to him that many applications are released every day, and it is possible that the application had not auto-parsed.

We navigated into the file system and found the Facebook Messenger app, which at the time was not supported by the particular solution. Inside this application, we located the common SQLite database and immediately located the data he had been looking for. The case was solved—not by a magical application, but by following a process and understanding the underlying file system and file types that made the difference. This was not a failure of the software, but a lack of process.

Chapter Summary

The principles of examination are no different between the disciplines of computer forensics and mobile device forensics. To be successful in the discipline of mobile device forensics, the examiner must understand that the process from seizure to analysis is no different from examining anything containing 0’s or 1’s—in other words, the same process used for a computer forensic examination should be followed and conformed to in mobile device forensic examination.

The user must seize the device legally, collect the device, and conform to common evidence best practices to extract the best evidence image, verify the output and image, and conduct a thorough analysis of the collected data. That information then has to be presented in a way that conforms to industry standards.

IACIS, ISFCE, and SWGDE contend that processes and procedures must be adhered to during the complete evolution of a digital device examination. Furthermore, processes and procedures should be part of a set of SOPs for all facets of the mobile forensic life cycle and location. These SOPs can be successful only if the entire company, department, or agency fully supports them, from top to bottom. With successful implementation of SOPs, along with complementary training, success can be guaranteed.

The many misconceptions of users—from seasoned examiners to first responders—can be circumvented with training and knowledge regarding not only the small differences in mobile device forensics but also regarding the similarities of successfully gathering and analyzing any digital data.

Now you understand the similarities of mobile forensics to computer forensics and the importance of the SOP for mobile forensics. The next chapter will explain how to begin a mobile forensic examination correctly by properly seizing evidence from the scene.

3 New Era of Digital Devices: IoT, Infotainment, Wearables, and Drones

Our world is wired. We were inundated with cell phones, then smart phones, and now IoT (Internet of Things) devices, infotainment devices, wearables, and drones. This chapter deals with a small set of device categories within the Internet of Things. Although infotainment devices and wearables could be categorized as IoT devices, because of their consumer functions and popularity, I'll discuss them as separate entities here. Many, including drone manufacturers, have argued that drones, or technically unmanned aircraft systems (UASs), could also be lumped into the IoT category.

Today's examiner faces a huge challenge with numerous devices more complicated than the smart phone. Just when mobile phone examiners were making progress in the "Wild West" methodologies, IoT devices, in-vehicle systems, wearables, and drones rode into town. Manufacturers of smart phones did not lay out their file system formats, and even today there are only a few published books that intelligently cover processing these devices. A limited number of articles have been written and research conducted regarding the devices discussed in this chapter, and they have been done completely by research and development groups, without assistance or documentation from any manufacture resource.

With these devices, it seems forensic examiners have returned to the Wild, Wild West. However, this time, not only must the examiner work with more types of devices, but he or she faces many more obstacles in the pursuit of forensic data—from finding the storage location, to gaining access to the controller, to "speaking" to the device, to extracting the data, and, most importantly, to understanding the data format and layout.

Note

These hurdles are similar to those I faced when I started in the mobile forensic field, but I believe the problems associated with this current field of devices surpasses those faced by mobile forensic pioneers in the early days.

Research into these types of devices, their investigation, and their analysis should be a priority. Delving and probing into the functionality of these devices and their relationship to investigations should drive this prioritization. By *prioritization*, I'm referring to increased understanding *how* and *if* a connected device or devices are or can be involved in the case or investigation that is underway.

As discussed in previous chapters, the methods we use to conduct a proper search for digital devices in a crime scene have changed with today's connected devices. In the triage of the crime scene, the investigator must be cognizant of connected devices. With so many electronic possibilities—from refrigerators, doors, windows, lights, thermostats, and televisions, to personal assistance devices (such as Alexa and Google Home), wearables (such as Fitbit or Apple Watch), and other devices—one thing is clear: nothing in digital forensics is absolute or routine. Today's investigator must be familiar with a radically confusing "electronic scene." Moreover, it is ever important that the investigator triage based upon various conditions, including involvement in the case, probability of extraction, and the likelihood of parsing/decoding the data. A simple example could be determining whether or not to seize a refrigerator. If the fridge is simply an "intelligent" refrigerator, or one that uses sensors to monitor the device and/or contents without network connectivity but the investigation involves a cyber crime, a seizure would not make sense. However, if the fridge was a truly connected device (an IoT device), seizing it as evidence could make sense if details from other sources such as a router or gateway pointed to this device. Or consider a wearable device found at the scene of a violent crime or a drone found at a residence known for drug trafficking. Today's crime scene triage has become a daunting task with the widespread use of these types of devices, not only for the fact of identifying them but also for processing and exploring the data they contain.

This chapter introduces the various connected devices, provides examples of related devices, discusses how they fit in the digital forensic investigation cycle, and shows how the investigation cycle might affect the approach of the examiner if, and when, such a device comes into the forensic lab.

IoT Devices

An IoT device is an electronic device that has been embedded with technology to be controlled and to interact over the Internet. *Forbes* magazine, back in 2014, said that anything that has an on and off switch to connect to the Internet and/or to another device can be considered an IoT device. Much like mobile devices, the number of IoT devices has exceeded the world's population—and the number is increasing at a much faster pace. In 2017, Gartner stated that there were currently 8.4 billion IoT devices in use globally; it also reported that by 2020, the IoT set of devices that are available throughout the world will reach 20.4 billion.

Figure 3-1 shows, in simple terms, what an IoT device is and what makes this technology quite useful for today's connected world. The implications of the device's digital footprint are limitless. Requests are made from a remote or local device through the Internet, and data is returned to the device, often with asynchronous communication between connected devices and/or remotely controlled devices. With both inbound and outbound communications, the exposure of the device to the outside world is a given, and with few security precautions available in many devices, cases involving unauthorized access will skyrocket.

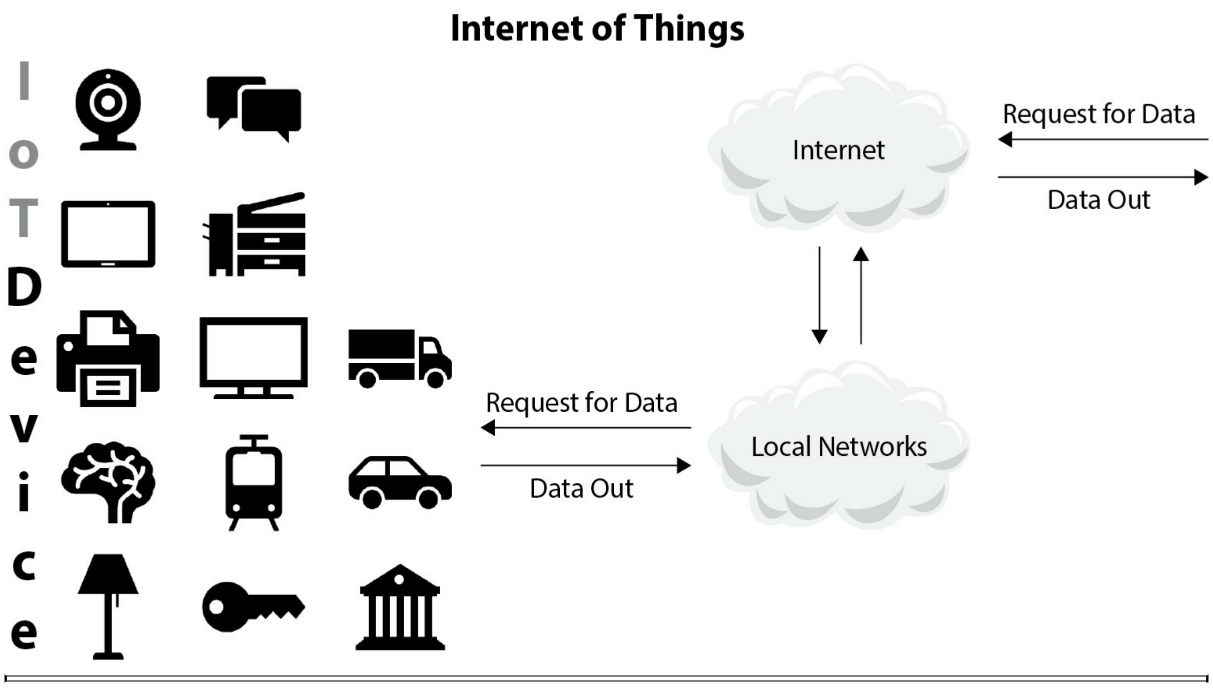


FIGURE 3-1 **Simple communication of various devices through an Internet gateway and back**

Note

Visit the Shodan web site to see just how many IP webcams and security cameras around the world have been exploited and compromised.

A noticeable change to the device architecture for an IoT device from that of a mobile device is the underlying operating system (OS) in most connected devices. In today's mobile device operating systems for both Androids and iOS, and even in the PC OS for Windows, there are limitations regarding the small embedded devices in the IoT world. One big limitation is simply the size, or footprint, of the traditional OS. Another issue with traditional OS and connected embedded systems is speed or timing of computations. Embedded devices within a vehicle, such as an air bag or automatic braking, that report back to the onboard computer and then to the manufacturer, or even data monitoring companies (such as OnStar), rely on timing from action to results. If a crash is detected, the safety mechanisms must respond without delay. This is where a traditional OS would be a detriment and a real-time OS is required. Enter the Real Time Operating System (RTOS) format.

RTOS uses absolute time iterations to complete the processing of an event. RTOS is not new: it is used in the firmware of Android baseband processors and feature phones, both old and new. Samsung's proprietary operating system is based on RTOS, as are devices from LG. (These feature phones are covered in Chapter 12, as are the file systems that are produced.) With today's connected devices and their embedded systems, it makes sense for manufacturers to utilize an RTOS because of its small footprint, low power requirement, and lack of latency—and in the case of medical and safety equipment, RTOS is a necessity.

Categories of Connected Devices

Connected devices can be placed into several categories based primarily on their functionality and use—

primarily consumer, enterprise, and industrial devices. Most devices encountered by investigators will be consumer-based devices, but in certain circumstances other types can be involved.

Consumer Devices

Consumer connected devices include stereos, televisions, appliances, thermostats, windows, doors, garage doors, and lights—to name a few. Most consumer IoT devices generally have the added word "smart" to the product name (smart TV, smart refrigerator, or smart thermostat, for example). With the availability of consumer devices increasing around the world, from high-end electronic stores to the local big box store, the likelihood an investigator will deal with "smart" devices grows every day. Consumer devices, much like all other connected devices, come with an operating system, which could be a simple Android system, or RTOS, Linux, Tizen, and many other variants. The OS in use is highly dependent on the size of the connected device—for example, the OS on a wearable is drastically different from that of a smart TV.

Table 3-1 lists some of the operating systems found in smart TVs that could help direct an examination of an embedded file system. Many of the operating systems listed are also found in appliances ranging from refrigerators to interactive thermostats with full-featured access to the Internet, as well as third-party applications.

TABLE 3-1 **Common Smart TVs and Their Underlying Operating Systems**

Devices and Manufacturer	Operating System (Derivative)
Amazon Fire TV and Fire TV Stick, Westinghouse	Fire OS (Android)
Apple TV boxes	tvOS (iOS), Apple TV (iOS)
Google Chromecast, Google Home	Chromecast (morph of Google TV and Android)
Smart TV for Hitachi, Insignia, Sharp, LG, JVC	Roku TV (Linux)
Smart TV Panasonic	Firefox OS (discontinued 2017)
Smart TV Philips, RA, Sharp, Sony	Android TV (Android)
Smart TV Asus, LG, Sony, VIZIO	Google TV (deprecated 2013)
Smart TV Samsung	Tizen (Android)
Smart TV LG	WebOS (Linux)

Note

Various file system examples will be covered in Chapter 16 along with a discussion of worthwhile artifacts within the systems of these devices.

Just because a device may have an OS does not necessarily make it a part of the IoT family. As mentioned, and clearly referenced in the acronym, the IoT device must be connected to the Internet. Consumer devices are generally connected via Wi-Fi, Bluetooth, and Ethernet. Most of these devices, for sheer portability and range, use Wi-Fi to transmit and receive data and to communicate with other IoT devices; many use ZigBee or Z-Wave as well. Connection information could be good artifacts to harvest in an investigation simply because the device can leave identifying IDs such as media access control (MAC) addresses. The MAC address is a unique electronic fingerprint that is assigned and embedded on the device when it's manufactured. Using this unique hardware number can lead the investigator to the connected device involved in a particular event. A connected IoT garage door or thermostat will have a unique address if it is used to communicate via a gateway to other connected devices or to the Internet. By understanding the gateway and control, you can start your investigations involving IoT devices that have been compromised from a more informed position.

With access to the Internet, a smart TV becomes an all-in-one computer, whether the user is browsing the Internet via the built-in web browser, using Facebook, viewing YouTube, talking on Skype, or even reading messages sent from a mobile device. All of this information can be available on the connected smart device. From Wi-Fi cameras to doorbells with video, data evidence of an event may be held within theses devices or their storage containers. The examiner must be prepared to dig deep into their file systems, because commercial tools do not currently support these types of device images and artifacts.

Enterprise Devices

Enterprise IoT devices offer connectivity from one device to another within the general enterprise sector. Enterprise devices include VoIP telephones, videoconferencing equipment, IP-connected security cameras (CCTV), smart HVAC systems, smart printers, and many other connected devices. With the proliferation of devices connected at the enterprise, in addition to mobile devices, it is no surprise that hacking attempts are at the forefront of business infiltration.

As with devices in consumer and industrial sectors, the security of an enterprise device is not the top priority of most manufacturers. In 2016, an ethical hacker reported in darkreading.com about the ease of hacking into such a device in the article "Common Enterprise IoT Devices Are Hackable in Under Three Minutes" (www.darkreading.com/iot/common-enterprise-iot-devices-are-hackable-in-under-three-minutes/d/d-id/1327300). This article may be several years old, but today's IoT devices are still vulnerable to attack with little difficulty due to the proliferation of devices within an enterprise. Enterprises can face security issues if IoT devices are unchecked for installed security measures. These days, the weakest links in the enterprise are consumer wearables (watches, sound devices, video devices, smart TV, and so on) that connect to the network. Consumer-grade IoT devices often have lower standards for security, and as a result, the user of the consumer wearable often inadvertently exposes the enterprise network.

From an infiltrated network creating zombies of IoT devices, to hackers unleashing attacks on external targets or gaining access via a device to spy on, listen to, or destroy a network, today's attack surfaces are immense with the introduction of IoT devices. Currently, network infection can occur in under three minutes, with a dwell time (compromise to discovery) of 101 days on average, according to FireEye's report "M-Trends 2018" (www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf). This problem will likely increase around the globe with the proliferation of IoT devices. Today's response plan and investigator must be educated to understand these enterprise-connected devices.

Industrial Devices

Industrial IoT devices, as a general definition, offer connectivity from a device to another device within the transport, energy, and industrial sectors. Within the transport sector, both machinery and vehicles can offer connected solutions, while the energy sector includes oil, gas, and water machinery as well as the power grid. Industrial solutions can include machinery interconnected to larger machinery and monitoring equipment. These examples do not represent the entire application for industrial IoT but easily paint a picture of the possible investigation implications.

The connection, or rather infiltration or damage, to any of these sectors via compromised IoT devices can have catastrophic effects on not only local populations but also globally. This possibility is why many of the industrial applications claim to be “air-gapped.” An *air-gapped* device or series of devices is not directly connected to an external network. The device essentially operates autonomously, without the possibility of external infiltration, in theory. Of course, penetration is possible—remember Stuxnet, the worm introduced to an air-gapped system using USB drives and new innovations such as LED data transmission interception? What does all this have to do with IoT devices? Actually a lot, because a large number of devices in air-gapped systems, especially those involved in coordinating maintenance needs, levels, and downed systems, must be connected to external sources.

A perfect example of this is the maintenance of airplanes. The United States Title 14 of the Code of Federal Regulations (14 CFR) part 91, § 91.1109 “Aircraft maintenance: Inspection program” and FAR 91.409 outline the requirements for mandatory inspections of aircraft. Within these requirements are sections that outline classes of inspection that state that airlines must maintain an “airworthiness” certificate. The minimum requirements are services every 12 months, 100 hours of service, and preflight checks. However, U.S. airlines utilize different guidelines on maintenance to indicate that an inspection must be made every 12 months (1–2 weeks out of service), every 6 months (1–3 days out of service), and 500 hours (10 hours out of service). Mentioned are the weeks, days, and hours out of service, which cost the airlines tens of thousands of dollars every day. In an effort to respond to, or rather predict, such issues, IoT devices are used to detect potential failure of parts before they actually fail. Sensors placed throughout the airplane machinery gather information, and the data is analyzed constantly to create the maintenance schedule, as well as order parts and the technician for the aircraft, all in an effort to predict the life cycle of each part. Are you ready to analyze tampering in infiltration of such a system?

Note

This book, of course, will not cover the triage of sensors or systems used in the aerospace industry, but you can be sure that experts in these areas are needed in the event of disasters or civil litigation.

Common Consumer Types

The investigator will likely run into consumer-based IoT devices more often than those within the enterprise or industrial sector. Two consumer-based devices are covered in this section: Amazon Alexa and Google Home. The information covered includes an introduction to the device, the device’s value to an investigation, what artifacts might be gleaned, and how well the device is supported by commercial or open source tool sets.

Chapter 16 will discuss details on file system artifacts, Cloud data, and advanced topics on both devices.

Amazon Alexa

On August 31, 2012, a patent was filed by Amazon for a device built as an artificial intelligence machine. Alexa was born just two years later, but not as a device: Alexa is the assistant that is used in the Echo family of devices (which includes Dot, Tap, Echo, Echo Plus, Spot, Look, and Show). Alexa will interact with built-in functions to play music, order items from Amazon, provide news and weather information, set timers and alarms, recite movie listings, update calendars, and more. Currently, Alexa can interact with other smart devices within the connected home (lights, thermostats, and so on) via Alexa Voice Service (AVS).

Alexa does not stop there, however. Amazon enables Alexa to use a crowd-sourcing mentality to build Alexa's "skills," the term used by Amazon to define what Alexa can do and how it can interact. Because there are thousands of other IoT devices, and even companies, that want to be able to be reached, Amazon allows these entities and devices to do so via Alexa. The Alexa Skills Kit is an API collection that facilitates the building of Alexa's vocabulary (skills), allowing the device to recognize thousands of additional terms to provide thousands of additional functions. From placing orders to Starbucks, to contacting Uber, to answering *Jeopardy* quizzes and question of the day, you can simply add skills and ask Alexa. The Amazon Echo Plus also began to utilize ZigBee to communicate with other IoT devices along with the standard wireless technologies, and now it can be considered an IoT hub as well.

Alexa needs user interaction to begin to use its skills base. This interaction is known as the "wake word." Alexa's original wake word is "Alexa," but with Amazon devices, the current wake words are Alexa, Echo, Amazon, and Computer. Most devices, however, enable you to change the wake word within the Alexa app. (The only exception to this is on the Tap. When a user utters a wake word, the device is set to listen to the next request. This request could be "find a recipe for tri-tip," "what is the score of the Washington Nationals game?," or even "what is the weather in Jamaica this week?")

The interesting thing about Alexa is the "learning" that goes on behind the scenes, on the Amazon servers. The learning side of these devices enables your requests to move from saying "Washington Nationals" to simply saying "Nationals" on subsequent requests. The recording will stay on the server, and a user can keep track of requests on his or her mobile device via the Alexa app. This information is stored in Settings | History. Here, the user can see what is on the server and remove the information one item at a time. If all items are to be purged, a user can use the app to clear the entire history all at once. The words stored on the Amazon servers are used for targeted advertisements and to learn as much about the user as possible.

So if a wake word is used to start using the tool, wouldn't that mean the device is always listening? The answer to this is yes; the device is always listening, but Amazon insists that nothing is recorded to the servers until the wake word is mentioned. In a patent filed by Amazon (US8798995B) and granted in 2014, and again submitted in 2017 under US20170323645A1, some language is included that should cause anyone to wonder whether the wake word really starts the recording process. Within both patents, Amazon specifically says otherwise and mentions actively listening to conversations. Here's an excerpt from the patent application:

> While phone conversations are described in many of the examples herein, it should be understood that there can be various situations in which voice data can be obtained. For example, a user might talk to a

> friend about purchasing a mountain bike within an audio capture distance of that user's home computer. If the user has authorized the home computer to listen for, and analyze, voice content from the user, the computer can obtain keywords from the conversation and automatically provide recommendations during the conversation, such as by displaying one or more Web sites offering mountain bikes, performing a search for mountain bikes, etc. If the user discusses an interest in a certain actor, the computer could upload that information to a recommendations service. ...

In April 2018, several news outlets, from BBC to Wired, suggested that Amazon had filed a patent for a "voice-sniffing" technology that would allow the device to listen to any conversation within its range and send that data to Amazon servers. This use, of course, was denied by Amazon, but the company did indicate that the patent was filed as a "forward-thinking" application. However, in May 2018, a user of the Echo product contacted Amazon as well as media outlets to report that a private conversation she and her husband had while in their house was, unbeknownst to them, sent to a contact in their phone book. Amazon admitted that the technical issue was a bug, and the company said it would fix the problem as soon as possible. But this is not the last time voice-sniffing will be in the news—or spoken about in the forensic lab.

Where Is Alexa Hiding the Data?

The devices that house Alexa are generally using approximately 256MB of RAM and 4GB of storage space. Currently, no commercially available tools can access the device via noninvasive means (such as cable or Bluetooth), nor are any tools available to process a raw file system that is acquired by invasive means (JTAG, ISP, chip-off), but access to the valuable data *is* achievable. If a raw file system is obtained, the underlying OS will be determined by the version of the hardware. Prior to the second-generation Echo Dot, the OS was generic Linux, which then switched to pure Android and continues to run Android but has transitioned to the forked version of Android, called Fire OS. Again, commercial tools are not currently extracting data directly from the device or supporting the direct parsing of an imported file system.

Alexa, however, stores data within the user's Amazon Cloud account and the Alexa app, as observed in research, as well as documented in Digital Investigation 22 (2017) S15-S25, "Digital forensic approaches for Amazon Alexa ecosystem." Information such as account, searches, communication (calls/messages), Wi-Fi (passwords), linked calendars, lists of skills, lists of third parties, user activity, devices using the service, and the voice file (utterance and actual user's voice) are stored on the Cloud. Information on the mobile device is generally stored within SQLite databases and can include token information, user identification, and a web cache. The web cache can contain valuable historical data to include URLs, searches, and more.

The information that can be obtained from this type of consumer IoT device is quite amazing and could be used in many different types of investigations. In fact, in 2015, a case in Arkansas was in the spotlight and hinged on Alexa data. After a murder occurred at a residence, the owner of the house was arrested. Subsequently, digital devices at the residence were seized, which included an Amazon Echo. Police sought to determine whether anything had been recorded on the device at the time of the crime. Via a warrant, prosecutors requested that Amazon release the data, which Amazon refused to do, indicating that the request was over-reaching. In 2017, the homeowner gave his consent to release the data from the device. Once released, however, the data was not used in the case and the murder charge was eventually dismissed. The data was not usable in this particular case, but as described, Amazon would not release the data even with a search warrant, most likely because the supporting facts in the affidavit did not support "why" this data may be relevant or "what" could be obtained.

Google Home

Google Home was released as a smart speaker in 2016, with Google Assistant added as the artificial intelligence operator for the device. Google Assistant had been preceded by Google Now, which was the voice search function within Android devices since 2012. The device, along with Google Assistant, enables the user to play music, control other smart devices for home automation, set a calendar, make calls, read news, and much more.

In 2017, two other devices in the Home series were released, the Home Mini and Home Max. All devices, much like the Amazon Echo, respond to voice commands. The wake word (or hotword) for Google Home is "Ok Google" or "Hey Google." A difference in Google Home hardware versus, say, the Amazon Echo Plus is the Amazon product's distinction as an IoT hub. Google Home hardware is not considered a hub in that it cannot relay data from one IoT device to another IoT device, primarily because it lacks a common, global wireless protocol such as ZigBee or Z-Wave.

Similar to Alexa's skills, the Google Assistant has "actions." Actions are how Google Assistant's functionality can be extended. Utilizing an API via the Google Client Library, a developer can build a story based upon what the user wants to do and the fulfillment of the action. Actions are available on the "Actions on Google" page at https://assistant.google.com/explore. All actions are carried out within the Cloud, so the voice request is recorded, sent to the Cloud and analyzed, and then returned to Google Assistant for a reply. All requests, either by stock Assistant functions or a third party, are logged in the user's Google account.

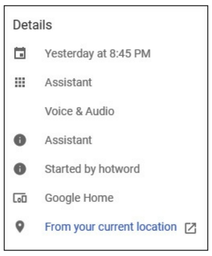
The Google Assistant and Alexa have many things in common, but one feature that can be of interest for investigators is the number of supported voices. Google Assistant and its Home products were first to allow the recognition of multiple individuals and deliver content based on the person speaking and/or making the request. Alexa, however, made this functioning available only recently. Both devices record requests to their servers, and investigators can play the recorded content to determine, by voice, who made the requests.

Google Home devices can be controlled and maintained by the Google Home app. Within the Home app any device (such as Chromecast or Google Home) can be controlled and viewed. The activity for the device can be viewed by selecting the device from More Settings | Devices, clicking the three dots in the upper-right corner of the screen, and selecting My Activity. Also, data can be found under the user account at myactivity.google.com.

Where Is Google Home Hiding the Data?

Like the Amazon Echo, Google Home devices do not have large storage disks or RAM; the Home device has less storage. Google Home devices have built-in 256MB flash for storage and 512MB synchronous DRAM (SDRAM). The operating system of the device is built on Chromecast, which happens to be based on Android and is called Chrome OS. Currently, commercial tools cannot physically attach to the device to extract data using regular means (such as USB or Bluetooth). Other invasive techniques are available to include the removal of the flash module. However, as with the Echo, parsing of Google Home device data will be manual because commercial tools do not support the file system artifacts.

Because of the format and functionality of the Google Home device, and much like the Amazon devices, the data is almost completely Cloud-based. The Google Home app, as well as the Google Assistant app, stores settings for the devices, but the data travels to the Cloud, is processed, and then returns to be delivered. Clicking any of the settings for activity within the Google Home app, the user is sent to the myactivity.google.com site, which is further evidence of the lack of persistent data within the app itself. An investigation of the data that is recovered from the Cloud store under a user account is extremely verbose. Items include location, with date and times, voice and audio transcribed and recorded, search history, and third-party app usage (such as Spotify, Facebook, and so on) via Google Assistant. All this information is displayed in the myactivity.google.com site for the user (Figure 3-2). Several commercial tools, including Oxygen Forensic Detective and Cellebrite UFED Cloud Analyzer, support the extraction of the Google My Activity.


Details

Yesterday at 8:45 PM

Assistant

Voice & Audio

Assistant

Started by hotword

Google Home

From your current location

FIGURE 3-2 **A quick look at recorded details of a Google Home request that includes incredible data for an investigator**

Infotainment Systems

Infotainment devices built into today's vehicles (in-car entertainment) are considered a subcategory of an IoT device, by definition, with "extra features." However, this inclusion speaks primarily to the user interaction area. But what about the car's performance or other systems that interact with the car's manufacturer, other components, or other vehicles? A clearer name, as it relates to data for an investigation, would be "in-vehicle-information system." In 2018, Gartner projected that more than 50 million automobiles were connected to the Internet through these devices. Gartner also mentioned that in the United States and Europe alone, each vehicle system produced more than 50GB of data per hour. In this section, these systems will be explained to help the investigator understand how important they can be to a forensic investigation.

What can an examiner expect from device systems within a vehicle that is involved in an investigation? From location data within built-in navigational systems, to the connected mobile devices, to what is or has been occurring in and around the car during driving time and stopping time—data can be available. To be more precise, vehicle forensics is an emerging technology in the digital forensic world that deals with vehicle telematics, GPS data, and the onboard infotainment system. Obtaining diagnostic information such as a check engine light, transmission, or other mechanical information is authorized by all vehicle manufacturers, and one can simply take the vehicle to an authorized shop to decipher the transmission. Extracting data from an infotainment system and/or telematics, however, is not an authorized service provided by the automotive industry, nor do car manufacturers outline the methods required to obtain this information. With many different vehicle models and vehicle manufacturers, this type of investigation can be extremely difficult. Much like the investigations in the early days in mobile forensics, it involves a lot of hit and miss.

The investigation of a vehicle's system can serve many different facets of an event or series of events. A vehicle's data can often fill in the blanks where a mobile device, CCTV, or other investigative tools lack a certain time frame. Obtaining information from a vehicle can help identify whether a vehicle was at a geographic location at a particular date and time, as well as the vehicle's route of travel. Furthermore, using a linked mobile device to an infotainment system in a vehicle can show a link between the two, and even identify the number of occupants in the vehicle—not to mention contents of messages and other artifacts. Think about determining a time line of travel, the stops made, and which doors were opened and closed; all of this data, and more, can be recovered from the vehicles of today.

Obtaining Data from Vehicles

Like data from any other digital device, vehicle data must be obtained to process and decode into a format to assist in an investigation. The on-board diagnostic (OBD) port can be used by investigators to recover diagnostic and vehicle information, as can a USB, or investigators can remove the infotainment center if one exists.

Both the OBD port and USB methods are considered noninvasive, since both can be connected to plugs and cables supplied by commercially available tools. However, the removal of the infotainment console to obtain a physical collection often requires the removal of the entire, or a portion of, the dashboard.

OBD-II

In 1988, the SAE (Society of Automotive Engineers) outlined and set a standard plug and diagnostic test signals and further expanded the set in 1996 using OBD-II. The OBD port is used by vehicle maintenance providers, diagnostic tools, forensic investigators, and even insurance companies. The OBD-II can hold valuable information for crash investigations and has been used for many years.

Note

The Verizon Hum has a great web site (www.hum.com/port) to help you find the location of the OBD-II port in a car.

The OBD port can communicate with the electronic control unit (ECU) or engine/electronic control module (ECM) to obtain information from sensors and other diagnostic controls that involve the vehicle's engine, powertrain, fluid levels, and so on. An addition on most vehicles with airbag support is the recording of events by the ECU, which acts much like the black box of an airplane. Although the name event data recorder (EDR) is often used, technically the EDR is a function of the ECU. The data often comprises the short time before and after a crash, including impact area, airbag deployment, speed, and acceleration. Vehicle manufacturers are not forthcoming on exactly what is stored on the ECU as it relates to crash diagnostics or other vehicle data, but since 2010 all vehicle manufacturers are required to give access to this data via the OBD-II port.

Note

CDR Vehicle Software used by crash reconstructionists provides a list of the currently supported vehicles that can be referred to in the event the OBD-II port becomes part of an investigation (www.boschdiagnostics.com/cdr/sites/cdr/files/CDR_v17.7_Vehicle_Coverage_List_R1_0_0.pdf).

There are many control modules, and each module may need to be interrogated to recover data. It may seem that there is only one ECU in the vehicle and that it holds all of the data needed for an investigation. But as Ben LeMere, CEO and cofounder of Berla Corporation, has explained during conversations, there are many modules throughout the engine compartment that hold valuable data, and gaining an understanding of what module holds the necessary data for the investigation at hand is paramount to the investigation.

USB Port

Access to data within the infotainment system of the vehicle can be obtained from the USB port, which is generally located in the center console or glove compartment. The USB port typically is coded as the media access to which the occupant can connect a mobile device to play music or perform other operations. This type of extraction is noninvasive and would be classified as a logical extraction. A standard connection to this port is not used (such as USB/USB), since the protocol will not be USB, but serial.

Tools such as Berla's iVe have a USB-to-Ethernet adapter that connects to this port to conduct the extraction of data. This type of extraction can obtain information such as Bluetooth and Wi-Fi of the connected devices, contacts if synced to the system, calls and SMS messages if a device was connected, any connected storage devices and/or media, the vehicle odometer reading, saved locations on the in-car navigational system and track logs, and more. What can be extracted is highly dependent on the vehicle manufacture. Extraction of this data often takes no more than 20 minutes, but as with mobile devices, if there is a considerable amount of media, this could affect the time considerably.

Physical

For some vehicles, the only avenue of analysis outside of the OBD port will be the removal of the infotainment system. This is an invasive and involved physical acquisition. It is invasive because the module must be removed from the vehicle, often requiring that the dashboard be dismantled, and physical because the circuit board is removed from the infotainment module and read by software. Once the board is removed, it is connected to specialized hardware and software and current is provided, generally using an external power supply for the extraction to commence. Recovered data is often quite verbose, as should be expected with this type of extraction.

Data recovered can contain all the information of a logical extraction, but also information from vehicle events. The events can include door openings and closings, gear shifting, parking light status, and time and system updates. Having this type of information, along with the standard infotainment extractions (such as SMS, Calls, Contacts, Bluetooth, Wi-Fi, and so on), can paint a much better collective picture of the event.

Note

Currently the most utilized software in vehicle forensics is Berla iVe software. The hardware and software can conduct the extractions and process both logical methods (OBD and USB) and physical methods.

Wearables

The term “wearable” is used thought the IoT industry to describe a computational connectable device that may be implanted in or worn on the body. Wearable technology is not new. In fact, the first wearable computer was used not to improve fitness or health, but to predict gaming outcomes in roulette. Granted, the device was not technically a member of the IoT world, simply because the Internet had not been invented in 1955.

In that year, Edward O. Thorp theorized that a wearable device could predict and inform a subject of the outcome while playing roulette. Roulette is a gambling game in which a small ball is launched around a circular wheel in one direction while the wheel is spun in the opposite direction. When the ball begins to slow down, it drops through various “pockets,” or deflectors, before landing and stopping at a number. Thorp built a pocket-sized computer that could fit within a person’s shoe. The wearable computer was activated by a wearer by the use of the big toe. Codes (musical notes) were sent via radio signal to the person making the bets at the other end of the roulette wheel. Differences in the sound would be used by the player to decide on the bet. Thorp and Claude Shannon put the computer into use in 1961 in Las Vegas. In 1998, Thorp penned a document describing this first wearable: “The Invention of the First Wearable Computer.”

Additional wearables continued to be manufactured, such as the large calculator watch popular in the 1980s, digital hearing aids, Bluetooth headsets for phone calls, and several other wearable computers. The Nike+iPod Sport Kit was one of the first athletic tracking devices developed by Nike in 2006. This device has a small transmitter placed in or on the wearer’s shoe, and a variety of information is transmitted to an available device (such as an iPod touch, iPhone, or iPod nano). In 2008, the first-generation Fitbit devices came to the market as the Fitbit Tracker, which measured distance, calories, duration, and floors climbed. Also, as an addition to the activity tracker, a sleep function that measured sleep cycles was included. The data on the device could be uploaded to the user’s account on the Fitbit servers.

The first smart watch, Pebble, hit the scene in 2012. What is interesting in the history of the Pebble is the fact that it was brought to the world in a Kickstarter campaign and would be able to interact with both Android and iOS devices, a first of its kind to bring the smart phone to a smart watch. The product was short-lived, but the company and intellectual property were purchased by Fitbit in December 2016 for $40 million.

In 2012, Google released Google Glass, a form of glasses that included a camera and Internet access. The operator could interact with the glasses with voice commands to capture video, images, and information from the Web and specific apps. It soon became apparent, however, that society was not ready for the glasses because of privacy concerns. Stories of businesses banning the devices and obvious limitations shut the door on the device in 2015. However, in 2017 a better target was selected: the enterprise. Now factory workers can receive instructions and more from the safety glasses they wear and send instantaneous feeds via video of their work. Google Glass was reborn, but for the enterprise—and into IoT.

During the same time Google Glass was being marketed, the first Samsung smart watch was released. In the race for the smart watch, the first Samsung Galaxy Gear was released in 2013, with 4GB of storage and

512MB of RAM. The device, as with the rest of the Gear series, enables the user to receive notifications from apps, calls, messages, and alerts, as well as health apps. In 2015, Apple released a watch that helped make wearable tech convenient and was developed with the intention of encouraging people to stare at their phones less but still maintain connection for notifications. A wearable iPhone generation was born. The Apple Watch enables users to interact with their iPhone and receive messages, calls, app alerts, and more, like the Samsung Gear series. Both Apple and Samsung devices interact via Bluetooth to the device, but they also can interact via Wi-Fi if the device is not within range. For both Samsung and Apple, new devices now also come with an embedded Universal Integrated Circuit Card (UICC) that allows for interaction with the LTE system without a device.

Gartner reported in 2017 that consumer wearables would grow at almost 17 percent worldwide annually. In 2017 alone, 310 million wearable devices were in use, with the Apple Watch leading the way. This, to the investigator in digital forensics, should sound some alarms. Once smart phones were the unknown, but now the industry has to deal with smart wearables.

Adding another source of data to an investigation is often just another problem to tackle, but with the number of users relying on data from their smart wearables, the investigator should take this evidence seriously. Most commercial tools still only extract data from Cloud sources for parsing, with some parsing the applications that support these wearables.

With the massive number of wearables that are available to the world, the impact to investigators will only increase. Understanding the various types of wearables that are available, and their general capabilities, can help shape the type of investigation that will be conducted.

Classification of Wearable Devices

Wearable smart devices can be classified into distinct areas that are generally based on the activity or job the wearable performs for the individual user. From heath care, to fitness trackers, to wearable cameras, glasses, and watches—there are many different types of devices and even more manufacturers.

Health Care Devices

The health care industry is utilizing wearables to help assist in remote monitoring of patients via surgically implanted wearables or unobtrusive wearables. These medical devices relay data wirelessly to a base that then transmits the data via the Internet to a collection point—typically the doctor's office or clinic. Not only are the devices used by doctors, but individuals also use them for self-monitoring. For example, under-the-skin implanted devices can be worn to monitor cardiac arrhythmias. The wearable wirelessly sends data through a hub to the cardiologist daily for several years. The use of wearable IoT devices in this industry has been quite dramatic, and today's investigators should be prepared to deal with them.

Fitness Tracking Devices

This wearable technology is based on the fitness industry. Forbes reported in 2017 that 59 percent of all wearable sales involved a fitness tracker within a smart watch. Nike+, Fitbit, Samsung, Garmin, and Apple wearables each have fitness tracking capabilities. With Fitbit leading the way in fitness tracking, Apple and Samsung pivoted from providing only app notifications to adding heath and fitness information. Investigators should be very cognizant of these devices during an investigation.

Cameras

You might not think of cameras as a wearable IoT device, but think about the proliferation of the GoPro revolution. From sporting events, to hunting trips, to law enforcement, and even weddings—people are wearing these portable cameras on their bodies. Many of these devices have internal storage, but many can and do transmit data via Bluetooth or Wi-Fi to a terminal or smart phone and then directly to the Internet. Google Glass is not only a camera but also has a built-in monitor to view messages, Internet feeds, and more. Video data that is stored and transmitted by these devices should not be overlooked by the investigator.

Other Smart Wearables

From smart rings to clothing, the world is becoming a digital investigator's dream (or nightmare). Wearable jewelry—rings, bracelets, and even cufflinks—can be purchased and worn to track activity and alert the wearer of calls and text messages. Because most of these devices offer minimal storage, evidence would be located on the smart device in the form of a Bluetooth link to a connected device and possibly an associated app.

Smart clothing includes shirts, pants, headbands, and more. Because most wearables focus on health and fitness, the smart clothing industry is 100 percent focused on this group. Companies such as Polar have always offered heart rate bands, but they now offer simple compression shirts that not only monitor heart rate but also track motion, duration, and GPS. This data is sent through a hub (typically an app on a watch or phone) in real time to whomever the wearer would like to view the data. One of the most innovative pieces of clothing is the Levi Commuter Trucker Jacket by Google's Project Jacquard. This jacket enables the wearer to answer phone calls, get directions, listen to music, and more, simply by swiping or double-tapping the cuff. Most investigators will have the evidence on the mobile device or hub, but seeing a data-storing denim jacket on the forensic lab table is not just for the future anymore—it is now.

Smart Watches

As mentioned, most smart watches focus on the wearer's fitness activity, primarily because the industry is providing products to supply that specific demand. However, the smart watch has many other functions—paying for merchandise; notifying the user of calls; enabling the user to make calls, send texts, receive texts, view application responses, and so much more. Typically, the smart watch interacts with a paired smart phone, but newer devices from Samsung and Apple have LTE capabilities and do not need the associated smart phone to operate independently since they have embedded UICCs. These devices will be the most prevalent for the investigator in all the different types of wearable investigations. Three major types of smart watches are on today's market: Apple Watch, Fitbit, and Samsung Gear.

Apple Watch The Apple Watch was the first new product from Apple since the iPad in 2010. The device was also the first new interface from Apple since the 2007 iPhone. The Apple Watch has since seen three generations after the first classic Apple Watch (series 0). The first-generation watch, as well as the second and third, all run a variant of iOS called watchOS, with the API kit called WatchKit. The current version of watchOS 5 on the new fourth-generation Apple Watch is 4.3.1, based on iOS 12. The hardware contained within the device is an S1, S2, or S3 processor (the number corresponds to the series), with 8GB of flash storage, near field communication (NFC), Wi-Fi, an accelerometer, a gyroscope, a heart rate monitor, a microphone, and a speaker. Newer devices from series 2 also include a built-in GPS and ambient light sensors. Series 3 devices added a barometric altimeter and Siri voice feedback.

The Apple Watch enables a user to receive notification of fitness activities (such as distance, steps, calories, and heart rate), outside temperatures, calendar, mail, maps, calls, messages, app notifications, and more. The Apple Watch uses the same methods the iPhone does to back data up (iTunes or iCloud). The interesting thing is that the device automatically backs up the data to the paired iPhone, so this information is found within the device iTunes or iCloud backup. Many commercial tools miss this data in the parsing of the iTunes or iCloud backup and simply report the health data.

Several interesting artifacts are contained in the Apple Watch backup. (Specific details and paths will be covered in Chapter 16.) The following list, from Apple Support (https://support.apple.com/en-us/HT204518), shows the items that are available in the backup:

- **App-specific data (for built-in apps) and settings (for built-in and third-party apps)** For example, maps, distance, units, and settings for mail, calendar, stocks, and weather
- **App layout on the home screen**
- **Clock face settings** Includes current watch face, customizations, and order
- **Dock settings** Includes the order, whether user sorts by favorites or recent settings, and available apps
- **General system settings** Includes watch face, brightness, sound, and haptic settings
- **Health and fitness data** Includes history, achievements, workout, and activity calibration data from the Apple Watch and user-entered data
- **Notification settings**
- **Playlists, albums, and mixes** These are synced to the Apple Watch and music settings
- **Siri Voice Feedback** Controls when Siri speaks
- **Synced photo albums**
- **Time zone**

Information that is not backed up includes Bluetooth pairing, credit cards used by Apple Pay on the watch, and the passcode for the watch. However, the data that is synced between the paired device and Apple Watch is retrievable from the watch backup that is contained in the paired device's backup. Information includes a synched address book, call logs, messages, Wi-Fi, MAC address, e-mail, voicemail, and any other shared third-party apps.

Note

When an Apple Watch is unpaired from the device, a backup of the device is automatically created and the data on the watch is erased. If the device is out of range of the paired iPhone when unpaired, the data will not be the most current in the backup.

A tool called iBus (S1, S2/S3) connector can attach to all Apple Watches and was manufactured to enable the device to be restored using the diagnostic port under the band of the watch. This connector, coupled with MFC software and a dongle, can read the details of the device, including iCloud account details. As with all tools in the wild, it is incumbent on the investigator to test and validate all tools prior to using them on evidence.

Fitbit Fitbit's entry into the smart watch world was in 2015 with the Surge. This device enables the standard watch capabilities but includes fitness functions such as steps, floors, heart rate, sleep monitor, and a built-in GPS. Fitbit's versions of smart watches progressed through the Blaze, Ionic, and now Versa, which added the ability to receive basic notifications from the paired smart phone to include calls, messages, and calendar; Fitbit has since added notifications from all supported apps in both the Ionic and Versa.

The Fitbit brand of smart watches has slimmer specifications than the Apple Watch with a max of 4GB of storage on the Versa, but still supports Wi-Fi, Bluetooth, NFC (on some models), accelerometer, gyroscope, altimeter, and a spO2 (saturation of oxygen in the blood) sensor. The Fitbit brand runs a proprietary operating system called Fitbit OS. As discussed earlier, Fitbit purchased Pebble in 2016, and the Fitbit OS is based upon the purchased IP.

How about the investigation of a Fitbit device? The device has already been used as a witness in several cases. One prolific homicide case in 2015 involved a tale of a husband and an alleged struggle and home burglary in Connecticut. The husband had alleged that the wife had struggled with intruders at the home around Christmas, while he was tied up. However, the Fitbit that recorded the health and the steps taken by the wearer painted a different picture of the events proffered by the husband. By recording almost 1300 steps taken by the victim at the time of the crime, as well as health and activity readings contradicting the events as explained by the husband, Fitbit became a new silent witness.

Fitbit users create an account, so the data is accessible online (Fitbit dashboard) and within the supported device's Fitbit app. The actual data on the Fitbit physical device is extremely limited because of the amount of code space for the underlying OS, apps, and music. However, data is stored in the Fitbit Cloud as well as the installed Fitbit app.

Data listed by Fitbit (https://help.fitbit.com/articles/en_US/Help_article/1133) that can be recovered from the Fitbit cloud includes the following:

- **Body** Weight, BMI, body fat percentage
- **Foods** Any logged food and water
- **Activities** Device data such as steps, floors, calories burned, and more
- **Sleep** Time asleep and time in various forms of sleep

Data found within the application on the device is paired with the device and syncs every time the app is opened. The data found within the app includes information listed previously, as well as any notifications that might have been received on the device. (Artifacts and additional analysis of this data will be covered in Chapter 16.)

Samsung Gear The Samsung Gear series of smart watches was first released in 2013. The devices have progressed from the Gear, to the Gear 3, which is available as of this writing. All devices in the family have

4GB of storage like the Fitbit, but the operating system used by the Samsung devices are largely based on their built-in operating system, Tizen. Tizen is an open source software like Android, and the interface of devices running Tizen is like that of the Android. Samsung's operating system is based on Linux, like Android, and the device has a USB debugging mode much like Android devices. Some of the Samsung Gear smart watches, such as the Samsung Gear Live, run on the Android Wear operating system. The devices have Wi-Fi, Bluetooth, NFC, accelerometer, gyroscope, barometer, heart rate, GPS, and ambient light detector.

The Samsung support includes much of the same features as both the Apple Watch and Fitbit to include health features from the S Health app, and also interfaces with the paired device to show notifications from calls, messages, and any supported apps. The device is also able to utilize Samsung Pay to store and manage credit cards using NFC.

The Samsung Gear devices utilize the Samsung Gear app for both iOS and Android, use Samsung Cloud for backups as well as on the mobile device with which it is paired, and store data within the actual watch. Accessing this data is often the biggest challenge for the investigator. The connection to the device running Tizen is often hampered by either no USB connection on the actual watch or simply lack of support by any commercial tool. Most commercial tools simply parse the app from the mobile device or extract the data from the Cloud service. Note that there is a Backup & Restore setting within the mobile device Samsung Gear app. The watch device can be backed up to a paired mobile phone, and the backup can be found in the mobile device file system at data/data/com.samsung.android.hostmanager/files/backup/.

With Tizen on the device, the first step in attempting to access the device directly is to place it into USB debugging. The watch can be placed into USB debugging by choosing Settings | About | Software Version and tapping 7 to 10 times. If the device has a cradle that can be used to plug into a PC, the computer should recognize the device if the current Samsung drivers are installed. However, if the device does not have connection to the device, such as the Samsung Gear 3, an ad hoc Wi-Fi network is created by turning on the Wi-Fi, connecting to the network access point, and on the computer finding the IP address of the device and using this IP address when conducting the communication via the PC and Smart Development Bridge (SDB) as outlined by Samsung in their development manual (https://developer.samsung.com/gear/develop/testing-your-app-on-gear). SDB is similar, if not identical, to Android Debug Bridge (ADB), where an investigator can pull data from the file system. Another similarity to the Android OS is the ability to obtain and the manner of obtaining root on the Tizen device by using Samsung Odin. (A deeper examination of SDB and the Samsung Gear file system will be covered in Chapter 16.)

Unmanned Aircraft Systems

This section will cover the proliferation of recreational drones and their impact on today's digital incident response. In the document "Drone Year in Review: 2017," the Center for the Study of the Drone at Bard College estimated that there were almost 900,000 hobby users of unmanned aircraft systems (UASs) in the United States alone. This number continues to grow each day, as new UAS users hit the streets armed with drones that can fly at great distances, record crystal-clear video, and even carry a payload. UASs are currently regulated by the FAA, which has provided guidelines to local law enforcement on how to handle drone incidents (www.faa.gov/uas/resources/law_enforcement/media/FAA_UAS-PO_LEA_Guidance.pdf).

Another document produced by Bard College, "Drone Incidents: A Survey of Legal Cases," specifically speaks to the legal issues of UASs, to include privacy, crashes, airspace, restricted areas, and smuggling. The following sections will summarize the data and speak to the massive forthcoming law enforcement issues.

> **Note**
>
> All UAS pilots who fly in the United States must register their crafts with the FAA and affix registration numbers to the device. Furthermore, devices have a built-in geofencing mechanism that is set up to disallow the UAS to enter into any airspace listed as restricted. The firmware of the device, replicated on the controller, is updated continually to provide the most current no-fly areas. The FAA maintains a web site (www.faa.gov/uas/where_to_fly/) for information on restrictions, no-drone zones, and a mobile app. The pilot must affix the registration number on the device, and they can also disable the geofencing capability of the drone.

Privacy

With the ability of the drone to be controlled by radio link, the device's distance from the person controlling it can be great. With every drone now equipped with a video camera that can live-feed the information back to the flyer, many people—rightfully so—have privacy concerns. No federal laws cover the use of a drone and privacy—only guidelines outlined by the FAA discuss the issue. The laws covering public concerns come down to state and local governments.

It is easy to imagine how a drone could easily invade the privacy of a person or place. Many state and local cases involve users flying a drone around businesses, in windows, and around homes. One documented case in Utah in 2017 involved the seizure of a DJI Phantom device, where a couple was charged with voyeurism using concealed or disguised electronic equipment. A person saw the drone hovering outside his bathroom and followed it until it landed in a parking lot. The subject removed the drone's SD card and read it via his computer, saw images of multiple people in bathrooms and bedrooms, and took the card to the police. The police located evidence on the card to identify a possible suspect from the video footage and later arrested the couple (http://archive.sltrib.com/article.php?id=4946115&itype=CMSID).

Crashes

Drones can travel at speeds up to 75 mph and at altitudes greater than 1000 feet, and they have spinning blades that can be made of hard plastic and even metal. Often, the pilot crashes the device into a building, tree, or other structure and the device plummets to the ground and onto an unsuspecting person below. In a case cited within the document by Bard College, a subject lost control of a drone and it crashed into a group of people at a parade. A motion to dismiss the case upon grounds that law enforcement could not prove the accused was operating the drone was made and is still under appeal as of this writing. This tactic is important to remember, but there are many artifacts stored on the drone, in the Cloud, and on the SD card that can help establish the owner and flyer.

Airspace

A UAS can fly at incredible heights, and the FAA has placed restrictions not only on altitude but also on proximity to airports. Aircraft pilots having to deal with birds is one thing, but they shouldn't have to deal with drones as well. The majority of cases cited involving airspace issues have to do with drones flying in and around a scene in which first responders are involved. News crews use drones to film the action from a better vantage point, and citizens use them to try to get a better view. Again, being able to identify the device owner from the information located in the various storage locations is critically important.

Restricted Areas

Depending on the location and laws, a UAS will not be allowed to fly in some areas. Airports, as discussed, are obviously no-UAS zones, but there are many other no-fly locations as well. From national parks, to museums, sporting venues, concert venues, government buildings, government facilities/bases, and prisons—each authority has different restrictions. The investigator must be familiar with the various regulations in the area to asses the crime properly.

Smuggling

Within the United States and United Kingdom, the use of drones to smuggle contraband into prisons or over borders has become a problem for law enforcement. What is quite interesting is that the FAA has not enacted specific requirements for flights over prisons, and it comes down to the state to create such laws. Many drones can carry a payload, and there are documented cases in which cell phones, tobacco, alcohol, and drugs have been smuggled into prisons. In a 2016 case in Georgia, subjects actually made several drops with drones of cell phones in a drug-smuggling conspiracy ring (www.justice.gov/usao-sdga/pr/16-defendants-indicted-federal-drug-trafficking-charges).

Obtaining Evidence from Drones

As outlined in the various cases, subjects can affect every facet of law enforcement by nefariously utilizing drones. With the availability of these devices at low prices, almost anyone can become a drone user, and investigations of these types of devices is still in its infancy. However, many commercial tools now support the processing of data from these devices. The drone evidence can be found on the device, the SD card, the mobile device, the controller, and the Cloud service, depending upon the manufacturer and product. The most prevalent family of drones produced for the public are from DJI and include the Phantom, Phantom 2, Phantom 3, Phantom 4, Inspire, Mavic series, and Spark.

In most cases that are processed in drone investigations, the data simply comes from the SD card. In most drones, an SD card is plugged into the device, and the card records the video/images and often log files. In the case of DJI recreational drones, the log files are known as .dat files and contain valuable information that includes altitude, direction of travel, speed, latitude/longitude, and additional device information.

The drone's internal flash memory is accessible via only a few commercial products. Oxygen Forensics Detective software enables the physical acquisition of the flash memory on the DJI family of drones. There is a caveat, however: For physical acquisition, the device must be turned on. Research has found that having to turn the device back on will reset the last logs from the device. The information on the device will contain only the latest flight logs because of the acquisition of a new home location. If the device can remain turned on and charged and isolated from any type of network, the acquisition will contain data not available if the device has acquired a signal or been turned on from the initial contact. Another free method that is also unobtrusive allows for the acquisition of the internal flash media.

Note

Removing the chip and/or using Joint European Test Action Group (JTAG) methods will not have the limitation imposed by a restart, and these are considered better methods for obtaining the information from the drone's internal flash memory.

The following guide will help the investigator gather information on all DJI devices, with only minor changes:

1. Use an iOS device or Android with installed DJI Go app (Phantom 3/Inspire only).
2. Turn on the remote controller (RC) that belongs to the DJI drone (Phantom 3/Inspire only).
3. Have the latest DJI Assistant software installed on the Mac or PC.
4. Connect the iOS device or Android device to the Wi-Fi broadcast by the RC. The SSID is usually located on the RC. The default password is 12341234 (Phantom 3/Inspire only).
5. Obtain a DJI Free account, and log in. Select the device name from the upper-left corner.
6. Launch DJI Assistant and log in with the user created in step 5.
7. Select Advanced mode and select Enter Flight Mode Data. (Phantom 3/Inspire only).
8. Connect the drone.
9. In DJI Assistant, select the device and enable SD card.
10. Use tools such as FTK Imager, Oxygen Forensic Detective, and Cellebrite to create a physical image of the device.

(The data acquired using these methods and artifacts will be discussed in Chapter 16.)

With both Android and iOS devices, the DJI Go app holds a tremendous amount of user information, including the device user registration e-mail address, user account information, device logs, media uploaded by the user, and more. Most commercial tools will parse and decode the information from the DJI Go app. (This application and the contents will be described in Chapter 16.)

DJI devices also transmit data from the DJI Go app to the DJI Cloud server. The DJI Cloud account holds a tremendous amount of saved user data, shared data, and flight information. Having access to the DJI Cloud can help the investigator if the drone or event associated controller device (RC controller, mobile device) has not been located or has been damaged.

Many of the recreational drones that may come into the forensic lab follow the guidelines outlined in this chapter: storage on the device, the Cloud, within the app, and on the external media. Data locations and additional information will be covered in Chapter 16.

Chapter Summary

Today's digital forensic investigators have their hands full with IoT devices, including devices that control lights, refrigerators, home security, and temperature; wearables; and connected vehicles. The amount of data that is gathered into Cloud-based services is unfathomable. Information that can be obtained from these sources will undoubtedly assist in any investigation.

The Internet of Things does not necessarily deal only with items in the home, enterprise, or industry sectors. The number of connected devices in the world has far exceeded imagination. We are a connected society, and our need to consume and transmit digital data is insatiable. Not only do we need to be connected using our smart devices, but we need to be connected using our watches, glasses, and even our clothing. Enter the ability to soar to great heights to video, photograph, and simply pilot a UAS, and some people have unfettered access to tools that can take a simple investigation into a digital device to a new level. The digital investigator who was an expert only a few years ago has so much to learn today.

This chapter briefly outlined the tech that is available to the user and general ways to access the data. As mentioned, Chapter 16 will cover specific locations and artifacts that can be examined to help solidify the digital forensic investigation of these devices.

4 Living in the Cloud: The Place to Hide and Store Mobile Data

The first thing to know about the mysterious Cloud is that it is really not that mysterious. Understanding the Cloud comes down to understanding the simple concept of cloud computing. In a nutshell, the Cloud is essentially a server, large or small, that is remotely accessed by another device—a computer or terminal that sends information to and receives information from the remote server.

The first "cloud computing" scenario was created in the 1950s; it simply allowed access to large servers and computers (mainframes) in data centers by dumb terminals on users' desks, so that users and businesses did not have to purchase complex computers for every employee. This progressed into using a single server in the 1970s, with multiple instances of an operating system and storage area using virtual machines instead. In this scenario, an entire company could have an instance of a virtual computer running on a single remote server, and even different types of operating systems. Quite honestly, this concept from the '70s with server space to run software, provide storage, and offer data reception and transmission is what we see today in everyday apps like WhatsApp, Facebook, Line, Snapchat, and more. Data transmitted to the remote server (on the Cloud) is stored and accessed by the app on the mobile device, as well as the data transmitted to the remote server from the mobile device. Instead of cloud computing, this is often termed cloud services or cloud storage.

A recent document showed the data center storage capacity worldwide in 2018 was 1450 exabytes (www.statista.com/statistics/638593/worldwide-data-center-storage-capacity-cloud-vs-traditional/). If you're not sure what an exabyte (EB) is, it's large: an EB is 1 quintillion bytes, or a 1 followed by eighteen 0's: 1,000,000,000,000,000,000. (To put that in perspective, Mars is 34 million miles away and Jupiter is 353 million miles away. That means a person could take 14,705,882,353 round trips to Mars from Earth, or 1,416,430,595 round trips to Jupiter, before they reached 1EB—let alone 1450EB!) A mind-blowing number of 0's and 1's are stored in data farms around the world. The storage of third-party data from applications on a mobile device is similar, and to save space, store data, and make the user experience better, the Cloud provides the answer.

The problem posed by the Cloud to the investigator is simple: What legal grounds does the investigator have to access the information from a cloud service? Chapter 5 discusses a legal argument that can be made when discussing search warrants and who really owns data. Investigators must understand how relevant the data from cloud services can be to their investigations. Moreover, they must comprehend the various methodologies of collecting information from cloud services, the types of protection used, the accessible data, and the tools available.

Clouds and Mobile Devices

Storage on a remote service is nothing new in the computing world. What brought the concept to the forefront was not the 2014 movie *Sex Tape* or the 2018 movie *In the Cloud*, but the use of mobile devices. In 2008, Gartner predicted that the Cloud would be mainstream in two to five years in its hype cycle diagram, posted by TechCrunch (https://tctechcrunch2011.files.wordpress.com/2008/08/gartner-hype-cycle1.jpg). As people began to use more and more software and services via the Internet, the use of mobile devices has brought this concept to the masses (sometimes without them even knowing it).

Most people understand how Dropbox, Google Docs, OneDrive, or any other file storage/saving service operates in the Cloud. If a file can be stored off a local device and in the Cloud, it must be a cloud application or service. But what about apps such as Chrome, Telegram, WhatsApp, Viber, and Messenger? Should these be considered cloud applications, even if they do not offer storage services like Dropbox or Google Drive? Of course, they should! Messaging apps store and offer their services via remote servers (aka the Cloud) to users.

If the discussion is about native mobile app or cloud mobile app, however, there is a difference between the two, which comes down to development particulars. A native mobile app's running environment is the mobile device, while a cloud mobile app uses a cloud server as its environment. The native app does not have to utilize a built-in browser to display the contents of the app, while the cloud-built app must rely on a built-in browser to display the app by using HTML 5, JavaScript, Cascading Style Sheets version 3 (CSS3), and a server-side code base. However, both types of app development styles can, and do, use the Cloud for user data storage, computations, and other needs.

If you're unsure about whether an app uses cloud services, here's an easy test: If the app is installed to another device, or even uninstalled from the device, when it's reinstalled, are all the app data, settings, and other user information still visible in the app? If so, the app is using the Cloud to store data. Furthermore, if a login and password or other authentication is used to start the app, or if the user must log into a web-based system, chances are the data is stored on the Cloud.

What Does This Mean to Investigators?

Once, investigators had to deal with data coming only from the physical mobile device. Data such as messages, contacts, call logs, images, apps, and more was retrieved from the mobile device by following a process and procedure (such as isolate, attach, extract, examine, report) to complete the task. Investigators had what they needed to finish the investigation and close the case. Training teams from various companies taught the in's and out's of processing the device in a forensic manner so that the information could later be used in testimony.

In 2011, when Apple brought to market the ability to store a device backup in the Cloud via iCloud, the world changed. Android, in an effort to keep up with security and Apple, released version 6.0, which forced the device's user data to be encrypted. Devices became more secure by increasing security with passwords, biometrics, PINs, and so on, and app developers began to impose security on their apps, eliminating the inclusion of the main app database from the backups. Since creating a backup of a mobile device for iOS via iTunes protocols and Android via Android Debug Bridge (ADB) protocols was the primary way commercial tools conducted an extraction, things became a little more difficult.

For Android, trying to obtain a complete physical extraction of the user data is futile if the device is fully encrypted. Today, the importance of cloud data in mobile device examinations cannot be overlooked. A single device can have multiple connections to various cloud sources simultaneously (Figure 4-1). Most apps on today's devices use cloud storage, as do the device manufacturers.

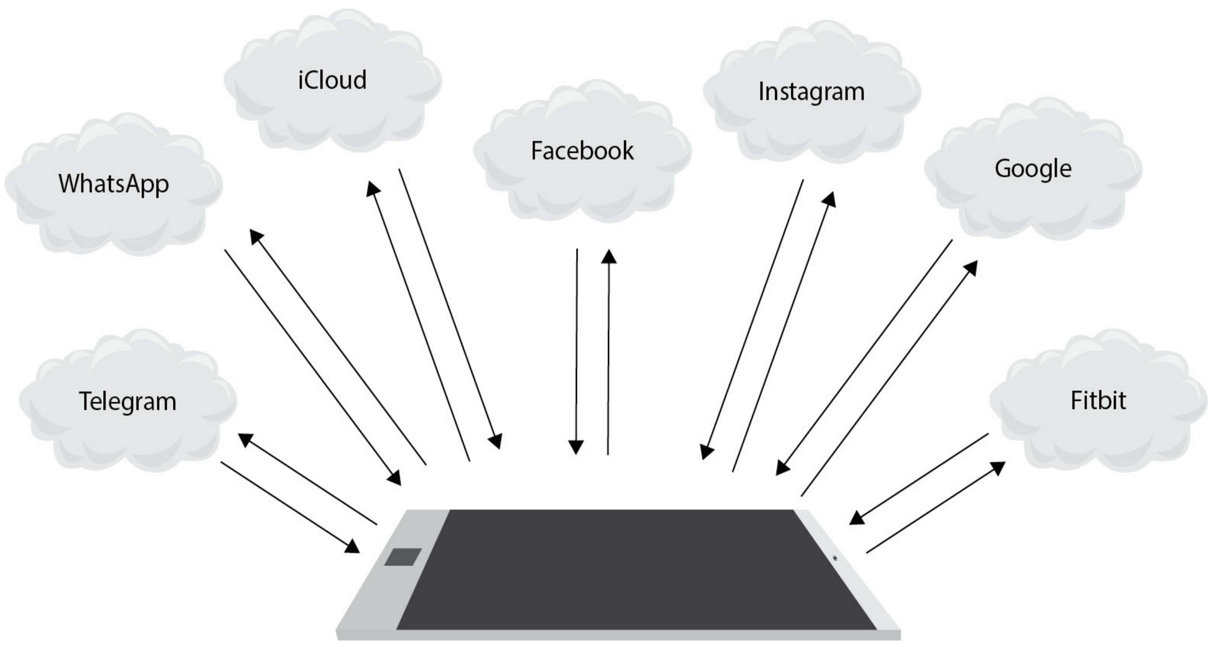


FIGURE 4-1 **Smart phones today have many simultaneous connections to various cloud services.**

Note

As with all other methods discussed throughout this book, any investigation should be conducted in a lawful way that abides by all state, local, federal, and international laws. Utilizing tools or methods for the invasion of privacy, or accessing cloud services without legal paperwork or consent, is illegal.

Today's digital investigators should not ignore the significance of the data stored in various cloud services. Without cloud data, the information that can be gathered from traditional sources (such as mobile device, flash media, or computer) is limited, inconclusive, or simply unobtainable.

No Device Needed

One of the immediate benefits when performing a cloud investigation is the fact a mobile device is not even needed in many cases. All that is needed, other than legal authority, are authentication credentials to the various cloud accounts. Think of the benefit the Cloud offers when a device has been damaged so badly that no customary method can be used to extract data from the device. Also, if the device cannot be located, is lost, or has been stolen, access to the cloud backup or apps is still available. How about if a device owner is missing, lost, or kidnapped, with the mobile device in his or her possession but it's disabled or out of power? By extracting data via the cloud services of the user's various accounts, investigators can obtain location information or other data, which offers immediate benefit to the investigator—and the victim.

Encrypted or Locked Device

With devices from Apple, to the entire Android line of more than 25,000 devices, investigators can encounter several problematic issues. The device can be locked, encrypted, or both. If an Apple device is locked with a PIN, some current options using GrayKey devices can unlock the device, but with Apple's insistence on shutting down these methods, this advantage might be a short-lived win for investigators, as standard mobile forensic methods will no longer work. With Android devices, investigators face the same conundrum: if the device is locked with a PIN and a locked bootloader, or the device data is encrypted, traditional methods and standard mobile forensic extraction may not work.

Access to the data within the Cloud, however, is still available and accessible by the investigator independent of the status or configuration of the mobile device. Essentially, whether or not the device security or encryption is enabled, investigators can extract data from the cloud services used by the mobile device, third-party apps, or the actual device backup process.

Corroborate and Collaborate

Often in investigations, an examination offers two sides of a story. One side may be a person who is using a mobile device to stalk, harass, intimidate, and bully another person; the other side is the person being stalked, harassed, intimidated, and bullied by means other than a mobile device, such as via desktop Facebook or desktop Messenger, who does not own a mobile device (yes, it can happen). By accessing the data from the cloud services of both subjects, along with the mobile device data, the investigator can paint the entire picture. Having the ability to extract information from cloud services for today's investigations is critical.

Disposal of Evidence

Many investigations today involve the subject wiping the device in an effort to remove the "smoking gun" on their mobile device. The devices of today, unlike devices of the past, do a good job of permanently removing data from flash memory with a factory reset and new OS installation. However, as mentioned, third-party app developers want their users to be able to reset their devices and subsequently persist their data. To the investigator, this means that the data that was on the device at the time of the crime could possibly be located in a backup on the cloud storage for the device and/or the app. Of course, accessing the cloud store and services often takes the cooperation of the owner of the device or the cloud service provider. This limitation, however, should not preclude the investigator from attempting to access this valuable data.

Additional Data

App cloud storage as well as cloud backups can contain additional data not available on a standard mobile device extraction completed by an available commercial tool, including physical invasive acquisitions. As an example, Apple iCloud backups can include three backups of a single device—the current backup and two historical backups. Imagine a scenario in which a subject resets the device and the device is backed up to iCloud, but unbeknownst to the subject, the backup performed the night before still is available to be retrieved. Apple indicates that the backup is an incremental backup, but using commercial tools to extract iCloud devices proves otherwise. Additionally, services such as Samsung Cloud, Dropbox, Google Services, WhatsApp, and several other apps cache and store deleted records, including messages and pictures in the cloud store, which is not synced or stored on the physical device. Conducting a robust extraction of the physical device is a must, but what might be missing or additional to the data stored in the devices are multiple cloud storage services.

Accessing the Cloud

Once the investigator decides to access the data from a cloud account and proper authority has been obtained, several considerations remain. For example, the investigator must determine the appropriate date or date range of the data, as well as the types of record he or she needs to investigate. The investigator must also understand the security methods of the cloud account, which can be user driven, a cloud service set, or a combination of both. These methods vary per service, app, and device at times. Finally, an investigator must be aware of the notification settings of the cloud services being accessed.

Date Ranges and Types of Records

In most investigations, the event in question and under investigation occurred at a certain time and on a certain date or range of dates. Having a tool in which the investigator can enter the range of appropriate dates will enable the data to be within the scope of legal paperwork and limit the amount of data that is extracted. Consider a WhatsApp cache of 300,000 messages, which would take about eight hours with most commercial tools to download. Many services store a variety of records (such as images, videos, audio, documents, and text). If the investigation is based upon a particular date range and involves only images, messages, or videos, a tool can be used to select the appropriate date range and type of data. Using a date range or data type to limit the number of records to be extracted can dramatically minimize extraction time and better serve the investigation requirements.

Notifications

Accessing a cloud service as part of an investigation is not as straightforward as entering a few usernames and passwords. Cloud services, particularly Google, can notify users of access to accounts that are deemed suspicious, not from a recognized device or based on time elapsed. In 2008, Google began notifying its users of IP variations in access to their accounts. If multiple devices are connected to a Google account in Alexandria, Virginia, but an IP address is used in Frankfurt, Germany, the owner of the account is notified. The owner can accept or deny this access. The device and operating system, along with the IP, are displayed to the user via the Google account. Investigators must take notifications into consideration when using tools to access the cloud account. If the investigator is attempting to access a subject's account covertly as part of an investigation, he or she must understand whether the subject will receive service notifications from the app.

Security

Cloud accounts can use various forms of security, such as authenticating via a username and password combination, using tokens or biometrics, and data encryption. To access the data within the Cloud, the investigator needs to offer authentication, and in some instances a second type of authentication in two-factor authentication (2FA). The encryption of the data on the server that is extracted can often be decrypted with the authentication method once the investigator identifies the type of encryption in use. The type of authentication used by the cloud extraction tool will also determine whether a secondary authentication type is required by the app.

Two-Factor Authentication

Two-factor authentication uses two types of authentication to "positively" identify the person requesting access. It has become the de facto authentication type in many mobile apps. Google introduced 2FA in 2011, and today the easiest form of 2FA with mobile devices involves receiving a Short Message Service (SMS) or phone call to the phone on which the app is installed.

Multifactored authentication adds security to a user's login by requesting additional information, which can include the following:

- **Something you know** A password, PIN, or other knowledge-based question only the user would know. This would be the simplest form of authentication and is often the easiest to bypass.
- **Something you have** Represents a physical item such as a fob, hardware token, RFID card, smart phone, and so on. Think of a chipped credit card: the PIN is known to the user, and the card is in his or her hand. The PIN must be entered along with the card and chip.
- **Something you are** Represents something that is integral to the person, such as biometric data—the person's iris, fingerprint, palm print, or retina, for example. This represents one of the most secure forms of security when used in conjunction with something the user has.

Multifactor authentication can present problems with cloud extraction utilities, but there are ways to create a backup if it is enabled. In fact, in 2018, a report in the Verge claimed that 90 percent of people using Google services still did not use 2FA (www.theverge.com/2018/1/23/16922500/gmail-users-two-factor-authentication-google), so perhaps lots of data can be extracted without a second form of security.

Username/Password

Username and password combinations make up the most common method of logging into app and cloud services. Many commercial tools can be used to extract username and password information from the mobile device (such as Apple Key Chain, Android Accounts.db, Android apps), and these can be used to access the user's cloud services. There are, however, disadvantages to using only a username and password to access user data, including notification to the account owner, as well as triggering 2FA.

Authentication Token

When a user authenticates successfully to an app or cloud server, the service returns a token, which is used to enable the user to access the service without having to enter his or her username and password again. A token is like a pass, and it is used, for example, when you open your Gmail account and it logs you in without requiring any interaction from you. Most tokens have an expiry set at the time of authentication, which varies per app or cloud server. Some are good for a single session only, others for two weeks, some for 30 days, and some forever if the user uses the app on the same mobile device.

Commercial tools enable not only the collection of usernames and passwords but also authentication tokens used during the extraction process of a mobile device. The advantage to using the authentication token rather than the username and password is that the account owner will not be notified of the access and 2FA is bypassed.

Methods of Bypassing Cloud Services Security

An investigator can access a cloud account with built-in services by accessing the account directly. As in any investigation, all processes and access to the mobile device, computers, and cloud storage must be backed by legal documentation. Without the required legal documentation, any information that has been extracted will be considered tainted and unusable in court. Discussed in the following sections are some methods the investigator can use to obtain credentials to bypass the requirements of a username, password, and 2FA.

iCloud Bypass

Apple iCloud is used for the backup of both the Mac and associated devices such as an iPhone and iPad. The devices that share iCloud are termed "trusted devices" and can receive a code for the 2FA requirement to access the devices' iCloud account. The investigator can also use a code generated on one of the trusted devices to satisfy this requirement and start the extraction.

If the investigator chooses to send the code to the trusted device, the device must be online and not in an isolated state. This must be documented by the investigator, because allowing the device on the network could cause the device to receive data (such as iMessages, app updates, and so on), to remote wipe, or to perform some other operation. There is perhaps a better way to get the code, however.

Apple allows the user of the device to obtain a code for the device either online or offline. To access this setting on an unlocked, trusted iOS device, go to Settings | [*UserName*] and Passwords & Security. When the phrase "Account Details Unavailable" is shown, select Get Verification Code. If the trusted iOS device is not available, a trusted Mac in the ring of trust can also be used. On the Mac, click the Apple icon at the top left, and then choose System Preferences | iCloud | Account Details. If the device is offline, select Get Verification Code; if it's online, select Security | Get Verification Code. This code can be used for the 2FA request.

Google Bypass

Like Apple, Google can use several types of forms when 2FA is enabled. The Google account holder can create app-specific passwords or use the Google Authenticator to bypass the 2FA requirement completely, and a user can even create a set of one-time use codes. An investigator can also receive a SMS message onto the device, which of course would mean the device has to be online—or, if using a UICC card, the investigator can place the card in another unlocked mobile device or in one with the same service carrier.

Within the Google account (myaccount.google.com), go to Sign-in & Security | App Passwords (required to enter password), and then under Select App, select Custom and then the device in Select Device. Tapping Generate Code will create a unique password that you can use along with the username within the cloud extraction application. This will bypass 2FA, even if it's enabled. All changes within the account are logged to include the event, date/time, and location.

Also, Google allows the use of one-time backup codes to be generated. Once a backup code is used, however, it becomes inactive and cannot be used on subsequent extractions. To obtain backup codes from an account, go to the Google account (myaccount.google.com), and then Sign-in & Security | 2-Step Verification (required to enter password), and then Backup Codes and click Show Codes. Enter the code in the cloud extraction tool when requested.

Google Authenticator is available for both iOS and Android devices. An investigator should determine whether the app is installed on the device, which will indicate that the user has already set up Google

Authenticator in the 2-Step Verification area. If the app has been installed on the device, simply open the app, and use the six-digit number in the cloud extraction application when it is requested.

Bypassing the security for cloud services often goes against the guidelines of device isolation in some instances. Investigators may need to receive a text message or call on the device if multifactor security is in use. In these cases, the investigator must document the steps that have been taken as well as any data that was delivered to the device. Similarly, the investigator should document the process and procedure regarding how a verification code or authentication code was used to bypass 2FA.

Accessible Cloud Data

The information an investigator can extract from a cloud store depends on the app data being extracted and also the product that is used to conduct the extraction. Tools that extract data from cloud store rely on the APIs that are used by the actual apps. To support online cloud storage, commercial tool companies rely on research and development of the communication between the app and server as well as documented research online. This is a slow process and is often hampered by app development teams that continue to change API and data stores to combat the access. It is truly a cat-and-mouse game between the forensic tool developers and the app development teams. Add 2FA to the mix, and access becomes even more difficult for the examiner.

Again, what data is accessible in cloud store will be based upon the available cloud storage API that the software extractor uses. Tools are continually updating the service to keep up with changes by app developers and newly identified access points.

The next sections cover accessible data and extraction information from iCloud, WhatsApp, Fitbit, Facebook, and Google My Activity.

iCloud

The accessible data for iCloud includes the following, as listed in Apple Support (https://support.apple.com/en-us/HT207428):

- App data
- Apple Watch backups
- Call history
- Device settings
- HomeKit configuration
- Home screen and app organization
- iMessage, text (SMS), and MMS messages (11.4)
- Photos and videos on the iPhone, iPad, and iPod touch
- Purchase history from Apple services, such as music, movies, TV shows, apps, and books
- Ringtones
- Visual Voicemail password (requires the SIM card that was in use during backup)

Within the iCloud backup is app data, which is often extracted separately from the main iCloud backup, typically because the data is encrypted by the third-party app and must be decrypted with another 2FA code. For example, WhatsApp data within iCloud is extracted using the username and password for the iCloud account, and then a verification code must be entered for the extraction. However, after a successful extraction, the investigator will receive a call or SMS message to obtain another authentication code that must be used to decrypt the data that was extracted.

Fitbit

Accessible data from the Fitbit cloud store includes the following:

- Activity logs (daily, lifetime, recent, favorite, goals)
- Body composition (fat, weight, goals)
- Device (devices, alarms)
- Food and water logging (meals, recent, custom)
- Friends
- Heart rate
- Sleep
- User information
- Subscriptions
- Dates and times of measurements

Many commercial tools can be used to obtain some of the information from the apps on the device with few actually accessing the data from the Fitbit cloud service.

Facebook

Data that can be collected from the cloud accounts of Facebook users includes the following:

- Groups (participants, posts)
- Profile
- Friends
- Albums
- Time line
- Messages (attachments)

If a user authorizes Google to authenticate his or her Facebook account, the user's Google credentials can also be used to extract the data from the Cloud. Furthermore, any account connected to Facebook, such as Instagram, Map My Run, or Endomondo, can also be accessed with the Facebook credentials to extract their cloud data. The caveat to this type of login to a different cloud store is that only the data Facebook has permission to access will be available to be extracted.

Tip

You can use another method to download all of the user's data if access to their account is available. In General Account Settings, choose Download A Copy Of Your Facebook Data. The output is more tedious to wade through, but this is still a viable option.

Google My Activity

In 2016, Google launched My Activity. This cloud service captures a tremendous amount of data and can provide a wealth of information for any investigation. My Activity can be disabled by the user. The data that can be collected includes the following:

- Web searches
- Web sites visited
- Time line with GPS coordinates (any signed in device tracks via pictures, known location)
- Map data (routes, viewed area in map)
- Device information (calendar, contacts, battery level, music)
- Applications (Google Play)
- Call and Message information (if using Google FI or Google Voice service)
- Voice and audio
    - Gboard (Google Keyboard) voice input (entire phrase from any app can be found as well as audio)
    - Google Assistant (Google Home, app) stores entire phrase in activity as well as audio
- YouTube (searched, watched, comments)
- Preferences (news, orders, movies, books)
- Casts to Chromecast (what was viewed)

The cloud service, and the items listed here, are available with authentication of a Google account. The Google Home device's activity is also available within the Activity module, which was covered in Chapter 3 and will be further discussed in Chapter 16. The information that is stored within this cloud store is unbelievably verbose and a complete treasure trove for investigators. Some investigators have utilized Google Takeout to extract the user's Google activity. Using this method should be conducted cautiously, however, as the user is notified immediately of the collection of the data.

Cloud Tools

A variety of tools can be used by the investigator to extract cloud data. Like any other tool, each has its particular pros and cons. Most of the competitive commercial mobile forensic tools provide cloud extractors usually as additional tools that must be purchased in addition to the main tool. In this section, both commercial tools and open source tools will be covered.

As a mobile phone investigator working with today's devices, you need a clear understanding of the capabilities in obtaining cloud data by each service. As mentioned, 2FA and even multifactored authentication are not supported by all available tools, which could pose a limitation. Also, the breadth of coverage for the cloud stores, and the type of data that is extracted from a cloud service, can pose particular issues.

In addition, it's important to know whether the cloud extraction tool allows for a subset of the data, as well specific data based on dates, to be extracted. Think of the importance of allowing an investigator to extract only a subset of the data. Suppose, for example, that the scope of a search warrant dictates that only images can be recovered from a cloud service. The ability of the investigator to select only images during extraction should be a consideration when choosing a tool. Or perhaps an investigator must extract data based upon a particular time period; this can also be critical when selecting a cloud tool. By setting date parameters for the data within the cloud store, the investigator can accomplish a couple of things in a variety of use cases: this would allow an investigator to receive and view only the data specified in a search warrant, and with large data sets, an investigator could target the event time and extract and view only this data, saving valuable time.

The following sections are dedicated to discussions of various commercial tools from Oxygen Forensics, Cellebrite, and Magnet Forensics. The information provided does not cover how to use the tools, but covers their functionalities and some of their limitations.

Oxygen Forensics Cloud Extractor

The Oxygen Forensics Cloud Extractor is built into Oxygen Forensics Detective. The benefit of this solution is that the cloud tool is built into the software—it's not an additional product that must be purchased separately. This enables the investigator to build cases not only from just the data on the mobile device and removable media but also data from the cloud that is related to the examined mobile device.

Oxygen Forensics Cloud Extractor supports, at the time of writing, 54 different types of cloud services, ranging from file storage, to messengers, drones, health apps, and social media. The main software, Oxygen Forensics Detective, extracts usernames and passwords during the mobile device extraction from iOS and Android devices that can later be used to attempt to validate the credentials to extract the cloud service data. The investigator can export a credential package from Oxygen Forensics Detective to be used in the Cloud Extractor on a computer that is online, or simply launch the Cloud Extractor from Oxygen Forensics Detective with the credentials that have been extracted.

The various cloud services currently supported by Oxygen Forensics Cloud Extractor are listed in Table 4-1.

TABLE 4-1 **Currently Supported Cloud Services in Oxygen Forensics Cloud Extractor**

Alexa	Google My Activity	Live Contacts
Android Cloud (Google)	Google Photos	Mail (IMAP)
Apple Watch	Google Tasks	Mi Cloud
Box	Huawei Cloud	OneDrive
DJI Cloud	iCloud Applications	QQ Mail
Dropbox	iCloud Backup	Samsung Cloud Backup
Endomondo	iCloud Calendars	Samsung Cloud Data
Facebook	iCloud Call History	Samsung Secure Folder
Facebook Workplace	iCloud Contacts	Swarm (Foursquare)
Fitbit	iCloud Drive	Telegram
Google Bookmarks	iCloud iTunes Store	Twitter
Google Calendar	iCloud Notes	Viber (Google Backup)
Google Chrome	iCloud Photo Stream	Viber (iCloud Backup)
Google Contacts	iCloud Photos	VKontakte
Google Drive	iCloud Safari Bookmarks	WhatsApp Cloud
Google Keep	iCloud Safari History	WhatsApp Google Backup
Google Location History	Instagram	WhatsApp iCloud Backup
Google Mail	Live Calendars	Windows Phone Cloud

Oxygen Forensics Cloud Extractor enables the investigator to use usernames/passwords and tokens to validate the account. Also, if 2FA is enabled for supported cloud services, the investigator is notified and several options are provided to the bypass the additional step. The steps are determined by the settings for the cloud service. An SMS or a phone call, a validation code, backup codes, captcha, or even authenticator code can be requested after the first validation of credentials. Each service can be different and can use two distinct operations during the process: one to extract from the Cloud and another to decrypt the data.

Oxygen Forensics Cloud Extractor also enables the investigator to select the service subtype (such as images, messages, videos, and so on) within the settings of each service after the credentials have been validated. At this time, the investigator can enter a date range for all services or for individual services to provide a focused extraction the court demands or that time allows.

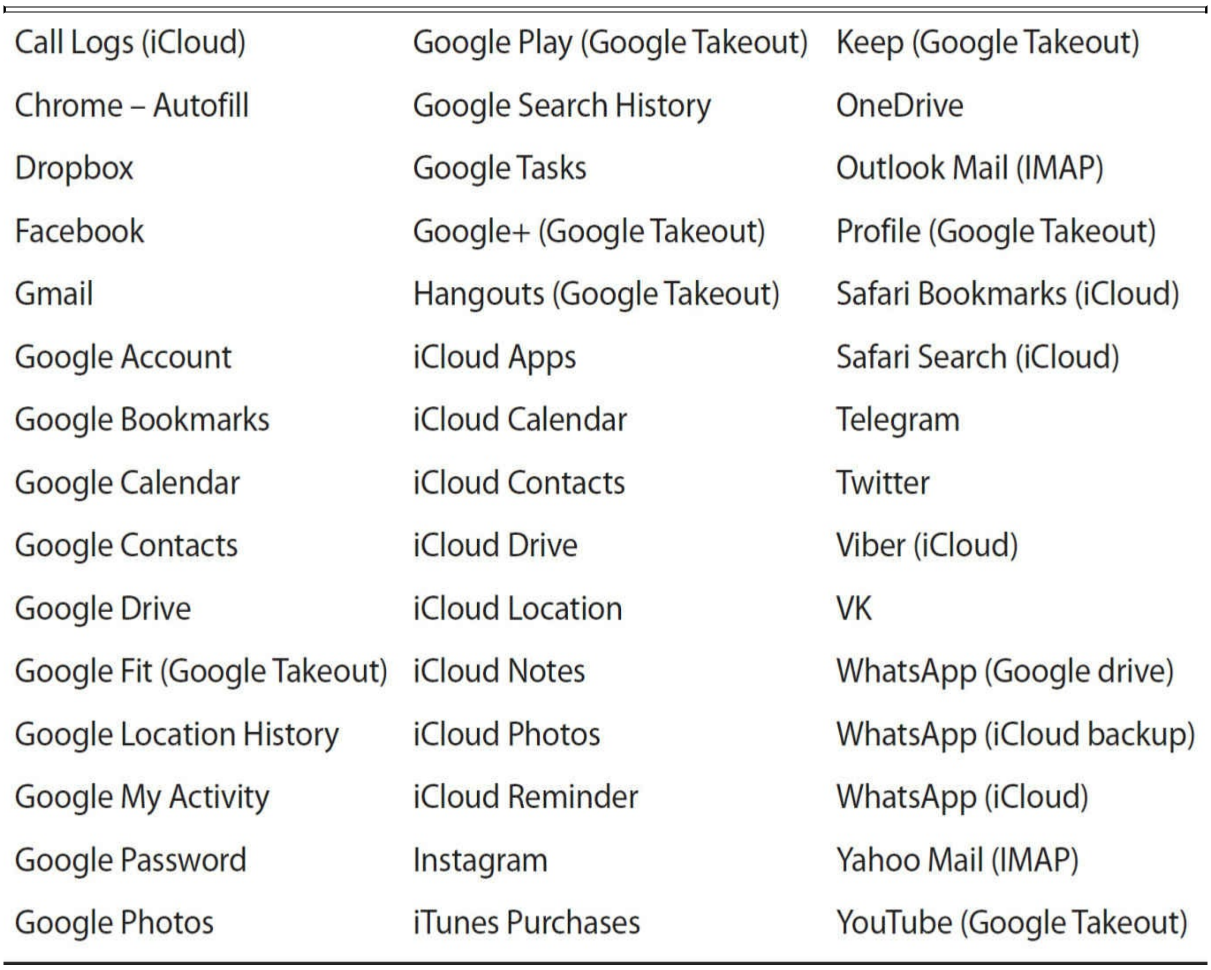
Cellebrite UFED Cloud Analyzer

The UFED Cloud Analyzer is available from Cellebrite for purchase when used with its platform UFED 4PC. Cellebrite offers different types of licenses—one public, one for non-law enforcement (private investigators), and a third for law enforcement only. Each can extract data from the Cloud, but each has separate pricing and requirements.

The public version allows only publicly accessible data to be extracted from cloud services, while the second tier, available to private investigators, includes the ability to obtain public data from cloud services, in addition to a module in which mobile tokens can be used, as can usernames and passwords. The difference, the extraction using usernames and passwords, is consent-based. The law enforcement version is sold only to members of law enforcement, the military, and intelligence agencies and includes the ability to import credential packages from their other solutions that contain usernames/passwords and tokens.

Cellebrite supports, at the time of this writing, more than 50 social media and cloud services. Like Oxygen Forensics Cloud Extractor, the services range from the most popular messengers to mail clients. Table 4-2 represents the currently supported cloud services by Cellebrite.

TABLE 4-2 **Currently Supported Cloud Services in UFED Cloud Analyzer**



Call Logs (iCloud)	Google Play (Google Takeout)	Keep (Google Takeout)
Chrome – Autofill	Google Search History	OneDrive
Dropbox	Google Tasks	Outlook Mail (IMAP)
Facebook	Google+ (Google Takeout)	Profile (Google Takeout)
Gmail	Hangouts (Google Takeout)	Safari Bookmarks (iCloud)
Google Account	iCloud Apps	Safari Search (iCloud)
Google Bookmarks	iCloud Calendar	Telegram
Google Calendar	iCloud Contacts	Twitter
Google Contacts	iCloud Drive	Viber (iCloud)
Google Drive	iCloud Location	VK
Google Fit (Google Takeout)	iCloud Notes	WhatsApp (Google drive)
Google Location History	iCloud Photos	WhatsApp (iCloud backup)
Google My Activity	iCloud Reminder	WhatsApp (iCloud)
Google Password	Instagram	Yahoo Mail (IMAP)
Google Photos	iTunes Purchases	YouTube (Google Takeout)

Magnet AXIOM Cloud

The cloud extraction product Magnet AXIOM Cloud is built into the AXIOM tool. Like the other solutions, Magnet AXIOM Cloud supports the extraction of cloud data from various sources using username/password combinations and/or tokens acquired from data acquisitions of mobile devices and computers. Also, if a mobile device or computer does not contain the necessary information, the investigator can enter the needed credentials manually to conduct an extraction of the cloud service.

AXIOM Cloud supports approximately 25 cloud artifacts in nine parent services to include Apple Box, Dropbox, IMAP/POP, Facebook, Google, Instagram, Microsoft, and Twitter. Each service is broken down into different subservices.

Magnet AXIOM Cloud also supports the extraction if multifactored authentication is being used. In addition, username/passwords and/or tokens can be used to perform the extraction of the supported cloud services, but tokens are not supported for all cloud services offered. One difference from the other tools is the fact that if two-factor verification is enabled for Apple devices, Magnet is unable to obtain the backup.

The tool also enables the investigator to enter a date range for the cloud source. As mentioned, setting a date range can be an important step, and Magnet AXIOM Cloud offers several range types: before a date, after a date, and custom date ranges are available. An important difference from Oxygen Forensics Cloud Extractor is that with some sources, such as iCloud backups and Google, the date range is ignored by Magnet and all the data is acquired. This can be a problem with extraction of data that contains unopened e-mails or other data not covered in a search warrant.

Magnet also supports the ability to select the types of data that will be extracted from the cloud service (such as images, videos, docs, and so on). This is an important feature to allow for customization by the investigator as well as to satisfy the scope of a search warrant. Table 4-3 lists the various cloud services that, as of this writing, are supported in Magnet AXIOM Cloud.

TABLE 4-3 **Currently Supported Cloud Services in Magnet AXIOM Cloud**

Box	Google Photos	Live
Dropbox	Hotmail	MSN
Facebook	iCloud Backups	Office 365
Facebook Messenger	iCloud Drive	OneDrive
Gmail	iCloud Mail	Outlook
Google Accounts	iCloud photos	POP mail
Google Contacts	IMAP Mail	SharePoint
Google Events	Instagram	Twitter
Google Hangouts		

Chapter Summary

Today's investigations have transitioned to a new area: the Cloud. The many reasons for this transition include locked devices, encryption, missing devices, or, most importantly, additional data not found on the device. Another, often overlooked, reason for investigating cloud data is simply because the cloud server is the only place an investigator might find data.

The access to the Cloud using commercial tools must follow a process much like any other type of investigation. First, the investigator must have legal authority to conduct the cloud investigation. Second, the scope of the legal authority should dictate both the type of data and the date range of the data that will be collected. Third, with the various forms of authentication into cloud services, it is important that an investigator understand the role of multifactored authentication as it relates to cloud services. Furthermore, in instances such as WhatsApp, the data that is extracted from the cloud service can be completely encrypted and secondary steps must be undertaken to decrypt the collected data. Finally, the way to bypass the multifactor security of the Cloud varies per service. Google data access can be gained via the Google Authenticator, backup codes, and specialized app passwords, while Apple can use verification codes on trusted devices. An investigator should understand the various methods to bypass verification and should ensure that the tool used will comply with requirements.

Although plenty of data is accessible in the various cloud services, much of it often goes unexamined. Apple's iCloud announced at version 11.4 that messages will be available for access, which is fantastic for investigators. Also, Google services such as My Activity often provide unbelievable caches of data, and having access to this data as an examiner can often complete an investigation. With information in Google My Activity, such as locations, searches, voice, audio, and more, which are unavailable from any other source (such as a mobile device or computer), examiners are sure to find the "smoking gun" evidence.

Many of today's users are into health wearables, from the Fitbit to the Apple Watch, which includes information such as heart rate, location, food intake, messaging, and other valuable data that is often available only on the cloud service and not on the mobile device. By gathering the many artifacts from the Cloud and the mobile device, the investigator can create a collective view of the user's digital footprint.

Commercial tools continue to advance in features to combat the massive amounts of data being stored on app cloud services. Tools such as Oxygen Forensics Detective, UFED Cloud Analyzer, and Magnet AXIOM Cloud all have their pros and cons. It is up to the investigator to look at the features and support before deciding on the type of cloud extractor to use. One of the most important aspects of any cloud extractor is how multifactored authentication is handled and how data from mobile devices and their cloud services can be examined within a single interface. Without these two features, an investigator can be dead in the water, with many of the cloud services enabling multifactored authentication.

As any investigator should know, a tremendous amount of data can be stored on a mobile device, but even more is in the Cloud. Chapter 16 will cover the locations and artifacts for many of the cloud services covered here.

5 Collecting Mobile Devices, USB Drives, and Storage Media at the Scene

Every successful examination starts with a successful physical seizure of the devices that hold the data and/or the repository (such as cloud storage). A physical seizure involves taking possession of the container or containers in which the digital data is stored. This could be, but is not limited to, a server, cloud storage, a computer desktop or laptop, a flash drive, a portable hard drive, and, of course, the mobile device itself.

Lawful Device Seizure

A successful physical seizure of the device along with its contents can be a complicated endeavor. Because the seizure process can be dictated by federal, state, and local laws pertaining to a person's rights, if correct procedures are not followed, your search may be deemed illegal. In general, this hinges on whether or not the seizure and subsequent search were completed by a government agent, an official, or a private citizen. In the United States, a person's Fourth Amendment rights protect them from unreasonable searches and seizures by a government agent, official, or private citizen who is acting on behalf of a government agent or official. That means if a person is not acting on behalf of the government, using a wiretap or other electronic surveillance and searching of mobile devices without a person's consent are legal because searches and seizures by private citizens are not covered by the Fourth Amendment. Of course, the private citizen is not immune from being civilly sued for the invasion of privacy by the subject of the search.

On the other hand, a successful seizure for agents of the government must comply with federal, state, and local laws governing the seizure of personal property from individuals. Any seizure of property must be lawfully authorized prior to the acquisition of the device and the digital evidence. If the seizure or collection occurs lawfully but the actual seizure occurs without lawful authorization, any data collected cannot be used in court and will negate the seizure of the device or devices.

If the location to be searched has been lawfully authorized, the seizure of the device holding the data must be secured. Establishing a chain of custody of not only the device but also the data is very important to the entire digital investigation.

Note

A *chain of custody* clearly details, in chronological order, every hand that the piece of evidence has been in—from seizure to disposition. Typically referenced in criminal cases during introduction of physical evidence, chain of custody can also play a large role in civil cases and should not be dismissed. The chain of custody should be clearly documented in a report or form that defines the appropriate details.

When it comes to electronic evidence from a mobile device, there are two different chains of custody: one for the physical device and another for the data collected from the device. If the device holding the information is not properly seized and a chain of custody is not established at the onset, any information gathered post collection becomes fruit of the poisonous tree—unusable. What does that mean to you as an examiner? It means that any information gathered at the scene that might implicate guilt or innocence can be dismissed at trial because the physical device that holds the digital gold was not properly seized.

For example, suppose a mobile device's data is collected at the scene without probable cause, but the owner of the device consented to your looking at the call records. You recover the collected data to a portable hard drive, and it is subsequently examined by another forensic examiner, who uncovers a heinous plot that is detailed in photographs and videos stored on the mobile device. The alleged suspect is apprehended and later arrested based upon this new evidence. During trial, the digital evidence that was located on the digital device is used, but it is determined that consent was not given by the owner to collect images and video digital data, so the plot information cannot be used in the trial. The rest of the evidence is considered circumstantial, and the case is subsequently dropped.

Note

Lemons v. State, 298 S.W.3d 658 (Tex. Ct. App. - Tyler 2009) states that even if consent has been given for call details on the mobile device, photos from the device cannot be observed and subsequently used under the consent initially given.

In another example, suppose a corporate employee is fired for probable policy violations. The mobile device issued to the employee is taken from him upon termination. The employee indicates that he stored personal phone numbers on the device and the company has no right to look at the information. The company does not comply and believes that since the device is owned by the company, it can examine the information even if a formal policy is not in place indicating employees will agree to a search of the device during their employment and upon leaving the company voluntarily or involuntarily. The company security examiner finds data of a criminal nature on the device and summons law enforcement. The law enforcement officer obtains the information and understands that the device is owned by the company; therefore, the information on the device can be examined without a warrant. The law enforcement agency collects the data from the mobile device and determines the employee has committed a crime and subsequently arrests him. The information found on the mobile device is later used as prima facie evidence in an attempt to convict the employee of a felony. As the trial progresses and the evidence of the device collection is introduced, it is determined that personal contacts, short messages, and pictures were stored on the device. Subsequently, the device was determined to be equal to a personal device, briefcase, or file cabinet that could not be searched without a warrant, even if owned by a third party. The information was located by the company and turned over to a law enforcement officer, who did not seek a search warrant based upon the device being owned by the company, not the person in question; this was fruit of the poisonous tree and could not be used in the trial. The case was later dismissed.

Note

United States v. Finley, 477 F.3d 250 (5th Cir. 2007) stated that even though the cell phone was owned by the company Finley worked for, personal data was stored on the device, and Finley had a reasonable expectation of privacy for the data contained on the device.

Obtaining electronic data using forensic tools should not be the start of the investigation, but the culmination of the successful seizure of the physical device. If the seizure of the physical container is not properly conducted, the examination is inconsequential and worthless should evidence be needed at trial. A proper procedure for mobile forensic examinations should include following a protocol that outlines the seizure of mobile devices in a manner consistent with local, state, and federal laws.

Before the Data Seizure

Prior to the seizure of the physical device or devices, you need to answer several questions:

- If a search warrant has not been executed, has the device owner consented to the search?
- If a search warrant has been executed, is the device included on the original warrant?
- If the device is included on the warrant, are the contents of the device also defined?
- In a corporate situation, is the device owned by an individual or his or her employer?
- Is a corporate policy in place to allow collection and subsequent analysis?
- Could the device contain personal information?

Answering each of these questions is imperative to your successful seizure of the evidence at the scene. Failure to secure answers to these basic questions could lead to the dismissal of evidence recovered at the scene.

The creation of documentation regarding the seizure of the physical devices and other articles related to a mobile device can be different for law enforcement and corporate investigations. What is consistent, no matter whether it is a law enforcement, private examiner, or corporate investigation, is the fact that misinformation typically comes from not asking the correct questions prior to the seizure of the data.

Fourth Amendment Rights

In the United States Constitution, the Fourth Amendment states the following:

> The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.

This amendment grants a “right to one’s privacy.” The government or its agents cannot examine a person’s digital devices without a court order or search warrant issued by a judiciary arm of government. In other parts of the world, privacy rights are also maintained that do not allow the government to obtain data from a digital device without first complying. In South Africa, for example, a mobile device and its contents are subject to search only after the agent receives a search warrant for the contents of the device, similar to the United States. The United Kingdom also holds in *Malone v. United Kingdom* that numbers dialed by a subject are “protected telephonic communications.” In that case, it would be interpreted that any other data that is stored on the device would be protected as well.

The Supreme Court and Mobile Device Data Seizure

Two cases, *Riley v. California* and *United States v. Wurie*, lay the foundation for the change in the doctrine long used when conducting a search of a mobile device.

In *Riley v. California*, a stop for a traffic violation led to Riley's arrest for a weapons charge. Incident to arrest, officers found a mobile phone in the arrestee's pants pocket. The data was accessed, and a repeated reference to a term known to be associated with a street gang was located on the device. Later, another law enforcement officer examined the device's contents in more detail and found media (images and videos) that assisted in charging Riley with a shooting. Riley attempted to suppress all the information from the mobile device on grounds that it was obtained without a warrant, but he was still convicted.

In *United States v. Wurie*, Wurie had been arrested and transported to the police station, where officers removed a mobile device from the arrestee. The officers noticed that several incoming phone calls had arrived, including one message that said "my house" on the screen. They accessed the device to reveal the phone number for the contact "my house," along with an image. The number was investigated, and officers determined that this was the arrestee's residence. A search warrant was then executed on that residence, where officers located drugs, firearms, ammunition, and cash. When Wurie attempted to have this evidence suppressed, he was denied, and eventually Wurie was convicted.

Ultimately, the Supreme Court of the United States (SCOTUS) decided in a historic vote that data on a mobile device should be covered by the same protection outlined in the Fourth Amendment, overturning both of the previous decisions. Furthermore, the opinion read by Chief Justice John Roberts clearly outlined in the decision that law enforcement generally may not search digital information on a mobile device seized via search incident to arrest without a warrant. The decision also stated that digital data from a mobile device does not pose a risk to officer safety. The device itself can be altered into a weapon, but the data itself will not cause harm—so a cursory examination of the physical device is permissible, but not an extraction. Also, destruction of evidence by remote wiping and even encryption could be circumvented with available techniques used by law enforcement, such as using signal isolation devices and disabling the locking mechanism. The most important decision in summary is by Chief Justice Roberts:

> Cell phones differ in both a quantitative and a qualitative sense from other objects that might be carried on an arrestee's person. Notably, modern cell phones have an immense storage capacity. Before cell phones, a search of a person was limited by physical realities and generally constituted only a narrow intrusion on privacy. But cell phones can store millions of pages of text, thousands of pictures, or hundreds of videos. This has several interrelated privacy consequences. First, a cell phone collects in one place many distinct types of information that reveal much more in combination than any isolated record. Second, the phone's capacity allows even just one type of information to convey far more than previously possible. Third, data on the phone can date back for years. In addition, an element of pervasiveness characterizes cell phones but not physical records. A decade ago officers might have occasionally stumbled across a highly personal item such as a diary, but today many of the more than 90 percent of American adults who own cell phones keep on their person a digital record of nearly every aspect of their lives (pp. 17–21).

As clearly explained in the historic decision, a person's rights against unlawful search and seizure by law enforcement extends to a digital device, today's diary. Before conducting a valid examination of a mobile device, get a warrant.

Warrantless Searches

Searching electronic evidence, particularly mobile devices, by civilians who are not agents of the government can occur without a warrant. Individuals may have an expectation of privacy regarding data on a mobile device, but if that expectation is violated by an ordinary citizen, no Fourth Amendment violation has occurred. Ordinary citizens can conduct a search of an electronic device, so long as they have not violated any other laws in order to examine the device—such as breaking into a person's house and taking a mobile device and then conducting an examination. Ordinary citizens do not need the consent of the party to extract data from a mobile device; this is referred to as the "private search doctrine."

In *United States v. Grimes*, 244 F. 3d 375 (5th Cir. 2001), a private citizen searched a computer without consent of the owner and recovered illegal pictures that were later turned over to the police. Because the citizen was not acting as an agent of the government, the search was deemed valid and the recovery of the data was not suppressed. For government actors and agents, however, this would not be the case. Likewise, if the private citizen, prior to proceeding in the investigation, intends to turn the data over to law enforcement, they are unequivocally considered agents of law enforcement and this would not be considered legal.

Note

Warrantless searches of digital devices, particularly mobile devices, by law enforcement were typically conducted using the "search incident to arrest" doctrine, as outlined in several cases. In 2014, SCOTUS changed the way in which law enforcement and government addressed searches incident to arrest as they pertain to mobile devices. The information in this section provides background to the case law used prior to the SCOTUS decision to lay a foundation for the historic decision.

Three particular cases govern, to an extent, when officers can search property found in the possession of or near an arrestee. *Chimel v. California*, 395 U.S. 752, states that search incident to arrest is limited to the immediate control of the arrestee when an officer's safety is a concern, along with preventing the destruction of evidence. Another case, *United States v. Robinson*, 414 U.S. 218, used *Chimel* to explain that a search of a pack of cigarettes found on the arrestee was valid because the risks identified in *Chimel* are always present in custodial arrests, even when there is no concern for an officer's safety or loss of evidence. The last case, *Arizona v. Gant*, 556 U.S. 332, deals with the search of a vehicle, where the arrestee has access to the passenger compartment or other places within the vehicle believed to hold evidence of the crime the person had been arrested for. Using the case law outlined in *Robinson*, it would seem that law enforcement had precedent to extend a search to a mobile device incident to a lawful arrest.

Again, the search incident to arrest clause avoided the need for a search warrant for a vehicle or the person after an arrest of the individual is made. This was to secure the person for transport due to officer safety concerns and remove all possible contraband from the arrestee's possession (such as wallet, keys, weapons, and mobile devices) to disallow access to these items that might harm the officer, the arrestee, or others who might come into contact with the person under arrest prior to booking. Subsequently, search incident to arrest also allowed the law enforcement officer to secure a vehicle if it was involved with the arrest. A search would be performed in the "lunge area" of the vehicle to ensure that the arrestee had not stored any contraband there during the initial contact and subsequent arrest.

Most statutes specify that if law enforcement comes into contact with any locked containers during a search incident to arrest, a search warrant would be necessary to look into the container. Laptops and briefcases typically fall into the search warrant clause, but initially mobile devices did not. Mobile devices, because of their small footprint, were searched either manually or using an automated tool at the scene of the arrest. In 2014, the SCOTUS cited this search and the reasons typically voiced to enable a mobile device search were in violation of a person's Fourth Amendment rights.

Another exception to the warrant has to do with *consent*. A law enforcement officer can stop and search, under reasonable suspicion based on "specific and articulable facts," a person for officer safety concerns, as defined in *Terry v. Ohio*, 392 U.S. 1. If the officer located a mobile device on the person's possession, the officer could request consent from the person to look into the device only if several conditions are satisfied. Consent is an interesting exception to the warrant for many reasons. If an arrestee has a reasonable expectation of privacy for a mobile device and a government representative did not have a warrant to access the data, the officer can request for consent. The person who is giving consent must have standing to give consent, must be capable of giving consent, and must understand that the consent can be revoked at any time during the search.

In *United States v. Meador*, 2008 WL 4922001 (E.D. Jan. 7, 2008), parental consent to search a mobile device owned by their son, but under their account, could not be given. Also, the scope to which the consent applies is important. This means that if a government agent requests to search a room and the mobile device is in another room and is seized, the scope of the consent was exceeded *(United States v. Zavala*, 541 F .3d 562 (5th Cir. 2008). In another example, *Smith v. State*, 713 N.E.2d 338 (Ind. Ct. App. 1999), the government agents requested to search a vehicle for several things but did not specify mobile devices. Several mobile devices were seized and subsequently suppressed at trial because the seizure exceeded the scope of the consent.

Obtaining consent to search a mobile device can be extremely delicate, and you should approach it with caution. When obtaining consent to search a mobile device, you must create a document that clearly details ownership, explains what is to occur, lists the tools that will be used, and provides the outcome if illegal information is recovered.

The last exception to a warrantless search is *exigent circumstances*, when there is not enough time to obtain a warrant for fear of physical harm to the government agent or others, the escape of the suspect, or the destruction of evidence. *United States v. Parada*, 289 F. Supp. 2d 1291 (D.Kan. 2003), indicated that because a mobile device has limited storage, there is a possibility that the information contained on the mobile device could be deleted or overwritten, so a search to retrieve the data was needed immediately to preserve the evidence. To argue, training in today's mobile forensic courses focus on maintaining the device in an isolated state, so that network connections are not allowed. This action would negate this type of exigency in most situations. In another case, *United States v. Morales-Ortiz*, 376 F. Supp. 2d 1131 (D. N.M 2004), it was also argued that access had to be made to the address book under exigent circumstances, but unlike *United States v.*

Parada, which involved call logs, this search and seizure was not justified. (There are, of course, exigent circumstances related to national security, but those instances are beyond the scope of this discussion.)

A somewhat murky area surrounds lost or abandoned property. If a subject leaves a mobile device in a store's restroom, for example, and the owner of the store turns it over to police, can officers search the device? In *People v. Schutter*, 249 P.3d 1123 (2011), an iPhone was left in a gas station bathroom and subsequently searched after the business owner gave the phone to police. Because Schutter returned to try to find the device, it was never lost or abandoned, and the information the government agent recovered was suppressed. In another case, *State v. Dailey*, 2010 WL 3836204 (Ohio Ct. App. 3 Dist, Oct. 4, 2010), a person caught after shoplifting fled the scene, leaving behind a jacket. Inside the jacket was a mobile device, which was later examined by government agents, who discovered an address book that was used to find the suspect. The suspect was located, and the evidence was allowed into the trial since the suspect had abandoned his property when he fled the store.

Always, the best course of action for a government seizure of data is to obtain a search warrant. Warrantless searches, unless by a private citizen, should be undertaken with caution. If a warrantless search is to occur, it must be clearly documented and outlined as to how consent was obtained or what necessitated the exigency. Without that information, it is highly unlikely that the evidence obtained will be usable in a criminal proceeding.

Location to Be Searched: Physical Location

When creating a legal document to search a physical place (residence, business, or site), the *affiant*, or officer who signs an affidavit to obtain a warrant, must describe the physical place, the address, and what should be searched for. This information is gathered after investigation and intimate knowledge based upon probable cause that the items exist at this location and in the particular place defined. This process protects the person and place to be searched from unnecessary searches and seizures—and, more importantly, the neighboring homes immediately surrounding a residence to be searched.

When developing an affidavit for a search warrant at a residence, the affiant must explicitly define clearly, but not be limited to, the color of the home; the type of home exterior construction (brick, wood paneling, and so on); color of any accents (shutters, trim, windows); the address of the residence; any trees, toys, and vehicles in the front of the residence; features unique to the target residence; and what is to be searched for once at the location. Not including this amount of detail in the search warrant upon the execution could have disastrous results.

Location to Be Searched: Cloud Location

What happens when the place to be searched is not a known physical location, but rather a server in another country, or even several servers scattered around the globe? When creating a legal document to search a cloud store, the affiant, or officer who signs an affidavit to obtain a warrant, must describe the company of record and what should be searched for. This information is gathered after investigation based upon probable cause that the items exist, are required, and are stored by the mentioned company. We should not think of the location to be searched as the location of the mobile device, but the *place of record* for the company.

When developing an affidavit for the search of server location(s) utilized by the application company, the affiant must define clearly that data from the mobile device can be stored in several server locations. Helping to explain the storage methods of virtual cloud computing companies will be essential to understanding jurisdiction. This is incredibly important to understand, because this is counter to the requirements of the Fourth Amendment to "particularly describing the place to be searched." Challenge of a company's location for storage services is, however, generally not an issue, because of the understanding that the customer's data can be spread across not only domestic boundaries, but international boundaries as well. Case law, as a result of this web of storage, has by precedence used the company's location of alleged crime as the jurisdiction, as seen in *United States v. Drew* (2009). That case involved a cyberbully, Drew, who used MySpace as a means to contact the victim. The victim committed suicide because of Drew's cyberbullying. In this instance, the state of California was used as the location, when Drew was a resident of Missouri. However, as indicated by Josiah Dykstra in *Seizing Electronic Evidence from Cloud Computing Environments*, (IGI Global, 2013), the jurisdiction was chosen because Drew violated the "site's terms of service." Dykstra further uses the SCOTUS ruling in *State v. Beddow* (1992) to outline the location, which should be determined using a number of factors to include the "site of the defendant's act, the elements and nature of the crime, the locus of the effect of the criminal conduce, and the suitability of each district for accurate fact finding."

Location to Be Searched: Mobile Device

Describing the location to be searched when completing the affidavit to obtain a search warrant for a mobile device should include ample detail. Simply identifying the device as a "Verizon, black, Android smart device" could lead to questions as to what device the original search warrant was drafted to cover. When drafting court paperwork to recover data from a mobile device, the affiant must specifically describe the place and property to be searched. Prior to seizure and subsequent search of a mobile device, the place (the device) and the property to be searched (data) should be defined. You must understand what should be at the scene and how it might relate to the overall mobile device examination.

It is best to document, when available, the following information regarding the "place to be searched":

- The mobile device manufacturer
- The mobile device model
- The mobile device serial number (not the device's phone number; it can be changed easily to another device)
- The color of the device
- The type of cover for the device—even if not unique
- The wallpaper visible on the device screen or lock screen wallpaper
- The presence of a camera on the front and/or on the back
- The presence of a headphone jack—top or bottom or side
- Description of any specific details unique to the device (scratches, broken screen, and so on)

The key is to be very specific, to define clearly to the reader and person conducting the seizure that there is no doubt that the device described in the warrant is the device to be seized. Accomplishing this task with precision in a warrant will lead to better results when challenged on the validity of the item to be searched.

Items to Seize

The second part of the warrant for the mobile device should include the items to seize, which can vary immensely depending on the type of device that is being examined, but more importantly on the circumstances of the seizure. The scope of the warrant will dictate exactly what can be searched for during the examination. Scope is typically dictated by the type of event that constituted the search of the device as described in the affidavit. For example, if the investigation deals with an embezzlement case, the mobile device is collected forensically for images, but audio data might not be within the scope of the warrant if it was not included when writing the affidavit. The same would be true if conducting a collection of a mobile device for voicemails if the investigation was dealing with an image-related crime. Every circumstance, if articulated correctly in an affidavit, should involve every piece of data to be collected from a mobile device.

One fact is crystal clear: data contained on a mobile device should cover the gamut of an individual's actions, regardless of the case. Mobile device data is relevant to each and every investigation in today's electronic investigations. The seizure of mobile device data ultimately comes down to the articulation and understanding of today's communication and its relevance to the investigation.

When specifying the data to be seized from a mobile device, follow these guidelines:

- Research the device and the data types that can be located on the device. Using sites such as phonescoop.com and GSMArena.com can help you locate the user data types. Also, use the device manufacturer's site to ascertain the types of data that may be contained on the specific mobile device.
- Understand that today's communication occurs via third-party apps, so all cases should involve seizure of this information.
- Documenting everyday life, capturing business documents, or storing important notes is often done using the device's built-in camera or microphone. These images, including both video and audio, are saved to the device and can also be uploaded or transmitted via the built-in media viewer, a third-party app, and even near field communication (NFC). Generalizing this information to include not only transmission but also reception of data is critical for every type of collection.
- Text messaging and multimedia messaging can transmit and receive notes, passwords, keys, company information, threats, confessions, audio notes, images, videos, social media shares, and much more. Every investigation should include a search of messaging.
- Personal information manager (PIM) data can include, but is not limited to, call logs, contacts, calendar, and notes. Typically, mobile device users store reminders, monikers, and important dates in these areas, which can constitute influential material for any investigation.
- Installed applications and "stock" applications, or apps, extend the use of the device to allow for access to the Internet, remote servers, gaming, and more. Apps also allow for the transfer of money, images, documents, audio, video, and much more. Understanding what is installed and what has been accessed is very important during any mobile device collection.
- Including deleted data in all mobile device warrant applications should be substantiated by indicating the type of data category (such as SMS, MMS, apps) and why this is needed. Today's devices store data on flash memory, and nonvolatile flash memory stores data even if it has been deleted by the user. Also,

applications used by smart devices use database files that can store data previously deleted by the user.

Note

Today, far more text messages are sent than voice calls every day. According to Domo (www.domo.com/learn/data-never-sleeps-5), in the year 2017, more than 913 million text messages were sent every hour of every day worldwide, not including app-to-app messages.

Articulating why the data is needed from the device is also key to a successful examination, analysis, and eventual presentation. In today's world, everyday life is stored electronically on handheld devices; it could be debated that search warrants could merely indicate "all data on the phone" and be unchallenged. This mindset, however, will eventually lead to case law mandating that collections and examinations be governed by a set of standards. If the standards are not met, the examination and collection cannot be used in a court proceeding. On the other hand, articulating the reasoning for the collection and examination of the various types and formats of data from mobile devices will do two things: the examination and analysis will have meaning, and the focus of the investigation will be clearly defined to assist subsequent examinations.

Tip

The Federal Judicial Center (www.fjc.gov) and U.S. Department of Justice (www.justice.gov) web sites provide sample search warrant applications for electronic evidence along with specific guidelines.

Location to Be Searched: User Cloud Store

Describing the location to be searched when completing the affidavit to obtain a search warrant for the cloud store of the identified user should be specific and detailed. Generic reference to any and all app data from the seized mobile device or identified cloud store will not be sufficient. When drafting court paperwork to recover data from a mobile device, the affiant must specifically describe the place and property to be searched; this is no different from data stored off the device. Prior to seizure and subsequent search of data from a cloud store, the place (the cloud store) and the property to be searched (data) should be defined. You must carefully explain what data should reside on the cloud store and how it will pertain to the investigation and the overall mobile device examination.

A discussion often encountered in speaking to examiners around the globe is the question of who really owns the data. Does the company that is maintaining the server space or the device user own the data? Knowing who exactly owns the data can help in determining to whom the affidavit and subsequent search warrant should be directed.

With all the service contracts and terms of service by third-party mobile applications using servers to store a user's data, it is quite clear that the user owns all rights to the data. For example, WhatsApp's terms of service at Section 5(B[ii]) clearly maintain that "To be clear: you retain all of your ownership rights in your Status Submissions." Furthermore, Google, under its terms of service, indicates under "Your Content in Our Services" that "You retain ownership of any intellectual property rights that you hold in that content. In short, what belongs to you stays yours." Snap, Inc., one of the most widely used apps around the world, also indicates in its terms that the user owns all rights to the content in section 3. Under "Rights You Grant Us" it states, “Many of our Services let you create, upload, post, send, receive, and store content. When you do that, you retain whatever ownership rights in that content you had to begin with. But you grant us a license to use that content.” Most company information is specific to explain that its ownership extends only to the IP of its product, the physical storage medium, and any supporting technology. It would seem, then, that a request for cloud data could be granted by the owner of the data or by a court order served on the owner himself—not necessarily the company hosting the data.

It is best to document, when available, the following information regarding the “place to be searched”:

- Application cloud store
- Unique credentials to access the cloud store
- Date(s) and time(s) of required data

Again, as with a physical device, you must be very specific to ensure that there is no doubt that the cloud store described in the warrant is the correct store to be seized.

Items to Seize

The second part of the warrant for the cloud store should include the items to seize, which also can vary, depending upon the underlying application that interacts with the storage cloud. As with a mobile device, the scope of the warrant will dictate exactly what can be searched for during extraction and subsequent examination. Remember, of course, that the requested items will be in direct relationship with the underlying criminal charge. For example, if the investigation involves a homicide case and messages from the Kik Messenger app, but data from Dropbox is extracted as well from a cloud store, the Dropbox files might not be within the scope of the warrant if the Dropbox files were not included in the affidavit. Every circumstance, if articulated correctly in an affidavit, should involve every piece of data to be collected from an app’s cloud store.

Cloud data should never be discounted. Every functional app on a mobile device uses some type of off-device storage medium. Simply put, a mobile device holds a portion of the facts, and the cloud stores hold not only duplicated data but also new and unique evidence. With frequent issues encountered with physical device extraction data (such as encrypted, locked, or wiped devices), an examiner’s only evidence might be located in the Cloud. If an examiner understands who owns the data, what data is required, and how to communicate the data’s importance to the case, a successful outcome is likely. For any mobile device examination, seizing cloud data ultimately comes down to understanding several facts.

When specifying the data to be seized from an app’s cloud store, follow these guidelines:

- Research the terms and conditions for the app. These are typically accessible on the company web site.
- Understand that digital data from almost all third-party apps resides on a cloud server.
- Cloud data is often not much different from what is stored on the physical device, to include but not limited to text and multimedia messaging, notes, passwords, keys, audio notes, images, videos, social media shares, and much more.
- Data on the Cloud will often include information that cannot be recovered from the actual physical device for several reasons: the data is stored only in the Cloud, the data was deleted from the device, or the data

was not yet synced to the device.

Securing the Scene

No matter where the crime scene is located, it must be safe—and this goes for a law enforcement scene, a corporate location, or an enterprise situation. The safety of the people at the location is paramount. Law enforcement showing up at a location to conduct a routine collection of a mobile device could be very stressful for the person or persons who own the mobile device. The data contained on a device is often compared to the information written in a person's diary—this information is typically considered private, and any seizure of the information can be traumatic for the user. Ensure that the collection location is free from outside distractions, including distractions from the owner of the digital device. Prior to requesting that any occupants leave the premises, you must conduct a guided discussion about the mobile device and any other details that might assist in the process. Sometimes securing the scene is as easy as locking the residence or location; in other cases, you might need to request that the scene be monitored by security while the search is taking place.

Digital data on mobile devices is extremely volatile, and announcing the probability of arrival to conduct an examination of a mobile device will typically lead to dismal results. With most of today's devices, the user can quickly wipe the device of data by navigating to a menu location or even sending a remote signal to the device. It is imperative that you take steps immediately once the scene is secure to ensure that the device or devices that are to be collected remain secure.

Data Volatility at the Scene

As discussed, a mobile device can be wiped of data either manually by the owner navigating to the appropriate menu or remotely by using the device, manufacturer, or telco software. You must take several steps to ensure that this does not occur. Simply removing the device from the possession of the user may ensure that data cannot be manually removed, but what about remote access to the device on the scene?

Cellular transmissions occur via radio waves, and data transmissions can originate and terminate at the device via the cellular signal or a Wi-Fi network. A remote wipe signal can be sent to the device via a cellular signal. Inhibiting the reception of this signal can ensure that the device will not be remotely wiped, thus protecting the data. Also, by inhibiting the cellular signal and Wi-Fi, you can ensure that the device cannot receive additional transmissions such as calls, texts, and other data-related deliveries.

When a mobile device is located, you must isolate the device immediately. Figure 5-1 outlines isolation steps that you should take for a mobile device, depending upon the type of device. By using this simple flowchart, you can be sure to isolate the device by putting the device in airplane mode, if available, or, in the case of a legacy mobile device, placing the device in a signal isolation bag.

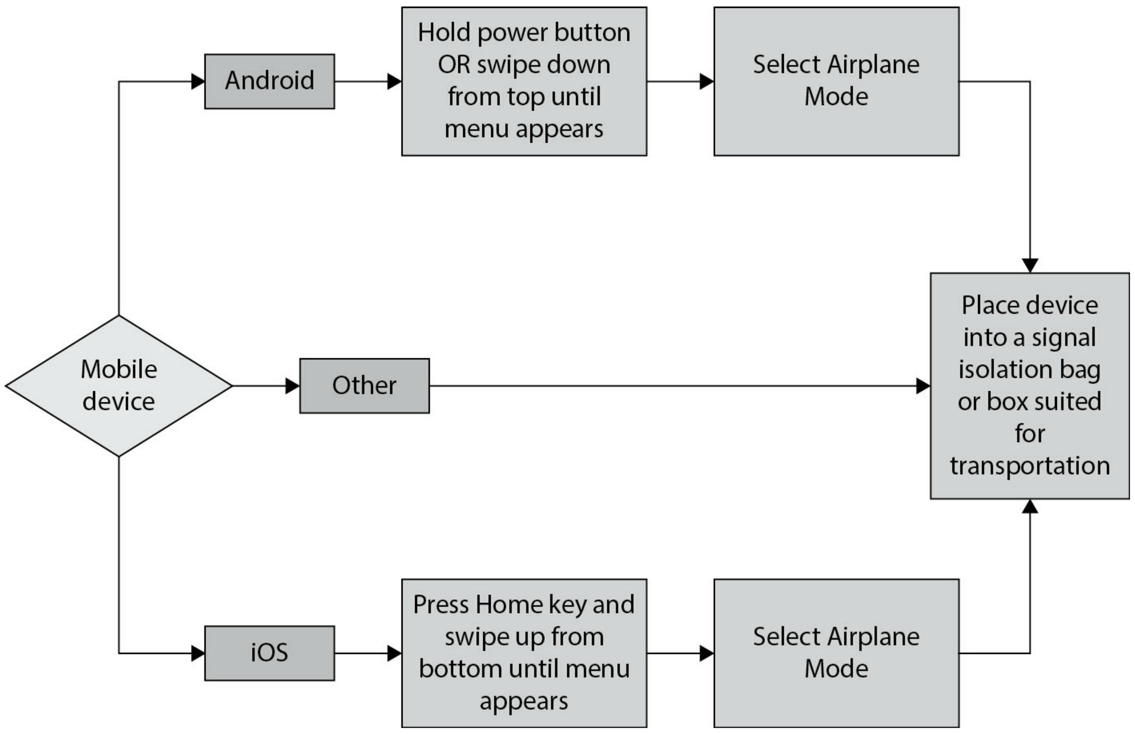

Android
Hold power button OR swipe down from top until menu appears
Select Airplane Mode
Mobile device
Other
Place device into a signal isolation bag or box suited for transportation
iOS
Press Home key and swipe up from bottom until menu appears
Select Airplane Mode

FIGURE 5-1 **Device isolation response on the scene when an examiner is not available**

Note

Newer devices using Bluetooth can also receive data from NFC devices.

Asking the Right Questions

The security of the scene and the mobile device is extremely important, but if the device is locked with a passcode, the investigator at the scene should speak with the device owner, which may help to elicit information that can assist with unlocking the device should the need arise.

The owner of the device can also paint a picture of daily usage, indicate what applications they may use on a regular basis, and indicate how often they use text messaging, who they most often speak to, and how many images the device can hold. Also, common apps have usernames and passwords to sync to the Cloud; obtaining this information is often overlooked. These questions may seem ordinary, but they can be used to direct the analysis of the information once it's collected from the device.

If an owner states that they use an app such as Facebook Messenger, Snapchat, Twitter, or Kik Messenger to communicate, you can begin by analyzing those apps before working with any others. Typically, users of smart devices who are over the age of 20 use standard text messaging; younger users communicate more often via third-party applications. Knowing the habits of the device owner can help you steer the examination later.

One of the most important reasons, but certainly not the only one, to have a conversation with the device owner comes down to security. Device security can inhibit the collection of data from mobile devices and is discussed in the following section, along with alternative possible data locations for mobile device data.

Note

Any government agent or person acting as an agent must take caution during any questioning of a device owner. If the person is in custody or not free to leave, he has the right not to incriminate himself under the Fifth Amendment. The Fifth Amendment states that “a person shall not be compelled in any case to be a witness against himself.” This could involve the person simply providing a password to access the device to a government agent or an examiner acting as an agent. Prior to any questioning, if a subject is in a custodial setting, he must be advised of his rights and consent must be in place. For cases not involving a government agent or actor, the implications of self-incrimination do not apply.

Device Security

Mobile device security can be a real problem during collection of electronically stored data. Two types of security can be enabled on a mobile device: user authentication device security and/or data security.

User authentication security mechanisms include passwords, personal identification numbers (PINs), passcodes and passphrases, patterns, and biometrics. Each provides a different level of security for the device, and sometimes multiple measures are used on a single device. Typically, legacy devices use a passcode consisting of numbers, and if a SIM card is available a PIN and/or PIN unblocking key (PUK) can be used. Smart phones can use locking devices ranging from passcodes to biometrics, depending on what the user has set and the type of options available.

Note

PINs and PUKs are numbers comprising up to eight digits, but typically four for a PIN and eight for a PUK. A PIN locks a SIM card, and a PUK is used to unblock a SIM that has been PIN-locked. These will be covered in Chapter 11 in more detail.

Data security could also pose difficulty to the examination. If the device and any storage areas (internal or external) have been encrypted, the ability to recover that data could be limited. Most smart phones have a feature that enables users to encrypt their data; this can be a separate code in the settings or the user authentication code for device security. Starting at Android version 5.0 devices, for example, data encryption is turned on by default to address user concerns over personal information availability should a device be lost. iOS devices also encrypt their data by default and enable the user to force the device to produce an encrypted backup, where a second password is needed to decrypt. Windows Phones were not capable of using Windows Phone BitLocker encryption unless the device was a managed device under a Mobile Device Management system at the enterprise level to allow for encryption. However, with Windows Phone 10, users are able to use a function called Device Encryption that utilizes the BitLocker technology. BlackBerry devices enable users to turn on encryption in the settings for both the device and the media card. A secondary passcode must be used to decrypt an encrypted BlackBerry device to obtain usable data. With the release of BlackBerry 10, the backup files are also encrypted by default.

Whenever possible, consider obtaining security keys from the device owner. If biometrics are used, the device must be unlocked by the device owner at the location in which the device has been seized, if possible, and all other security measures should be removed. If any other methods of security are used, the agent should request the information and attempt to access it in the presence of the owner. This way, if the code is not correct, the agent can obtain more information and make another attempt. This method may seem to be less than ideal, but with current devices, this may be the only way to collect data from a device forensically. If the agent receives the correct information to unlock the device and remove encryption, if possible, he or she should make the setting permanent so that the device can be successfully examined at a later date if necessary.

Backups

Valuable information can also be located from backups of a mobile device. A smart device, such as an iOS device, creates a backup of data from the device on a computer to which it has been connected, either using a USB cable or via Wi-Fi as well as to iCloud. Other devices, such as a BlackBerry, a Windows Mobile phone, and an Android, can also back up their data to a computer as well as to cloud stores. All of these operating systems can also encrypt the information that has been backed up, which adds another level of difficulty when collecting and analyzing the information. Request information on any devices that might have been backed up if the backups are password protected. The collection and analysis of backups are covered in Chapter 12 (BlackBerry) and Chapter 13 (iOS).

Examining the Scene for Evidence

After gathering information from the occupants or the people at the scene, the agent should search the area systematically. Searching can occur in a pattern, by working from the outside to the center of the location, by using a back-and-forth search, or by dividing the location into several smaller portions. The key is to use a method that will not duplicate the work of others or confuse the team conducting the search.

Prior to the search, take photographs of the area from all angles to allow for documentation. In addition, create a sketch of the area to document dimensions and locations of various pieces of evidence. A sketch will also help when you're searching the various portions of the scene. Each search area of the overall scene should be small enough that a single person can search quickly while a second person provides security and takes photographs. When dealing with electronic storage devices, particularly mobile device evidence, the pieces can be extremely small. Breaking down the search area into small, consumable portions will help searchers avoid overlooking evidentiary items. On the other hand, if the area is too large, a small SIM card or microSD card can easily be missed. The photographs and scene sketch of the area prior to the search can be used to compile a more precise evidence map in the final documentation when creating the report of findings.

Many items related to a mobile device can be found at a scene. These are covered in the following sections. These electronic storage devices and accessories are essential to your investigation's overall success.

USB Drives

Removable USB drives, often known as flash drives or thumb drives, can contain up to 1 terabyte of storage space (as of this writing). Serving as an inexpensive means of transferring data from one device to another, these drives are often used to store images, documents, and even mobile device backups. A mobile device typically does not have a USB connection that would allow direct output to the removable device, but a lot of mobile tablets do provide the connection. Mobile devices do not have much storage on board, and using removable media such as a USB drive can help expand that storage space. Also, data from a mobile device is often sent to a remote location either by messaging, e-mail, or third-party applications and can then be saved to the removable media. Any USB devices collected should be included in the mobile forensic examination.

Chargers and USB Cables

Often during examination of a mobile device, the power for the device becomes depleted. Obtaining the power cable during the seizure of the device can help lower the cost associated with purchasing power cables for every device examined. The search warrant should always indicate seizure of power cables to negate the possibility of losing power once the device is back at the lab for examination.

USB cables are also typically included when the device is purchased and are likely part of the charging unit. USB cables are needed to connect the device to a mobile forensic tool for collection of the mobile device's digital data. Some mobile device units have a unique cable, and if these are not obtained at the scene, processing the device may be postponed until a cable can be located. Most mobile forensic software tools also come with a cable kit, but some do not. If a kit is not part of the mobile forensic solution, a new cable must be purchased for every device that will need to be processed. Make sure the mobile device USB cable is on any search warrant or court paperwork.

SD Cards

Mobile devices do not have infinite storage capacities, so manufacturers have allowed for expandable storage using microSD cards. These cards are extremely small, and locating them can be very difficult if they are not installed in the mobile device. The cards can also be exchanged between different devices—for example, a digital camera SD card adapter could contain a microSD card that was once used in a mobile device. Examiners have reported finding microSD cards with mobile device images that were taken with a digital camera, and vice versa. Because more and more digital devices are using some type of flash storage, you must locate and seize all devices and media storage drives during the scene examination.

SIM Cards

Small SIM cards can be difficult to locate if they are outside of the mobile device. The portability of SIM cards was one of the features that set GSM (Global Systems for Mobile Communications) devices apart from CDMA (Code Division Multiple Access) phones, but this also means multiple SIM cards can be lying around, possibly hidden, at the scene. Old SIM cards, and even SIM cards belonging to different devices, can also be located and collected during the search.

Older Mobile Devices

Often during a search for a particular device, older devices are neglected. But older mobile devices can contain critical data that may be relevant to the current case being investigated. When seizing an older device from the scene, remember to grab the chargers and USB devices that match the old device.

Personal Computers

Computers can contain mobile device backups and information that can be used to unlock a locked device. Apple devices that are backed up to a computer use a property list file that can help iTunes synch data without the user entering the lock passcode. Using this file, the examiner can use forensic software to obtain a backup without knowing the passcode of the device. This information will be covered in Chapter 13 in our analysis of backups and Apple devices.

Conducting a forensic analysis of a computer can also yield passwords for the device, additional synched data from third-party applications, media automatically synched, and documents and settings. Quite often, an expert can use a word list created from a computer examination as a dictionary for passwords on a mobile device. Any computers located at the scene should be further analyzed by forensic examiners for additional information that may assist in the examination of any mobile devices located and later collected.

Once You Find It, What's Next?

When searching a scene, excited searchers can forget all their training and work when they locate the specified target item or items. A search should be directed, meticulous, and calculated to avoid this. After each sectioned area has been thoroughly searched, the team can move to the next area. A forensic search is like an archeological dig, where a 3000-year-old bone could be destroyed if people are moving too quickly. Getting overly excited can often lead a person to miss a piece of evidence such as a small microSD card or SIM card in the same area of the iPad or smart phone that was the primary focus of the search. When items are located to be seized from the scene, follow a set plan for the actual collection; working without a plan can often lead to improper collection and the dismissal of the artifact and possibly the case.

This phase of the search should involve a minimum of two people per team. One person collects the physical artifacts while the other documents the seizure. When at least two people are involved in the exercise, each can realistically monitor the other. This can alleviate any accusations of improprieties. It also enables each individual to focus on his or her assignment. Focus is important when conducting a search and seizure of evidence at any scene. If the people conducting the search are not focused on their assigned duties, things will be missed.

Inventory and Location

The person responsible for documentation should photograph each area from multiple angles. (These photographs are in addition to the overall scene pictures that were taken at the onset of the search.) Once an artifact is located, the searching team must slow down the process to move to the documentation phase. The person conducting the search should identify the location of the artifact without moving it and notify the person in charge of documenting it, who photographs each item in its original position and from multiple angles. If possible, the team should use measurement labels to assist if the area has to be reconstructed and to indicate relational size. If the device is powered on, it is extremely important to photograph the screen that is visible and to document whether incoming messages, calls, photographs, or other data arrived on the device. With some devices, the first few lines of a message are visible and can offer immediate, actionable intelligence. Because a lot of devices can be remotely wiped, documenting data on live mobile devices should be done quickly. Older devices do not have the storage space available on today's devices, so information on these devices could be overwritten with an incoming message or call.

Once the device is seized and the following information is ascertained, the evidence should be placed into an evidence container or RF-shielding material. Before packaging a piece of digital evidence, answer the following questions:

- Will the device be analyzed at the scene?

- Does the device need latent processing?
- Are there biological concerns?
- Is the device security enabled, or has the security been disabled?
- Is power depletion of the device a factor?

Document all of these questions for later retrieval and to determine the way the digital device will be packaged. If the device must stay powered on because of security concerns (such as locking, encryption, or passwords) and power to the device is not a concern, place the device into a radio signal isolation bag and then into an evidence bag to inhibit radio signals reaching the device. (Isolation techniques are discussed in depth in Chapter 6.) If the device can be powered off, place it into a standard evidence bag. If the digital device has any type of biological material that might be contacted during analysis, you'll need to take additional steps (described in Chapter 6).

Before packaging, consider whether fingerprints or other evidence should be obtained from the exterior of the mobile device. If the device is powered on and has to remain on due to security locks, decide on the necessity of immediate retrieval of the data on the device prior to transporting to a lab or a field recovery of latent fingerprints. If the device is to be transported, place it in a radio signal isolation box or bag and then in an evidence container.

For evidence located at the scene during the search, the individual documenting the collection of artifacts should label each evidence container with the following information:

- Location found
- Evidence description
- State of device (on/off and locked/unlocked)
- Processing need (at scene/at lab)
- Owner name (if known)
- Case number
- Person collecting
- Chain of custody (person, date/time)

This detailed information is critical. With the appropriate data documented on each evidence container ahead of time, the examiner can quickly determine what devices need to be collected at the scene. If the data must be collected at the scene, the examiner can perform the extraction of data from the piece of evidence while maintaining an unbroken chain of custody.

Note

Establishing a chain of custody is one of the most important details regarding evidence collected at a scene. This is extremely important in cases when the evidence is located at the scene by one person, who hands that evidence to another, who then takes that device to the person who will conduct the device extraction. As described, the chain of custody would be extremely difficult to maintain without documentation. If documentation is not captured correctly, the entire process and the admissibly of the evidence will undoubtedly be questioned.

Proper handling of digital evidence will be covered in greater detail in Chapter 6.

Data Collection: Where and When

Determining where to collect a mobile device's electronically stored information and the duration of collecting and completing the collection are typically dictated by the type of investigation that is underway. If possible, collect the mobile device in an environment where distractions are at a minimum, safety is not of concern, and time is sufficient. The collection of the data using forensically sound methods and techniques is just as important as the physical collection of the evidence from the scene. Having the luxury of collecting data and examining it in a lab setting is not always possible. More often, data collection, especially for law enforcement, occurs at the scene. Collection circumstances can vary broadly, ranging from a device that locks if power is depleted, to an owner who grants consent to search but will not allow the device to be removed from the location, to the need for immediate actionable information.

If power is depleted or a device is turned off, security settings can render collection of the device data impossible. If the possibility of the device locking will hamper the investigation and device collection, the extraction should occur immediately at the scene.

Consent to search a device is often requested by law enforcement if the device is not part of the original search warrant, the device is on the person under arrest, or the device is at the scene of a call for service. A consent search is an exception to the search warrant rule, as explained earlier. If the person giving the consent has proper authority and waives his or her rights, you can safely obtain information from a mobile device. The person giving consent can also revoke the consent at any time, however, so it is imperative that the data be collected at the scene.

In a corporate situation, a policy can be in place to allow a search of a personally owned mobile device at the workplace without employee consent. Such a policy should be clear, much like the Texas Workforce Commission policy that states the following: "Make it clear to the employee that the employer reserves the right to search any devices, with storage or memory capabilities that they might bring to work and can make copies of any files found therein." For a corporately owned device, the employee doesn't usually expect privacy, and the company can collect the data whenever reasonable.

In either situation, the company will benefit by your searching a mobile device in the most effective way to decrease the time the device will be out of service. Collect a personal device as quickly as possible to help mitigate an employee's emotional response, and collect a corporately owned device as soon as reasonably possible.

At times, the device data is recovered and collected at the scene to be used immediately in the investigation. For law enforcement, that could mean gathering contacts and SMS messages from a device to locate stolen property that is due to arrive that same day. It could also mean the need to locate an accomplice in a crime using the person's contact list. For corporate mobile device collections, it could mean recovering SMS messages of a threatening nature, sexual harassment allegations, or pictures of an incident that need to be collected immediately to gain as much intelligence as possible at the onset of the investigation.

Whenever possible, however, the collection and subsequent examination and analysis should occur away from the scene and in a controlled environment.

Chapter Summary

Creating a detailed plan on the location and seizure of the device is just as important as a detailed plan on the collection and analysis of the data. Understand that privacy rights exist not only regarding a person's place of residence or workplace but also regarding the contents of a mobile device along with cloud storage. Recent court decisions state that a mobile device is the equivalent of a person's private history, and as such it should be treated as a locked file cabinet. If, for example, a significant text message for a case is intertwined with a year's worth of unrelated personal data, and segregating that data is not possible, a search warrant for the content of the device should be obtained. Understanding that in corporate situations, a person's personal data can be stored on the same device with company data makes retrieval of information even more difficult if policies are not in place for collection. And so much more data can be stored off the device on servers located around the world; careful consideration and understanding of ownership is paramount.

When you are conducting a search of the area, making sure to document the scene prior to seizure and identifying where a particular piece of evidence is located will assist later when you complete your report. While on the scene, be aware of items such as storage devices, computers, and media cards, because mobile device data can be stored on each medium. Passwords and encryption may be enabled, so speak to the device owner and look around the scene for any documents that might contain password bypass information. During the search and seizure, clearly document every person who contacts the evidence to maintain a solid chain of custody, no matter whether it is a law enforcement or a civil case. Finally, understanding that the examination of the data cannot occur if the physical device or cloud store is accessed unlawfully will help guide the initial state of the mobile device examination.

While you're at the scene, all evidence, including the device, must be packaged correctly depending on how and in what state it was located and what type of examination will take place. This information is covered in the next chapter.

6 Preparing, Protecting, and Seizing Digital Device Evidence

Conducting a proper initial investigation is a vital part of any forensic examination. How the evidence is prepared, seized, and then packaged after the proper seizure can be just as important, however. Data can be easily destroyed by improperly preparing and packaging a device.

The old adage “don’t sweat the small stuff” does not apply to the investigation of digital devices—particularly mobile devices. The small things are extremely important to the overall success of every investigation. “Bagging and tagging” the device using the proper procedures is not the only concern; the state of the device at the time of seizure and during property transportation and storage is also important. You must also understand the ways in which a mobile device communicates and the types of security used with mobile devices so that you can determine the most appropriate ways to protect the device before and after it is seized.

Data from a collected device can be suppressed during a trial, even after a proper seizure, if it was improperly documented regarding the chain of custody. If investigators fail to package the electronic evidence properly because they are in a hurry, they know the device data will be collected immediately, or if they are unaware of the problems associated with an active or security-enabled network, their hard work is for naught and the investigation can be inadmissible in a court procedure.

Finally, the way in which a device is transported from the scene to a lab or storage facility is also important.

Before Seizure: Understanding Mobile Device Communication

A mobile device is an active device—in other words, a mobile device is attached to several networks that can allow outside communication to interfere with physical collection and subsequent extraction of data. A mobile device can be connected to a cellular network, a Wi-Fi network, Bluetooth, or a near field communication (NFC) device. How you deal with an active device can determine the success of the evidence collection. The physical collection of the actual mobile device and any accessories can also determine the validity of the introduced evidence if used in a legal proceeding.

In the early days of mobile device forensics, the cellular network was the only communication method examiners typically had to eliminate when collecting a digital device. Today’s mobile devices communicate in several ways. Eliminating the risk of contamination of important digital data can be extremely challenging. Those involved in seizing mobile devices must understand how the devices communicate—not only with the cellular network, but with other devices as well. Understanding device communication can help examiners choose the appropriate forensic equipment for use in various scenarios in the lab and in the field, and it can help inhibit a mobile device from all communication to avoid data loss and image integrity issues.

Cellular Communication

A mobile device uses radio frequency to transmit and receive cellular communication and data. Several factors determine the radio frequency used by the mobile device and also affect the way an onsite team and an examiner approach a device to eliminate the possibility that it will initiate or receive communication during or after seizure. Determining factors include the type of device, the cellular telecommunication company, and the allocated frequency band used by the country in which the mobile device is operating.

Chapter 1 discussed the cellular network and how radio waves travel along the air gap from the cellular system's mobile towers to the mobile device. Because the device transmits and receives information via the cellular network, evidence collectors and examiners must use a signal isolation technique to inhibit the radio waves from being received and transmitted by the device. These techniques can include switching the device off, placing the device in airplane mode, wrapping the device in a material that blocks cellular signals, and placing the device into a radio isolation box. Some forensic examiners use large radio isolation rooms completely devoid of windows and lined with special copper wallpaper.

Note

Radio isolation techniques use a principle Michael Faraday discovered in 1836 with the invention of a Faraday cage. Faraday (1791–1867) was a scientist who discovered that electrically charged particles that are approaching a metal object (in this case, aluminum or copper mesh) remain on the exterior of the object. A Faraday cage shields items inside the cage from static electrical fields. All electrostatic charges and electromagnetic radiation are distributed across the exterior of the cage, blocking electric charges and radiation from entering the cage. Similar devices or bags can be used to block radio signals from reaching a mobile device.

Device Frequencies

True 4G devices must use either Frequency Division Duplexing (FDD) or Time Division Duplexing (TDD) Long Term Evolution (LTE). FDD LTE is used globally by more carriers, but TDD is gaining carriers in countries such as China and the Middle East. Newer smart phones are beginning to use both frequencies for communication. The LTE frequency bands extended to 44 with the addition of TDD LTE. Band 43 in the LTE spectrum runs at 3600–3800 MHz, which is not covered by a lot of isolation bags and enclosures.

Tables 6-1 and 6-2 contain the various LTE-allocated frequency bands. Current frequency bands used by true 4G LTE must be isolated by signal isolation techniques.

TABLE 6-1 **FDD LTE Bands and Frequencies: Frequency Allocation Table**

LTE Band	Uplink (MHz)	Downlink (MHz)
1	1920–1980	2110–2170
2	1850–1910	1930–1990
3	1710–1785	1805–1880
4	1710–1755	2110–2155
5	824–849	869–894
6	830–840	875–885
7	2500–2570	2620–2690
8	880–915	925–960
9	1749.9–1784.9	1844.9–1879.9
10	1710–1770	2110–2170
11	1427.9–1452.9	1475.9–1500.9
12	698–716	728–746
13	777–787	746–756
14	788–798	758–768
15	1900–1920	2600–2620
16	2010–2025	2585–2600
17	704–716	734–746
18	815–830	860–875
19	830–845	875–890
20	832–862	791–821
21	1447.9–1462.9	1495.5–1510.9
22	3410–3500	3510–3600
23	2000–2020	2180–2200
24	1625.5–1660.5	1525–1559
25	1850–1915	1930–1995
26	814–849	859–894
27	807–824	852–869
28	703–748	758–803
29	n/a	717–728
30	2305–2315	2350–2360
31	452.5–457.5	462.5–467.5
32	Downlink only	1452–1496

TABLE 6-2 TDD LTE Bands and Frequencies: Frequency Allocation Table

LTE Band	Allocation (MHz)
33	1900–1920
34	2010–2025
35	1850–1910
36	1930–1990
37	1910–1930
38	2570–2620
39	1880–1920
40	2300–2400
41	2496–2690
42	3400–3600
43	3600–3800
44	703–803

Note

If a device can be powered off and security has not been enabled, the device should be placed into airplane mode prior to powering it off. By doing this, the device can be powered on at the forensic lab without concerns about isolation. Of course, if the device cannot be powered off, using portable isolation bags rated for new cellular frequencies is highly recommended.

With the reality of 5G technologies, the range of the current spectrum is estimated at 3300–4200 MHz and 4400–4990 MHz (the primary current bands are between 1 GHz and 6 GHz). On July 14, 2016, the U.S. FCC indicated that approval for bands above the 24 GHz range would be available for use—a first for any country in the world. This spectrum, as the FCC has confirmed, will operate at "fiber speeds" for Wi-Fi. With a quick look at the outlined range, it is no wonder that the United States' early commercial 2018 deployments in the range of 27.5 to 28.35 GHz and 37 to 40 GHz, plus Korea (26.5–29.5), Japan (27.5–28.28), China (24.25–27.5 and 37.0–43.5), Sweden (26.5–27.5), and the European Union (24.25–27.5), have raised eyebrows. This wireless spectrum for a mobile device is close to that of communication satellites, cable and satellite companies, satellite radio, modern radar systems, and even approaching microwave communication. The leap from 4G to 5G technology is astronomical for the industry of wireless communication and data.

Bluetooth Communication

With Bluetooth technology, a user can move data between a mobile device and a computer and attach headsets, headphones, and wireless speakers to a mobile device. In the SANS research paper "Dispelling

Common Bluetooth Misconceptions" (www.sans.edu/research/security-laboratory/article/Bluetooth), author Joshua Wright points out that although organizations typically consider Bluetooth to be short-range technology, class 1 devices can operate at ranges that are typical of a wireless network at 100 meters (328 feet). Granted, to operate at that distance, a class 1 device would have to be at both ends of the communication. Today's mobile devices, including the most prevalent Android and iOS devices, operate as only class 2, at a range of 10 meters (33 feet). You should understand, however, that the version of the Bluetooth device will define its capabilities. Most new iOS devices, to include the Android Samsung 8 and Apple iPhone X, operate using version 5. This version's release in 2017 was to support the emergence of the IoT devices (version 4.2 actually supported IoT) to factor in the speed of data transfer, quadruple the range, and multiple devices.

Some companies employ Bluetooth Smart Beacons, which enable retailers to transmit location information to smart devices. The location can then identify and target the device, sending location-specific data immediately to the device to notify it of a retail sale and gather analytics regarding where in the store most people are shopping. Once considered as just a way to transfer data to a device, Bluetooth can now identify a smart device by using location-based technology.

Note

Location-based technology actually originated from the first hacking, or "Bluejacking," of a Bluetooth device. Bluejacking involves sending messages and controls via Bluetooth to another Bluetooth-enabled device. (Don't confuse this with "Bluesnarfing," in which access to information on the mobile device is compromised and stolen from the device, or "Bluebugging," which controls the device to become a listening device.) All of the Bluetooth hacking techniques are limited by distance because most mobile devices cannot be accessed from greater than 10 meters. With the use of directional amplified antennas, however, some hackers have been able to penetrate Bluetooth device security at up to 1 mile—this is referred to as "Bluesniping." This technique can be especially troubling because a lot of vehicle communication systems in today's automobiles are class 1 Bluetooth devices.

Apple iOS devices, Androids, Windows Phones, BlackBerrys, and most modern mobile devices allow for Bluetooth connections and maintain a list of devices that can connect with the device, their associated media access control (MAC) address, and other available Bluetooth-enabled devices that were nearby but that did not connect. These lists can be obtained using mobile forensic software to observe the connections made with the device and those that were available and were not connected. These device Bluetooth profiles can be used to assist in performing various functions, including transferring files to and from the device; communicating hands-free; accessing Internet hotspots; using a mouse or keyboard with the mobile device; and sharing messages, contacts, and calendar data with another device. All devices must have their Bluetooth enabled for a connection to occur. Androids and Windows Phones must also have their Bluetooth visibility enabled and available to Bluetooth-enabled devices. The device then must be paired to the other device to transmit and receive data. The current Android SDK does not allow for unpaired connections. For iOS devices, the connection to the device must be encrypted and a key must be shared between devices.

Wi-Fi Communication

Wi-Fi is a local area wireless technology (also called a wireless local area network, or WLAN) that enables a device to be connected to an access point that is connected to the Internet and/or a local area network (LAN). The first device reported to have Wi-Fi capabilities was the Calypso Wireless C1250i, demonstrated at the 2006 3GSM World Congress trade show in Barcelona. Most smart devices sold today can connect to a WLAN.

A Wi-Fi connection uses a frequency band identified by the Institute of Electrical and Electronics Engineers (IEEE) using the IEEE 802.11 media access control and physical layer specifications for WLAN communication using the 2.4, 3.6, 5, 6, and 60 GHz frequency bands. A Wi-Fi network's range is typically limited, but as the frequency increases and obstructions decrease (outdoors), the range increases, with a current maximum range of 250 meters (820 feet).

Mobile devices can also operate as a Wi-Fi hotspot using a cellular signal so that other WLAN-capable devices can be wirelessly attached to the mobile device. Because of the various potential infiltration points, a Wi-Fi–enabled mobile device must be removed from the WLAN as soon as possible at the scene.

Wi-Fi connections, like Bluetooth connections, are stored within a file maintained in mobile devices. This allows for the device to connect immediately to known and authorized sites and devices.

Note

Versions prior to iOS 5 maintained a list of all Wi-Fi connections that were available to the device—not just those used by the device. This list was stored in the consolidated.db file in the device file system. When accessed, it identified all the Wi-Fi connections along with their latitudes and longitudes, which could be used to track the device, and the person using it, as it was moved to different locations. This vulnerability was fixed, and the consolidated.db was moved to the operating system partition; however, it came back with iOS 7 and still is visible in iOS 11, but with very limited available information.

Wi-Fi connections contain name and sometimes password information. Android devices are notorious for storing each and every successful connection to the device, information that could lead to security issues if an access point has been compromised. This is true not only for Android devices but also for most devices set to connect automatically to an access point without notifying the user.

If a device is not properly isolated from a Wi-Fi signal during the initial seizure, the device could become vulnerable to penetration. An individual with network knowledge can exploit the device by posing as a known access point, creating an ad hoc network, and identifying the network with the same Service Set Identifier (SSID) used for a legitimate access point. If the device is not disabled from connecting automatically, it can connect with the rogue access point. Some devices, such as the iPhone, come preprogrammed with Wi-Fi SSIDs and can be vulnerable to this attack. A perfect example is an iOS device from AT&T with the "attwifi" hotspot preconfigured. A person posing as a valid AT&T Wi-Fi hotspot could penetrate the device and possibly compromise the data before the device is even examined.

Near Field Communication

Today's devices also can operate as NFC devices that transfer and receive data simply by being near another NFC device or system set up to transmit and receive the signal. NFC is a short-range wireless technology that enables mobile devices to connect by touching the devices together or bringing them within a few inches of each other. NFC enables small amounts of data to be shared between an NFC tag and a mobile device or between two mobile devices capable of communication by NFC.

An NFC tag is based upon the NFC Data Exchange Format (NDEF). Mobile devices that are capable of NFC can make transactions, exchange content, and connect devices. NFC includes a lot of the components of contactless card technology and can be used to control multiple instances of a contactless card (such as hotel keys, work key cards, and so on).

Because NFC is close-proximity based, it is unlikely (though not impossible) that problems will occur during the seizure of a mobile device if NFC is enabled on the device. Attack of an NFC-enabled device would typically occur with another device using a relay system to capture data from the device and transfer it to a proxy card or device that is emulating the actual mobile device and being read by the NFC receiving device, such as a point-of-sale (POS) machine. The technique would enable the relay system to act as the actual POS machine and capture data because the individual's device "believes" that it is communicating with the actual POS device. Software could also be installed on the mobile device to act as a relay system to a fraudulent card emulator communicating with a reader device. Although NFC issues are highly unlikely when you're seizing a mobile device, you should consider isolating the device from the possibility of an NFC attack by turning off NFC.

Something to consider in the mobile device forensic world is the ability to utilize NFC for credit card purchases. Applications such as Samsung Pay and Apple Pay have become common payment methods used around the world. These applications use security protocols (such as biometrics and passcodes), but artifacts can be left behind.

Understanding Mobile Device Security

As mentioned in Chapter 5, a user can enable several types of security on a mobile device. Understanding these can help you determine the appropriate course of action during and after collecting the device.

If a device is powered on without security enabled, you must collect the device at the scene in a forensic manner to counter any security features that might engage if the power of the device is lost or if the device becomes locked before it can be examined at a lab. After a quick collection, talk with the owner to determine the type of security enabled and any ways to bypass it.

Note

Prior to speaking to the device owner, you must have legal standing even to ask questions. If you are a government agent or an actor, the device owner must understand his rights and accept that his testimony could be used against him.

Remember that multiple security settings may be used, such as the device password or PIN, a subscriber identity module (SIM) PIN, an encryption password, and a password for the device's backup if backup encryption is enabled. If you can conduct a quick forensic collection prior to the conversation with the owner about the device's security settings, there is a far better chance the owner will provide the correct information. If the owner does not provide the security information or subsequently has given information that is later determined to be false, the initial forensic collection at the scene will prove valuable. On the other hand, if you obtain the correct information, the forensic examiner can further examine the device at the lab and compare information to what was collected initially. This comparison can be used to show that information was not altered during or after seizure.

Both scenarios—the owner disclosing or not disclosing information—are quite different from the same scenarios in traditional computer examinations, although some drive encryption tools have made computer collections just as challenging. Today, a lot of mobile device forensic collections must occur at the scene because of data volatility and device security. Being prepared for an on-scene collection will help you combat a tremendous number of problems associated with today's device and data safeguards.

Tip

Request the information from the device owner, verify that the information is correct, collect the device on scene if possible, and if the security of the device is not a concern or has been circumvented, power off the device. If the security of the device is a concern for any reason, the device should be collected forensically at the scene.

Most mobile devices encountered at a scene or forensic lab will be smart devices, but legacy phones can also have security features, including passcode or PIN and/or a SIM PIN. Although the general guidelines outlined here will suffice, the following sections discuss the various smart device security features by manufacturer. Your ability to recognize the type of device and respond in an appropriate way will be important for the follow-up collection.

Apple iOS Devices

Apple iOS device security depends upon the model, OS version, and user configuration settings. First-generation iOS devices used only a simple passcode (a four-digit number) and a SIM PIN. These devices were available only for Global Systems for Mobile Communications (GSM) markets until 2010, when an iOS device capable of running on the Verizon network was released. Subsequent generations allowed for both simple and complex passcodes along with a SIM PIN. With a complex password, the user could enter numbers, letters, and symbols up to 37 characters, providing greater security against brute-force attacks. (About 10,000 possible combinations could be guessed in about 13 minutes at a rate of 12 per second.) The latest generation of iOS devices allows for simple (must be a minimum of six digits), complex, biometric (fingerprint scanner) security, and facial recognition according to user settings.

If the passcode is entered incorrectly, a wait period will be required before a user is allowed to access the device. Apple documentation indicates that the device will be disabled for one minute after six incorrect entries and will be completely disabled after ten failed attempts, and the device will have to be restored and/or unlocked via iTunes. Table 6-3 lists the waiting times in relation to failed attempts.

TABLE 6-3 **Failed Attempt Consequences for iOS Devices (from support.Apple.com)**

Failed Attempts	Added Waiting Time	Total Waiting Time
1 to 5	none	none
6	1 minute	1 minute
7	5 minutes	6 minutes
8	15 minutes	21 minutes
9	60 minutes	81 minutes
10	Disabled	Disabled

Apple has added a layer of security to enter a passcode/password even when Touch ID has been enabled per their support (https://support.apple.com/en-us/HT204587), to include the following:

- When the device is restarted
- When you have not unlocked the device in the last 48 hours
- Five unsuccessful Touch ID attempts

What is also interesting is the fact the last point has led to a somewhat "kill switch" for users under investigation by law enforcement. If a user believes he might be compelled to unlock the device with Touch ID or even Facial ID, he can click five times to require a passcode/password, essentially locking the examiner from the device by disabling the biometric security.

Recognizing the various screens that might be encountered can assist as you document, seize, and further examine the device. Figure 6-1 shows the possible iOS screens when requesting a simple passcode and a complex passcode. For a complex passcode that contains only numbers, the number pad will be available instead of the iOS keyboard, along with the complex passcode text box for the numeric entry.

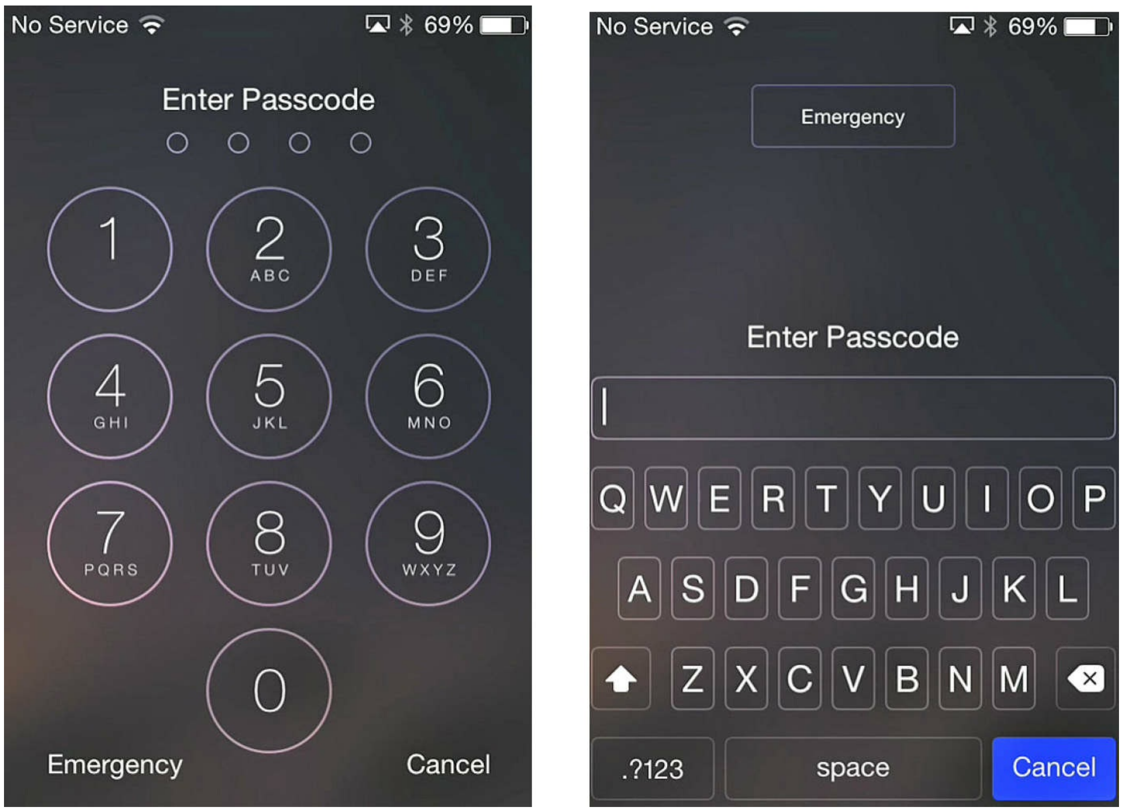
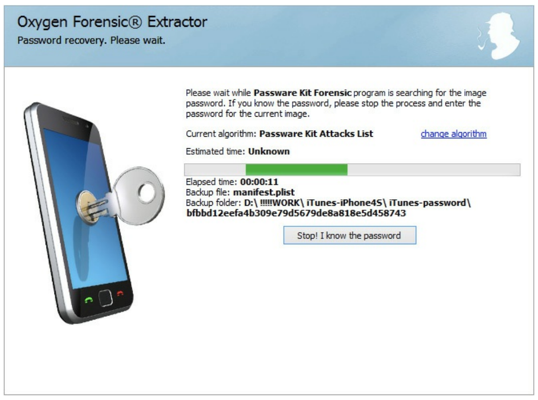


FIGURE 6-1 **Simple (left) and complex (right) iOS passcode entry screens**

Devices starting with iOS 4 added full disk encryption. This means that any unallocated space on the device remains fully encrypted even if the password is known. Apple also enabled the user to encrypt backups using an iTunes setting. When enabled, this sets a flag on the iOS device when synced to encrypt the data stream as it leaves the device for backup. So even if the device was not protected by a user password, the data that was collected would be encrypted if the iTunes password was not known. There is no visible setting on the iOS to indicate whether the device has been set to encrypt the backup. To determine whether backup encryption is enabled, you must launch the iTunes application on a computer, plug the iOS device into the same computer, and examine the iTunes software device information screen. The information screen indicates whether backup encryption is enabled. Some mobile forensic software, such as Oxygen Forensic Extractor, will indicate whether backup encryption is enabled during connection and prompt for the password, as shown in the next illustration. Some tools will even allow for a password attack.


Oxygen Forensic® Extractor
Password recovery. Please wait.

Please wait while **Passware Kit Forensic** program is searching for the image password. If you know the password, please stop the process and enter the password for the current image.

Current algorithm: **Passware Kit Attacks List**

Estimated time: **Unknown**

Elapsed time: **00:00:11**
Backup file: **manifest.plist**
Backup folder: **D:\ !!!!!WORK\ iTunes-iPhone4S\ iTunes-password\ bfbbd12eefa4b309e79d5679de8a818e5d458743**

The investigator should request the iTunes password from the user of the device at the time of the device seizure when possible. If the user refuses to provide this information, some mobile forensic tools can be used to bypass and recover limited user data as indicated earlier.

Note

The iTunes backup password can be different from the device password or code.

In iOS 8.0, released in 2014, Apple changed the way device encryption worked to allow for greater security. The iOS device can use the passcode of the device to encrypt the device so that Apple itself is unable to recover the user data stored on the device if the passcode is not known. Prior to iOS 8, law enforcement officers had been allowed to send locked iOS devices with proper court documents to Apple security analysts, and if specific criteria were met, the officers would receive a disc image of the user partition for analysis. With the changes in iOS 8, Apple no longer used the same methods and was unable to assist with these types of recoveries.

Now, with iOS 11, the fact still remains: if an iOS is locked with any of the security measures described in this section, the device password must be obtained from the user of the device before any type of forensic collection can occur. Some exceptions have been brought to light, however, in the wake of the attack in San Bernardino, California, in December 2015, when Apple refused to offer assistance to the United States government to unlock an iPhone 5C.

After some time, a research company brought a solution to the government for approximately a million dollars. The device was purportedly unlocked using a NAND mirroring technique, but the evidence was dismissed. This technique involves removing the NAND chip, creating a clone, and running passcode attempts—then repeating, essentially allowing unlimited passcode attempts on the clones.

Another alternative that has been used for bypassing iOS screen locks was the IP-Box. This hardware box, with a software component, is used to flash an iOS device to allow for unlimited passcode attempts by shutting down the device and attempts prior to the imposed restriction by the device. The device then can use either a number dictionary or number sets. The change in iOS failed attempts was a direct result of the success of the IP-Box with security passcodes. IP-Box still maintains it can unlock iOS 10*x* devices, but there are great possibilities that the device could be wiped—not good for evidence and not recommended for examinations. In 2018, however, Grayshift, a U.S. company, released a device and a method to obtain a raw file system extraction using its tool GrayKey. This tool is currently available to law enforcement.

The examiner should document any type of security, if enabled, along with the device type and operating system of the iOS device. This information should also include any known passwords at the time of seizure for both the device and iTunes. Having this information will assist with collection at the forensic lab.

Android Devices

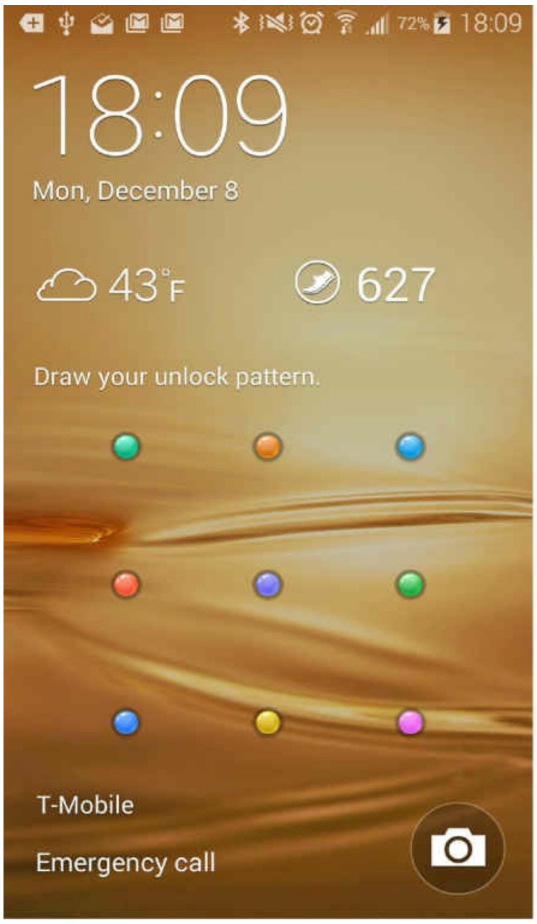
Android devices brought a new type of security to the mobile device: the pattern. The first release of the Android OS allowed the use of a four-point pattern within the 3×3 grid, but newer devices allow the use of all nine points. The increased number of points elevated the security for the pattern, but the use of a pattern is still the lowest form of security for an Android device. With a nine-point pattern, about 50,000 restricted (same dot only once) pattern combinations are possible; with a four-point restricted pattern, about 1400 combinations are possible. In fact, a "smudge attack" can typically reveal the pattern of the Android user if the device is held at a 60-percent angle to a light source, as outlined in the article, "Smudge Attacks on Smartphone Touch Screens," published by the University of Pennsylvania (www.usenix.org/legacy/event/woot10/tech/full_papers/Aviv.pdf).

Note

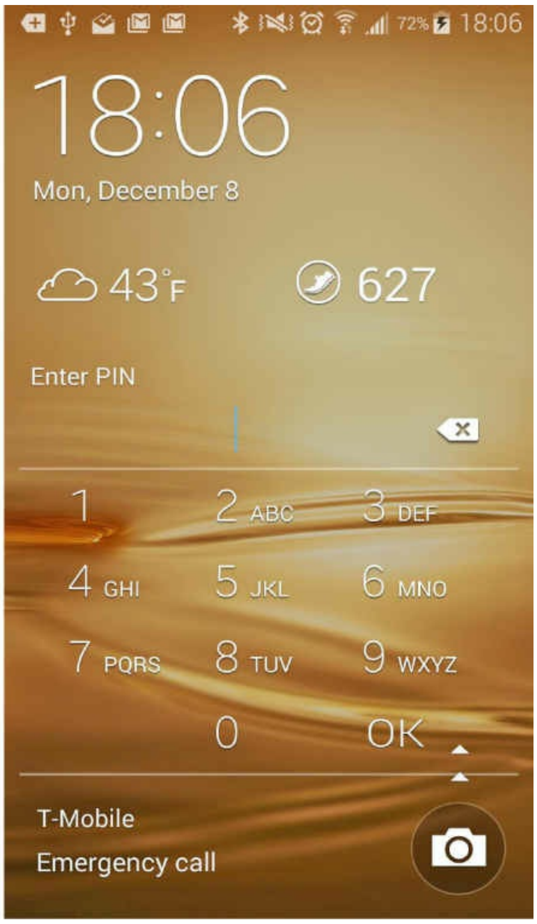
A *hash* is stored in a key file within the Android file system. This file can be extracted and analyzed using forensic tools to reveal the pattern used to secure the device. This will be discussed in Chapter 15.

Because of the various security vulnerabilities, more security options were added to later versions of the Android OS. Android devices now allow the use of patterns; PINs; passcodes; passwords using letters, numbers, and symbols; and biometrics. The user enables security in the device settings, and if the examiner knows the type of security that is enabled, he or she can determine the viability of a bypass during a mobile device collection. More than 12,000 different Android devices are available on the global market, and identifying the exact type of security on these devices is difficult at best.

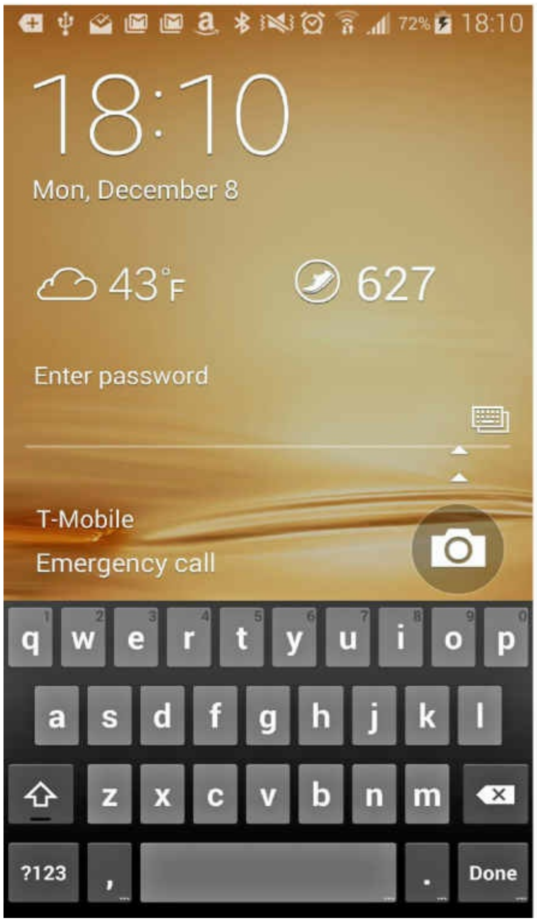
It is recommended that an examiner look at the device screen for clues as to the type of security in use. As shown in Figure 6-2, a set of dots indicates a swipe pattern, and a dial pad with nine numbers indicates a PIN. If a text cursor and keyboard are shown, a password or passphrase is required. If a fingerprint image is used, the device will show a location for the finger press that will unlock the device. With devices containing built-in fingerprint readers, the user is notified on the screen to "Use fingerprint or swipe screen to unlock." Not shown in the figure is a screen with an active camera, which indicates that facial recognition is required. All biometric security features are backed up with another form of security—typically a PIN. If the device cannot be unlocked by biometrics, a PIN can be used.



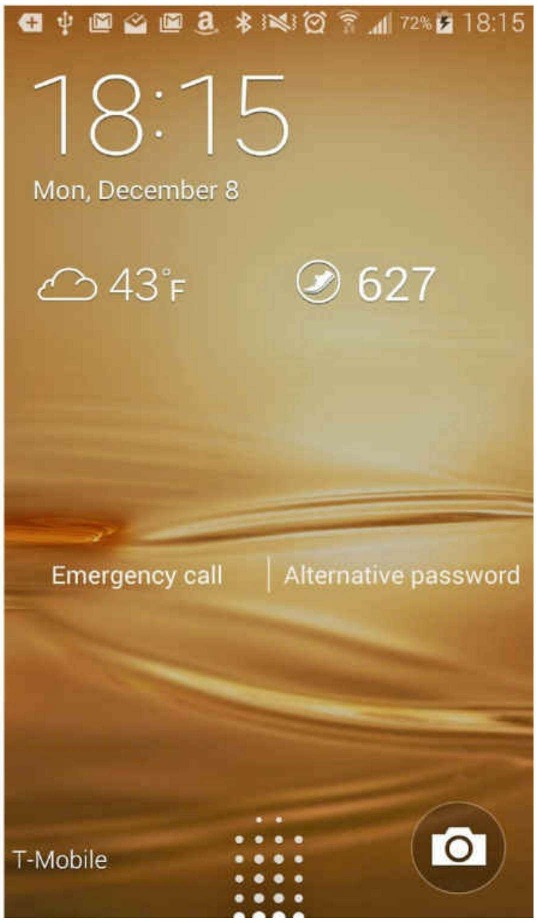
72% 18:09
18:09
Mon, December 8
43°F
627
Draw your unlock pattern.
T-Mobile
Emergency call



72% 18:06
18:06
Mon, December 8
43°F
627
Enter PIN
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
OK
T-Mobile
Emergency call



72% 18:10
18:10
Mon, December 8
43°F
627
Enter password
T-Mobile
Emergency call
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
,
.
Done



72% 18:15
18:15
Mon, December 8
43°F
627
Emergency call
Alternative password
T-Mobile

FIGURE 6-2 **Android devices showing (clockwise, from top left) a swipe pattern, numeric PIN, biometrics (fingerprint), and passphrase**

No matter what security is used, even if the device is locked, it can be accessed with mobile forensic tools if the Android Debug Bridge (ADB) is enabled. However, ADB is typically not enabled by default. If it is enabled on the device after it is seized, the device can be forensically analyzed even if the screen is locked. (ADB will be covered in depth in Chapter 15.) When an Android device is seized, if access to the Settings menu is available, the device should first be placed into airplane mode, and then the device ADB setting should be enabled and applications turned off. These settings will depend on the version of Android. (Refer to Table 15-1 in Chapter 15 for version information.)

Prior to Android 3.0, full disk encryption was not available. With version 4.0, encryption is included in the system settings, and the user can choose settings to encrypt the contents of the mobile device along with data on the external memory card. With Android 5.0, full disk encryption is turned on by default. Android 7 and later versions now utilize file-based encryption. The difference now is that each file can have a different encryption key, versus full disk where a single key is used. Device encryption will not inhibit the standard user data collection of most mobile forensic tools should extraction be conducted logically. Physical collection, however, could be impacted simply because the entire produced image still will be in the encrypted state upon extraction. There are mobile forensic tools, such as Oxygen Forensic Detective, that incorporate passware to combat this disk encryption problem.

However, all device profiles are not supported. Nevertheless, it is important that the device be properly placed into ADB mode as soon as possible to at least allow for a robust logical extraction.

The type of security enabled, along with the Android device type and operating system, should be documented. This information should also include any known passwords at the time of seizure.

Windows Mobile and Windows Phone

Windows devices have transitioned from simple PIN and strong alphanumeric passcodes in Windows Mobile 6.0 and 6.1 to passwords only in Windows Phone 7 and 8 devices. All devices can also use SIM PINs to block calling features. The security password can be set by the user in the system settings of the Windows Phone 7 and 8. These devices are extremely difficult to examine without knowing the passcode or PIN. This information should be obtained from the user if the device will be later examined. On July 11, 2017, Microsoft stopped supporting Windows Phone 8.1 while moving all research and development to the Windows 10 phone. Targeting corporate usage, these devices offered not only the security of the previous devices, but now biometrics to include fingerprint, facial recognition, and iris scanning while incorporating Windows Hello. Industry professionals in both the law enforcement and private sectors indicate they seldom encounter a Windows Phone or Windows 10 phone for processing. This is indicative of the less than 1 percent market share for this Microsoft product.

BlackBerry Devices

BlackBerry devices have always been known for their security, which is not easily bypassed. BlackBerry devices first used a simple PIN and a password or passphrase. Later versions could use a PIN, a passcode, or passphrase, plus a password for data encryption. There is no known way to bypass BlackBerry device security to collect the device's data without the handset's lock code when using forensic tools. A BlackBerry Enterprise Server (BES) can reset the device passcode if the device is part of the BES and that setting has been enabled previously. If the device also has a passcode set for data encryption, this must also be known. Gathering this information from a device user is especially important when dealing with these types of devices.

BlackBerrys can create a backup of their data, which can also be protected by passcodes or PINs. If the device password or PIN and encryption password are known, an unencrypted backup can be produced. If the password or PIN is not entered for the handset lock, a backup cannot be initiated and the device's data cannot be recovered. BlackBerry 10 devices added another tier to the security of the file system backup. Even if a device password is known and entered into the device, and even if the device is unlocked, the username and password for the BlackBerry Link software must be entered. The BlackBerry Link software is the Research in Motion (RIM) software used to update firmware and software and sync the mobile device with a computer. In 2015, BlackBerry released the first device (BlackBerry Priv) devoid of its proprietary OS that instead uses Android 5.1.1. Currently, there have been only Android devices released by BlackBerry with the latest, BlackBerry Motion, released in 2017 and running Android 7.1. The security utilized is similar, if not identical, to what was described in the Android section previously. There are some serious forensic implications imposed by the BlackBerry Android version that will be discussed in Chapter 15.

Photographing the Evidence at the Scene

Photographing a device prior to the physical collection can be as important as documenting what is found at the scene. When a mobile device is located, and it has been determined that the device will be seized as evidence, it should be photographed in place as it was found.

Before taking a photograph, assign an evidence number to the device and transfer that to an agency evidence tag, which is placed next to the device to be photographed and seized. The photograph should include the device along with the evidence number. The photographer should work around the device, shooting at all angles and including as much detail as possible.

Photographing the device in place is important for many reasons. Without this information, questions as to the validity of the extracted data could arise.

- A photograph provides a visual documentation of the device exactly as it was found, including the condition of the device and its screen or keyboard, plus any noticeable scratches or other noticeable anomalies indicative of use.
- Visually documenting this information will help to dispel potential accusations that the device was destroyed or damaged by the person collecting it from the scene.
- Such documentation can also be used to determine whether the device has any evidentiary value.
- If the device is powered on, a screen saver or wallpaper might provide information of interest.
- The date and time that appears on screen is important information as well.

After the face of the device has been properly photographed, the device can be turned over so that the opposite side can be photographed, again including the evidence tag number. The back side or battery side should be photographed to document serial numbers, missing batteries, missing backs, and any damage. After the device has been photographed, it can be placed properly in an evidence bag, as documented in the following sections.

Tagging and Marking Evidence

Each piece of mobile device evidence has unique characteristics and requires specific handling procedures. Mobile device evidence can be sensitive to changes in state, and you must take care when handling, tagging, and marking the evidence. Protect the device and any other mobile-related artifacts by wearing gloves to avoid adding fingerprints. Also, by maintaining the user's finger smudge marks on the capacitive screen, you can determine a passcode or phrase later to unlock a mobile device that uses a swipe or pattern for security. Maintaining the integrity of the user's marks on the device could be extremely important.

Before marking, tagging, and bagging memory cards, you must be grounded electrically to avoid sending electrostatic discharges (ESDs) onto the card. Memory cards and SIM conductors are highly susceptible to damage from ESD. A small static charge transferred from a person to an object can deliver from 3000 to 25,000 volts. A memory card's flash and SIM conductors that operate in the range of 1.8 to 5.0 volts can be corrupted and even destroyed by as little as 30 volts. Considering the high voltage that can be delivered in the process of identifying and bagging memory and SIM cards, anyone collecting the artifacts must take proper precautions.

When tagging and marking mobile device evidence, identify each piece with a unique number. The numbering should include the current year, the department case number assigned to the current event, a location number, and an article number. Maintaining this consistency when collecting evidence to be examined later may seem like overkill, but if the same information is collected as part of the organizational policy, this will be second nature. When an organization follows the same policy and procedure for tagging and marking evidence, every person involved will understand the procedure if requested to assist or take over the assignment. Also, days, months, or even years later, anyone examining the evidence will be able to refer to the documentation for a clear understanding of the event and where the artifact was seized.

The evidence numbering system is straightforward. For example, suppose a mobile device was recovered in a bedroom during an execution of a search warrant in 2018. The bedroom had been identified as room 2 of 5 and documented as such during the scene assessment. The departmental report number assigned to the event is 555123. The tag and marking for the piece of evidence would look like this: 2018-555123-2-1. The final 1 indicates the number assigned to this piece of evidence. Using this format clearly defines all the appropriate information about the evidence.

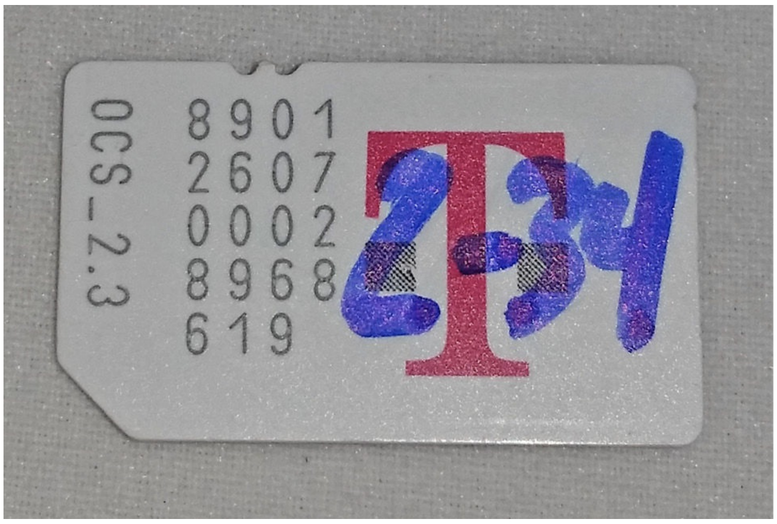
A tag or label should be affixed to every piece of evidence prior to it being placed into an evidence container, which will contain the evidence number as previously described. If an artifact cannot be tagged or labeled, it should be marked. The tag or label should indicate the following:

- Date and time
- Collector's name/identifier (such as badge or employee number)
- Evidence number (year-case-location-artifact)

- Description of artifact following the appropriate guidelines

You may need to mark the evidence directly if the artifact is not large enough to affix a label or tag to. In these instances, the following guidelines can help you ensure that these artifacts are properly documented:

- Mark the evidence location and number. In the example "2-34," shown next, the 2 indicates location 2 and 34 is the artifact number. The marking should not cover any unique identifiers (such as the ICCID on a card).



- If marking is made directly on the artifact, document why it was necessary to mark the evidence.
- If the artifact is placed into a container, affix the label or tag to the container.

Documenting the Evidence at the Scene

The documentation of the seizure of a mobile device and any related materials should be very clear and interpretable by anyone in the chain of custody. It can be difficult to keep track of many small items, such as SIM and memory cards, in an evidence bag if the evidence is placed loosely without regard into the bag without accompanying documentation. Instead, every item seized should be documented on a prepared form that includes not only descriptions of the evidence but also where each object was located on the scene. Furthermore, each article must be described so that it will not be confused with another piece of evidence.

Describing evidence can be challenging, not only because of the volatility of the data and current status of the device but also because certain types of mobile evidence often do not have discernible features. Detailing any unique attributes will allow for successful identification and can be matched to the completed property/evidence sheet that documents all artifacts seized from the location. Being able to match the item to the property/evidence sheet will be helpful if an item is somehow separated from the evidence container.

The property/evidence sheet should list all items seized from the location for the particular case. The minimum details that should be listed on the property/evidence sheet should include but not be limited to the following:

- **Item number** This is a unique value assigned to the seized property.

- **Quantity** This will indicate the number of items for a single item type (for example, micro USB cables, chargers, and so on).
- **Property description** This should include serial numbers, markings, and so on (including serial numbers, markings, and so on).
- **Owner**
- **Location found**

Following are some suggestions for documenting various items of mobile device evidence.

Mobile Device

For a mobile device, the documentation should include, but not be limited to, the serial number, make, model, color, size, condition, and the telecom company. The status of the device should also be indicated—that is, whether it is powered on or off. If the device is powered on, does it use cellular service or Wi-Fi? If there is a network connection of some type, it is critical that you quickly isolate the device from the network. Also, take care in retrieving the serial number, since most device serial numbers are printed on a label in the battery compartment area. Obtaining the serial number could involve removing a battery and powering off the device, which in turn could lock the device.

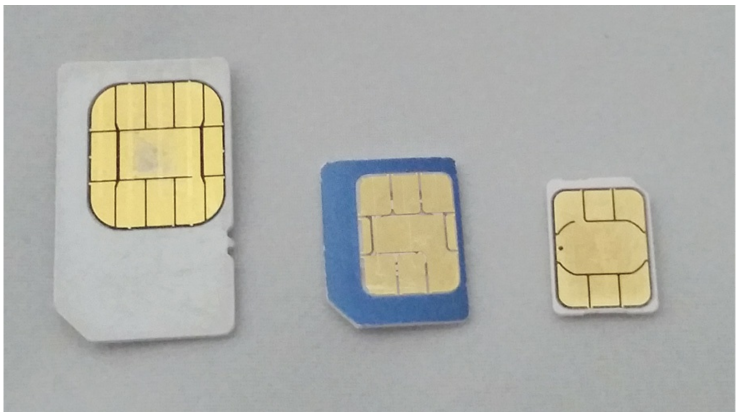
If the device is already powered off, does it contain a SIM card or memory card? If cards are inserted, document only that they are inserted into the device along with any discernible serial numbers. It is not recommended that you remove the SIM or memory card from the device while at the scene unless you're conducting an on-scene collection.

Mobile Device Accessories

The documentation of all items associated with the mobile device can be important should an accessory be needed to charge or connect the device to the hardware while conducting the forensic exam. Document the number of power cables, USB cables, and cases that match the seized device. If the accessory is connected to the device, photograph it prior to disconnecting, and label the connections. This is helpful if the connection will need to be re-created at some point during a detailed exam. This can also help the examiner determine which charger or cable is assigned to each device, especially when multiple devices of the same type are seized.

SIM Cards

Multiple SIM cards on the scene may not be inserted into the actual mobile device, but they could contain valuable information and should be seized and documented along with the SIM card inserted into the device. The location of each card should be indicated along with its integrated circuit card identifier (ICCID) number, type (standard, micro, nano, as shown next), color, condition, and the name of the telecom.

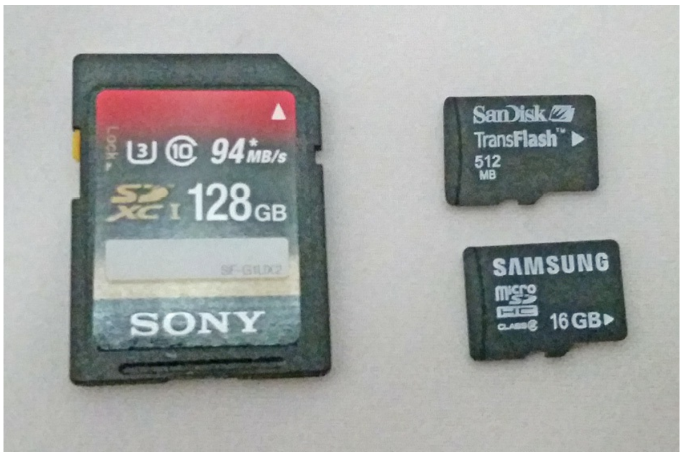


Note

The ICCID is the serial number of the SIM card, a unique number assigned to a single SIM card. Using this number can help you positively identify the SIM. The ICCID is discussed in Chapter 11.

Memory Cards

Identifying and documenting a memory card can be difficult, because the serial numbers are typically not located on the exterior of the card. Memory cards, like SIM cards, can contain an extremely large amount of stored data and all should be collected, not only the cards installed within the mobile device. The location of the card at the scene should be documented along with the size (512MB, 32GB, and so on), type, color, condition, and brand, as shown next. Any numbering on the exterior should be documented. If multiple identical memory cards are collected, each card should be labeled and marked appropriately so they can be uniquely identifiable.



Dealing with Power Issues: The Device State

After you locate a mobile device, it's critical that you determine whether the device is powered on or off and whether or not the device is password protected. If the device is not locked and does not currently have security enabled, it should be processed immediately by a trained examiner. If the device is powered on and the password is known, do not attempt to enter the password to confirm while at the scene. In both instances, the necessity of immediate processing ensures that information can be obtained prior to the device locking or if the password is not known.

If the device is powered on but security is enabled and the owner will not unlock the device or the password provided does not unlock the device, the device can be powered off and bagged. If the device is powered off, leave it powered off and bag and tag the device as appropriate using an anti-static bag.

If the device will remain turned on, attach a portable power source to maintain a charge. Several portable power sources enable the continuous charge of mobile devices for up to eight hours. The documentation should indicate that the device is currently powered on and that a portable power source is being used. To preserve the device integrity, you must isolate it by placing it in a shielding bag or by putting the device in airplane mode.

The device should remain isolated from the cellular network using both airplane mode and a Faraday bag (see the next section for information on bagging). Understanding the simple fact that the device is currently powered on and system functions are already running will help to alleviate concerns that data is being changed by navigating to airplane mode. If the device is powered on, you should recognize that data is constantly in flux and the changing of a simple device function will not alter the user data, only system information.

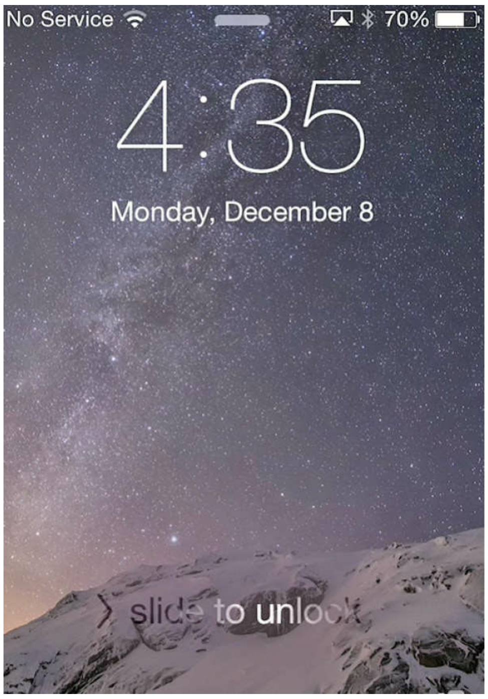
Note

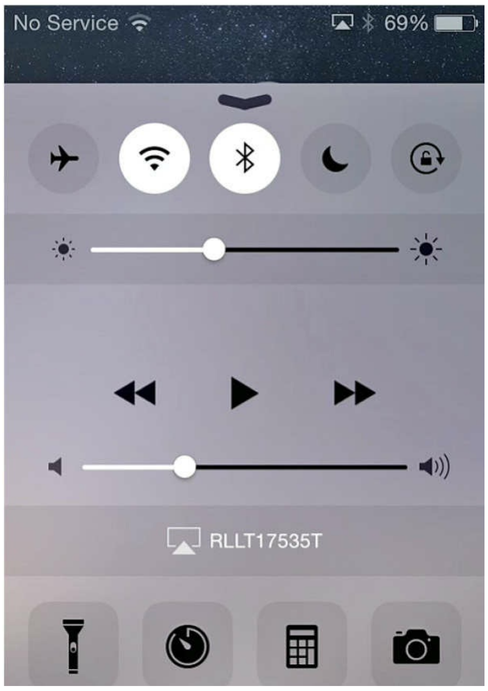
Criticism for manipulating the device by pushing buttons or pressing the screen in an attempt to locate airplane mode has been included in many documents and at conference meetings for years. However, the practice of inhibiting the device by placing it in airplane mode using the device menu is now referenced as proper procedure in materials from SANS to NIST.

Placing the device in airplane mode will deactivate the cellular signal, Wi-Fi, and Bluetooth connections. (Wi-Fi can be manually enabled while in airplane mode.) The most popular devices and the locations and methods required for placing the devices in airplane mode are listed next.

Apple iOS 7 and Later

Swipe up from the bottom of the iOS device screen depicted in the first illustration. This will expose the submenu with an airplane icon in the upper-left corner, as shown in the second illustration. If the airplane icon is not highlighted, tap it to enable airplane mode. A notification on the top of the screen will then indicate that the device is in airplane mode.

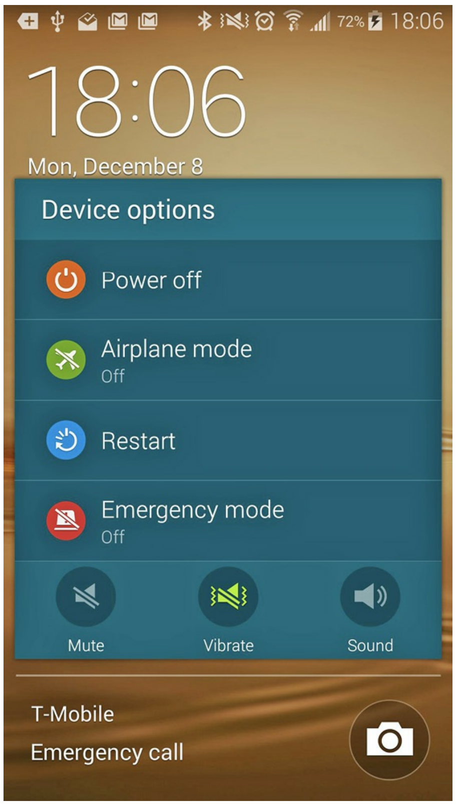


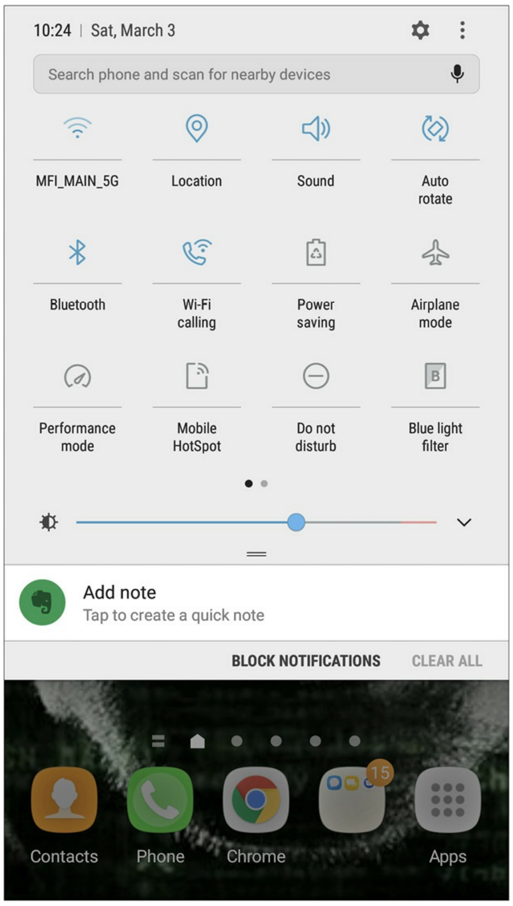


In earlier iOS versions, you access airplane mode from the main page of the Settings application. To locate the Settings application, tap the gears icon. An airplane mode toggle switch will be visible next to the label. Switch the toggle to the OFF position.

Android

Android has changed the way that users locate and toggle airplane mode on and off with updates, depending on the version of the operating system. If you're unsure of the version, first press and briefly hold the power button to bring up the power dialog. This dialog will show options for powering off the device, setting airplane mode, and other selections, depending on the OS version (as shown). Tap Airplane Mode to isolate the device from network connections. Newer Android OS devices are now similar to Apple devices regarding access to airplane mode. The examiner can simply swipe down from the top of the device, even when it's locked, and select the airplane icon.





Windows Phone 7, 8, and 10

Place a Windows Phone (7, 8) into airplane mode similar to how it's done on a laptop running Windows 8:

on the device Home screen, flick left, select Settings, and then tap Airplane Mode.

Place a Windows 10 phone into airplane mode by swiping down from the top of the device. In the Action Center window, tap Airplane Mode.

Why Leave a Device Powered On?

With mobile devices in the personal digital assistant (PDA) era, a constant charge had to be maintained or all the user information would be deleted from the device's volatile memory. The flash memory used in today's mobile devices is nonvolatile—so why should a charge on a mobile device need to be maintained when it is bagged for transport? In the years of volatile memory, all data was lost when a charge was lost. But today's mobile data may be lost for a different reason: If a security-enabled mobile device is allowed to lose its charge, any security measures employed by the user could be enabled, making access to data impossible. Maintaining a charge when a mobile device is powered on and possibly has security enabled or if the device has been unlocked temporarily by the user can be important to maintaining access to the data.

Bagging Sensitive Evidence

Place electronic devices into an evidence bag that is anti-static and protected from electrostatic discharge (ESD). All other mobile device artifacts such as memory cards and SIM cards should be placed in separate anti-static bags. These items should then be placed in appropriate evidence bags or containers specified by the agency, company, or department policy.

Plastic evidence containers or bags should not be used to encase mobile device evidence that has been packaged in anti-static bags. Cardboard or paper bags should be used instead, because sealed plastic containers can allow for humidity and condensation to build up, which could damage mobile device evidence.

Both memory and SIM cards are also susceptible to damage from the oils and acids on a person's skin. When handling these artifacts, evidence collectors should wear appropriate gloves and avoid touching the gold portions of memory cards or SIM cards.

Types of Bagging Equipment

When bagging mobile device evidence, you must place the artifacts in the appropriate type of evidence container. Electronic evidence contains data that can be destroyed or corrupted by simple exposure to static electricity, elemental factors (rain, snow, and so on), network transmission (cellular and Wi-Fi), and power concerns. The type of container used should be appropriate to the circumstances.

Use electrostatic bags or anti-static bags during the seizure of mobile device evidence. Anti-static bags are usually used for shipping computer parts and can protect the static-sensitive parts from ESD. There are different types of electrostatic bags, and it is important that you understand the differences and when each should be used.

Some bags simply dissipate electrons across the surface of the bag so the electrons do not build up. By not allowing the electrons to build up, contact charging will not occur. Contact charging can occur if two electrically charged pieces of matter rub against each other. With electronic equipment, this could pose problems with ESD. Typically, the bags that help to dissipate the charge are pink in color but can also be purchased in different colors such as blue or green. The bags should be labeled as anti-static.

Another type of bag protects from and stops an ESD, similar to how lightning rods on top of buildings protect them from damage during electrical storms. These rods are used to stop an ESD from damaging a structure by allowing current to travel safely to ground. A similar concept is used when ESD bags stop electrical current from damaging circuits or flash memory. These bags typically are slightly opaque with a mirrored or metal look because of an aluminum coating. They offer both an anti-static and ESD protection and should be used to protect evidence from static shocks when someone is handling the evidence both on the scene and in the lab.

Signal isolation bags, commonly referred to as Faraday bags, are also needed at the scene in case a device needs to be shielded from cellular signals. Several studies have been conducted on the success rate of signal isolation bags, and most show significant weaknesses in protecting a device from network signals. Most weaknesses are related to the proximity of the mobile device to a cellular base station, however. The closer a device is to a tower, the stronger the signal, and the more likely a signal will reach the mobile device even when it's inside the bag. Take care when using signal isolation bags, and test them prior to deployment and use in the field.

Properly Bagging Mobile Device Evidence

Properly securing a mobile device will help to maintain the integrity of the follow-up examination and later presentation of the facts. A mobile device contains various points of vulnerability or compromise. Most modern mobile devices contain headphone jacks, memory card compartment covers, camera lenses, and toggle switches. The state in which the device was found and seized should be maintained and documented. To maintain the state in which the device was found, you must do several things during the bagging process.

All the various settings of the external switches, cameras, and any other exterior observations should be documented and photographed. Then all evidentiary items should be protected using the following methods before covering the items with evidence tape and marking the tape with the collector's initials:

- **Exterior switches** Some mobile devices have exterior toggle switches that turn the sound on and off or up or down. Cover these switches with evidence tape to maintain the position at the time of seizure. If a device must be processed for latent fingerprints, document and photograph the position of these switches.
- **USB port** Cover any exterior ports with evidence tape.
- **Headphone port** Headphone jacks can be used to attach ear buds or a data reader for point of sale (POS) scanners and other commerce tools. Cover this port with evidence tape.
- **Camera lens** Cover all lenses with tape to prevent them from capturing any pictures or video after device seizure.
- **Battery compartment** Access to the battery area would allow access to the SIM card and memory card. Cover the compartment with evidence tape.

Once you've taken steps to ensure the security of the mobile device, place it into the appropriate evidence bag. Then mark the exterior of the evidence bag or containers with the following information, which is similar to the information you placed onto the tags and/or labels on the actual piece of evidence:

- Case/incident identifier
- Item number
- Date
- Location of collection
- Item description
- Serial number
- Collector's information (name and identifier)
- Where is the evidence going? (lab or storage)
- Comments (immediately process, phone on and charging, biohazard, and so on)

Affix the label to the outside of the evidence bag or container, and then seal it properly with evidence tape. The collector should initial the evidence tape to ensure that any tampering will be obvious.

Transporting Mobile Device Evidence

The transportation of mobile device evidence can be just as important as proper seizure, bagging, and tagging the artifacts. Be aware of the following when transporting mobile device evidence:

- Avoid placing evidence in an area that had previously transported materials that contained caustic liquids or other wet materials.
- Mobile device evidence can be susceptible to shock and vibration, so be sure that the evidence is secured prior to transport.
- Mobile device evidence can be damaged by electrostatic discharges and magnetic fields produced by speakers, radios, and large electronics mounted using magnets. Electronic evidence should not be transported or stored in close proximity to these devices. It should be transported either in an enclosure designed to resist ESD and magnetic fields or stored in a location away from devices known to produce electrostatic discharges.
- Extreme temperatures can damage mobile device evidence. Avoid prolonged exposure to extreme temperatures.

> **Caution**
>
> Placing any mobile device evidence into the trunk of a patrol vehicle where the radio transmitter and receiver are located can be disastrous. The police radio transmitter can create an electromagnetic field that can destroy the contents of a mobile device.

To Storage

When transporting evidence to be stored for a prolonged period, notify the evidence custodian of any mobile device evidence that might be susceptible to data loss or security measures. For any evidence that needs immediate attention, instruct the evidence custodian to notify the responsible party who will conduct the follow-up on the mobile device evidence.

A property/evidence sheet should have already been completed, listing all of the property that is being transported. During the transfer of the evidence to the storage facility, all evidence containers should be inventoried and matched with the proper property/evidence sheet.

The delivery should be documented, indicating the transportation of the evidence and any problems noted upon inspection after arrival. The documentation should also include to whom the delivery was made, the facility to which the evidence was delivered, the date and time of the delivery, and any other pertinent information.

The mobile device evidence should be stored in the appropriate manner. Ensure that the evidence is not exposed to extreme temperatures, humidity, ESD, and/or magnetic fields.

To the Lab

When mobile evidence must be immediately processed for trace evidence, latent evidence, and forensic digital evidence, the articles will be delivered directly to a forensics lab. It is extremely important that any evidence transported and released to the lab be documented, including details on the evidence transportation and any problems encountered upon inspection after arrival. The person receiving the evidence, along with the time and date and reason for the release of property, should be documented on the exterior of the evidence bag or container, the property/evidence sheet, and any supplementary documentation.

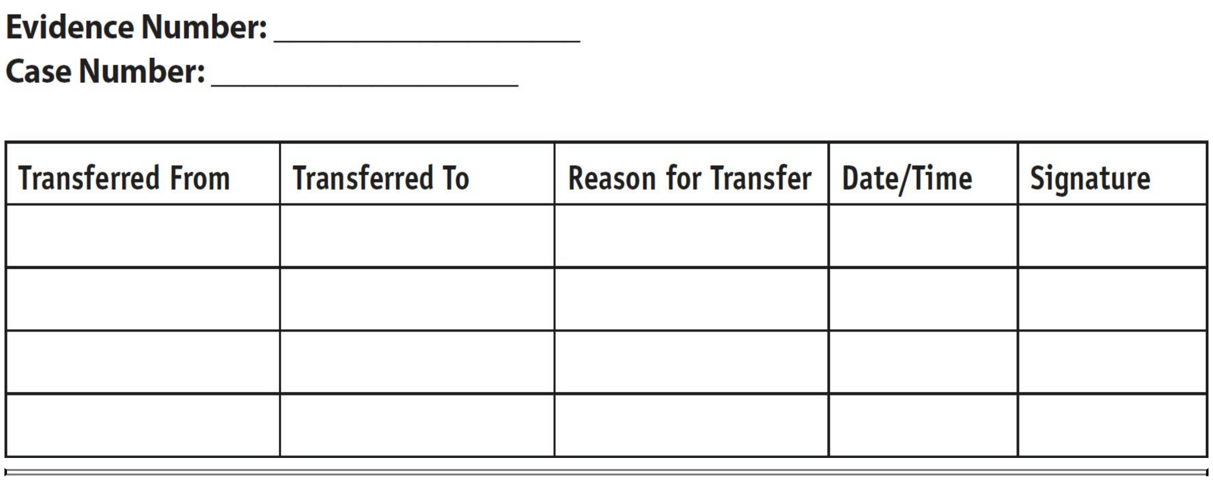
Upon reception of evidence, laboratory staff should store it in an appropriate manner to ensure the integrity of the articles.

Establishing Chain of Custody

As discussed in Chapter 5, the chain of custody document clearly details, in chronological order, every hand that has been in contact with the piece of evidence, from the seizure to the disposition. Mobile device evidence may have been handled by many different people, from the scene where the artifact was collected and bagged, to the storage area, and to the lab, where the article is analyzed. Documenting this information can be easily accomplished if procedures are in place at the onset.

Requiring all persons in contact with mobile device evidence to identify themselves on a chain of custody form will make the documentation life cycle much more manageable and inherently invulnerable to later challenge. Failing to document the chain of custody properly for each piece of mobile device evidence can have devastating results if later that evidence is used as a critical piece of the investigation.

Using a simple chain of custody form on each piece of evidence packaging will help to satisfy this prerequisite. Evidence cardboard boxes and bags that contain a preprinted chain of custody form are used by many departments and agencies. If these types of evidence containers are unavailable, using the simple form shown in Figure 6-3 can satisfy this requirement.



Evidence Number: ____________________
Case Number: ____________________

Transferred From	Transferred To	Reason for Transfer	Date/Time	Signature

FIGURE 6-3 Sample chain of custody form

Chapter Summary

Before the collection of the data from mobile device evidence, all the appropriate physical artifacts from the scene must be properly packaged, marked, tagged, and bagged. The mobile device, the device state, and device security must be considered and clearly documented. Knowing whether or not the device will have to be immediately examined and collected is determined by the current state of the device.

Outside environmental influences, including network communication, weather, magnetic fields, and electrostatic discharges, must also be considered throughout the evidence life cycle. Improper handling or transportation could lead to the destruction of evidence contained on mobile devices.

Before conducting a forensic collection of a mobile device and any accompanying artifacts, you must correctly document the custody of the artifact. The process of evidence handling is often attacked during both criminal and civil trials because of poor chain of custody documentation. For every piece of mobile device evidence, a completed chain of custody form is required to ensure that every person who comes into contact with the evidence is documented.

7 Toolbox Forensics: Multiple-Tool Approach

After the mobile device evidence has been successfully seized, tagged, and packaged, the next steps involve preserving, processing, and analyzing the data stored on the devices. As discussed in preceding chapters, mobile forensics is different from computer forensics in many ways, and one significant difference has to do with how the devices are processed during examination. With mobile device forensics, using only one tool to process and analyze the evidence can be a detrimental task, although a computer forensics examiner can sometimes get by with using only a single tool. This poses problems for a lot of mobile device investigators for several reasons—in particular, the cost involved in using several software solutions. Mobile device forensic software does not come cheap. Although open source solutions are available, one solution will not provide all the tools necessary to accommodate every type of device, every type of operating system, or all different types of stored user data. This is a limitation of all mobile forensic solutions because of one simple fact: no one tool can process and analyze all mobile devices.

This chapter covers the various reasons you'll need to use a multiple-tool approach to the examination and analysis of a mobile device. To choose the best tools, you need to know not only what a tool is doing to the extracted data but also what methods are used to validate the tool, the collection, and the process. As a forensic examiner, you will encounter a variety of pitfalls, whether you are trying to use a single tool or multiple tools. Your understanding and preparation for these challenges will help reduce criticism of either approach and will help you improve your expertise in the mobile forensic field.

Preparing for the challenge of collecting digital data from a mobile device is based primarily on the methods and tools that you use. Of course, most examiners are not capable of financially supporting each forensic solution currently available on the market. With that in mind, you must be knowledgeable of what your current software is capable of and its limitations. More importantly, you must research the available software and purchase the most competent software that will handle the majority of devices you will encounter. Research the types of devices, operating systems, and available and appropriate technologies; armed with this information, you can make the right decision regarding the best tools to use.

Choosing the Right Tools

Your decision as to which tool to use in an examination can be affected by a variety of factors, including price, ease of use, and, more often than not, using what "everyone else" uses. Each of these reasons has its own merits. Studying a tool and using sound judgment can make your decision much easier.

Commercial mobile forensic tools are some of the most expensive software examination tools available. Although their prices vary significantly, what is included in each is not necessarily related to the price—in other words, the manufacturer's suggested retail price does not correlate to what is actually included with the solution. Commercial tool prices range from $500 to well over $20,000 for a single license for a single year. These prices may then be reduced to approximately 30 to 40 percent for yearly updates. Keeping up with the technology of mobile device forensics is costly, but with wise research on the tools for your mobile forensic toolbox, you can justify the expensiveness of some tools. The astronomical prices of some software solutions on the market are often attributed to the fact that these companies think the money they've invested in research should be passed on to the consumer. Because of the pricing disparity and the variety of technologies available, it is extremely important that you conduct diligent research on each tool prior to purchase.

If some examinations and collections will be conducted by first responders or persons who have not been formally trained on forensic practices for the collection of digital evidence, it is important that you look at the tool's ease of use along with how it maintains forensic integrity. Does the tool enable those who are not technologically savvy to perform the steps necessary to recover the required digital data from a mobile device? Maintaining forensic integrity requires a tool that packages the collected data in a format that probably cannot be tampered with or altered. This is typically completed by the software in the form of hashing. Hashing of the overall data in its current form can help to fingerprint the data collected, but not all solutions hash the data in the same way or in the same format.

Note

In the National Software Reference Library (NSRL), NIST describes hashing as a mathematical technique used to produce a unique signature of a file. Hashing a file involves a one-way computation when the file has been compressed to a fixed length of 0's and 1's into a single string. A unique number is created based upon the mathematical computation and is often referred to as the *digital fingerprint* of the file. If even a single 0 or 1 is changed within the file, the entire digital fingerprint will change.

The way and format in which the software hashes data can be very important. If the software simply hashes the extracted personal data (such as Short Message Service [SMS], contacts, or call logs) after it has been extracted into a Microsoft Excel spreadsheet, it is doubtful that this hash can be verified using another solution. If the secondary solution extracts the same data, but this solution hashes the file in which the data actually exists, the hash could be entirely different. It would make sense that additional metadata would also be in the overall file where the data had been extracted, which would create an entirely different hash.

Extracting the physical files, hashing them, and producing an overall hash value of the entire collection contained in a single file would be the best method. With this method, all the collected data is contained within a single file, and if any portion of the file is altered, the fact that the overall hash has changed could be immediately identified. Also, to check the integrity of an individual file, the container could be opened and each file's hash could be rehashed to show that the file had not been tampered with since first being collected. To verify the reliability of the collected data, each collection and subsequent analysis must be repeatable. Having tools that will allow for the creation of digital fingerprints, or hashes, of the files and overall container should be a priority when you choose a mobile forensic solution.

What types of mobile examinations are going to occur? Will you examine the actual collected device or the backup, or both? These are important questions to consider when you're selecting a mobile forensic solution, especially if only backups are going to be analyzed. Choosing a software solution that covers the collection of the mobile device when the target is always going to be a device backup obtained from a PC or network server would be frivolous and unneeded. Understand the target of the examination when you're making a determination of the types of solutions to fill your mobile forensic toolbox.

You should also look for software solutions that include support of current technologies. Look for solutions that offer the greatest support for not only the collection of data but also for superior analysis of the underlying structures. Today's devices contain storage databases and configuration files that are loaded with digital information that is often overlooked by software solutions. Choose a tool that can not only collect but also digest and analyze the critical files found within the newest smart devices.

Remember that different parts of the world use different types of devices, different parts of a geographic region use different types of devices, and different areas of a company or agency use different types of devices. Tailor your mobile solution toolbox to the devices and, more importantly, the types of devices that are appropriate to your area. If a region deals with Nokia legacy devices exclusively or BlackBerry devices exclusively, don't choose a tool that specializes in iOS or Android devices. If you're conducting examinations in Europe, don't purchase a tool that is most appropriate for Code Division Multiple Access (CDMA) mobile devices, which are used primarily in the United States and in some Asian countries.

Consider the following facts as you determine the best tool to use in examinations of today's mobile devices:

- More than 58 percent of users globally in 2017 used a smart device.
- In North America in 2017, most forensic labs deal with Android and iOS devices in almost 96 percent of their digital investigations.

Supporting global mobile device statistics should not influence your decision as to what type of mobile forensic software you select if these statistics are not consistent with what you will deal with locally.

Using the multiple-tool approach can be limited at the analysis phase simply because various software products do not allow for the analysis of another tool's data. Make sure that the software you use to conduct data analysis can import various types of mobile device data from multiple tools. The ability to import multiple image types will allow for validation of the collected data and will also provide a complement to another tool. This feature will help you uncover even more information that may have been missed by another tool.

Finally, one thing that should *not* influence your decision for a competent mobile forensic solution is the fact "everyone is using it." Just because software is popular does not necessarily mean the software will be useful to all examinations in all regions. Carefully consider and investigate all facets pertaining to a localized tool set. Research the type of software and its features, interoperability, and forensic implications before making a decision as to the software that you will use for mobile device examinations. Most companies allow demonstration copies to be evaluated prior to the purchase of the software; this can be a good way to determine whether the software is right for you before you purchase.

After you've gathered forensic tools, you must properly assess them and learn to use them before conducting a live investigation. Learn about the software's operating functions, features, and limitations. Consider and seek out training on the software, which is often provided by competent companies. Without undergoing training on the tools you will use, you may end up using them in less-than-optimal ways. Moreover, because technology changes quickly, what was once a current method for conducting a collection may no longer be valid.

Note

Technology changes quickly. For example, "long ago" in mobile forensics, when Apple changed the processor in its iOS devices from an A4 (S5L8930X) to an A5 (S5L8940) chip, it was no longer possible to create a physical image of the iOS device. The exploit that had been used for devices with A4 and older chips was not valid for new iOS devices containing the newer chips. This is still de facto in mobile forensics of iOS devices today.

Analyzing Several Devices Collectively

When you're dealing with many devices and collective evidence, keep in mind that some mobile forensic tools do not allow the analysis of more than one device image or different types of media (such as computers and mass storage devices) concurrently. In these instances, you may find it better to use a conventional computer forensic solution for detailed analysis. This might not be the best conclusion, however, based upon some available mobile forensic tools. Many mobile forensic tools such as Oxygen Forensic Detective, MSAB XRY, and Cellebrite UFED 4PC can collectively analyze multiple mobile devices and mass storage devices, drones, and cloud data in a single case.

Oxygen Forensic Detective can uniquely analyze data from all aforementioned stores across multiple cases within a database. This type of feature analysis will enable you to examine all digital information and multiple devices simultaneously, which provides a very big picture of the collected evidence. Using different types of software (such as computer forensic), not just different types of mobile forensic software, can help you paint a collective picture of the events. More and more mobile devices are becoming mini computers with extended memory, storage, and capabilities. Also, more investigations of a single event will contain multiple pieces of digital devices, so the ability to analyze all of the data collectively and in a single platform will provide digital evidence synergy. Looking for a mobile forensic solution that offers a "big view" of collective data across devices will offer fantastic results!

Collective Analysis: An Example

In 2010, investigators were zeroing in on a location they believed was the epicenter for a narcotics trafficking operation. This location had been under surveillance for many months until the investigators believed they had enough information to obtain a search warrant of the premises. Believing that computers and other digital devices were on site, the investigators' search warrant included information regarding the collection of the devices and the forensic analysis of the data.

The day of the search arrived, and prior to the investigators' arrival, two vehicles departed the location. Information on the vehicles was sent to a neighboring jurisdiction, and while officers conducted a traffic stop of the vehicles, investigators initiated and executed the search warrant at the location.

The officers who stopped the vehicles communicated with the occupants, who denied any involvement with any persons at the location where the search was being conducted. A records check of both occupants in the two vehicles indicated that they had outstanding warrants, and they were placed under arrest. During the search of the subjects, officers discovered that both carried mobile devices. These devices were searched incident to arrest at the scene (this occurred prior to the *Riley v. California*

Supreme Court decision, as discussed in Chapter 5) using a mobile forensic solution.

When officers viewed the informational report, it was quickly apparent that both suspects knew one another, because the officers located contacts and SMS messages between the two suspects. In the report, no information tied them to the search location, but they were seen leaving the area prior to the search warrant execution. Lead investigators indicated that the subjects had been seen previously at the location, but they were unsure of the subjects' involvement in the narcotics operation.

Investigators arrived and conducted interviews of both vehicles' occupants, who still denied knowing anything about the location at which they were seen leaving. They were shown the report documenting the communication between them that had been collected from the mobile devices. Both indicated that they knew one another and worked together on several projects. At the conclusion of the interview, both subjects were taken to jail for their outstanding warrants. One subject paid the bond on his warrant and was out of jail within two hours.

Simultaneous to the arrests, investigators at the scene of the search warrant seized several computers and flash drives, along with several types of narcotics paraphernalia and a small amount of controlled substance. The lone occupant of the location, who was not listed on the actual lease agreement, was taken into custody on possession of a controlled substance and booked into the local jail after an unproductive interview.

The computers and flash drives were later examined by a computer forensic examiner who was assigned to the case a week after the search warrant was issued. During the forensic examination of a computer laptop, the examiner discovered several fraudulent identification documents, including driver's licenses, Social Security cards, and power bills. He also located a device backup of an iPhone with several SMS messages indicating that the "product was completed and being shipped to the house," "the tracking number is 9102 6537 1256 4987 3098 12," and "the explosives are also purchased." In this backup, the device ID was found along with a very large contact list.

Investigators contacted the U.S. Postal Service with the tracking number to determine the location from which the package was sent. The delivery location was the same place at which the search warrant was initiated, but the sender's location was approximately 2500 miles and several states away.

When a second computer was analyzed, examiners located recipes for explosives, instructions for how to make fraudulent documents, and images of a very sophisticated narcotics manufacturing system, along with images of weapons. This information helped to substantiate the belief that the location was involved in illegal activities, but investigators did not have the primary actors in this investigation—at least that's what they believed, considering the digital evidence the forensic examiner was currently reviewing.

The forensic examiner began to look into the property and noticed that no mobile devices were collected from the scene. He had found a backup of an iPhone on the computer, so he contacted the lead investigator. The investigator met with the forensic examiner and explained that two subjects had left the scene and were stopped in another jurisdiction. Their mobile devices were examined at the scene, and there was nothing to tie them to the location based upon the information in the report created by the mobile forensic software—just the fact that the two subjects had communicated with each other. The forensic examiner asked for a copy of the report.

After he reviewed the report, the examiner immediately recognized the device ID of the iPhone from the backup analyzed on the computer and contacted the lead investigator. Luckily, the device was still in jail property because the subject was still in custody and could not post bond due to the outstanding warrant. The forensic examiner obtained a search warrant for the device based upon the information obtained from the computer backup along with the information obtained in the initial collection of the device after the traffic stop. What the examiner located after conducting a full forensic analysis of the second collection with the computer data was amazing.

The actual iPhone contained several exterior images of a residence that also contained Exchangeable Image Format (EXIF) data with GPS coordinates. These coordinates, when placed into Google Maps, landed right on a house with an address that matched the shipping point identified earlier. Further examination of the iPhone revealed several chat apps that contained information implicating the user of the device in the creation of fraudulent documents found on the examined computers. The same chat app implicated the second driver as a courier for narcotics, and an image of the courier was found on the computer, indicating a fraudulent driver's license had been created for him, plus PDF documents from Dropbox that included ledger information on narcotics transactions.

Armed with this new information from the forensic examiner, the lead investigator and a tactical team from the local law enforcement agency arrived at the site revealed in the pictures on the iPhone and the tracking information from the computer. A seizure of the location was initiated, and what was found inside the residence was a testament to the value of collectively analyzing data from all digital devices. They had found not only the largest narcotics manufacturing plant in the state but also an intricate lab for manufacturing explosives.

It was determined that the iPhone owner was the leader and actually the owner of both locations. The second driver who had bonded out was not located, but an arrest warrant was issued based upon the analysis of the iPhone and the corroborating data from the computers at the scene of the first search, along with the device information obtained at the scene of the traffic stop.

If the digital investigator had not decided to look at the collective big picture, this case could have easily been dismissed.

Verifying and Validating Software

After you have determined which software to purchase and received training to use it, but prior to your first examination, you must verify and validate that the software performs as you expect. A lot of software vendors will explain exactly how the software *should* perform, but ultimately it comes down to how the software actually performs within the environment in which the examiner will be operating.

Verification and *validation* are often used synonymously in digital forensics, but they are two separate processes used in different situations to test a software solution against a predefined set of knowns. International Organization for Standardization (ISO) 9001:2015 outlines quality management system standards and defines both verification and validation separately. In both definitions, *objective evidence* is used, which ISO 9001:2015 defines as "data supporting the existence or verity of something."

- **Verification** Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled.
- **Validation** Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.

In other words, *verification* consists of determining, through testing and examining specific tasks or data, whether the specified performance of a tool is met satisfactorily. *Validation*, on the other hand, involves determining through testing that a tool, when used in a certain way, performs as intended.

Applying the concepts of verification and validation within the context of mobile forensic examinations is similar. Verification typically can be performed using a single software solution and observing whether the data that is reported matches what is stated in the guidelines and documents regarding the software. Validation typically involves using multiple tools to show whether or not the initial tool has altered the original evidence while performing a collection as defined by the software manufacturer.

Both verification and validation techniques, along with the results, can be used in other atypical ways. Sometimes these techniques can be used to determine a difference in a control sample, which could indicate compromise, while other methods might be used to determine data integrity compromise. Important to understand is that reliance on what a product developer states should be the output is not as important as what the examiner determines using a *baseline*. The only way to complete this task is by verification and validation of the software solutions in the forensic toolbox.

These baselines should be performed using the same reference device(s) when the software is first installed and configured and also after each major update.

Establishing Baselines

In science parlance, a *baseline* is a point, or a reference point, at which the experiment will be measured. Once a baseline reference point is made, any and all other experiments can be measured against the baseline to determine what changes have occurred. A baseline will shift only if it is created using a different medium, which would create an entirely different reference point.

In mobile device forensics, the creation of a baseline will allow for the verification and validation of the software. By having a reference point, the examiner can quickly identify any changes that have been made by the software to the recoverable data from the mobile device, by the mobile device operating system, or even by malicious infiltration of malware or unauthorized entities.

How the baseline is created will depend upon the desired result—verification of data changes or validation of the software. If the desired result is to identify the various data that is collected from the baseline device after changes in version, versus what the product literature indicates it will collect, a single method should be used. If the desired outcome is to determine whether user data has been altered by a forensic tool or by hostile programs or actors, another baseline should be used.

Creating a Baseline Image for Data Verification

When creating a baseline for data verification, your objective is to determine whether collected user data from the mobile device has been altered between updates to a forensic software solution or if data has been added or deleted by malicious, accidental, or hostile means. As indicated, baseline devices should be purchased and populated with data and will be used only for verification of collected data from a mobile forensic solution. If the objective of the baseline is to determine hostile, accidental, or malicious intent, the actual suspected device will need to be collected to create the baseline. Both software verification and device integrity verification should be covered. Pay specific attention to how a baseline is taken, compared, and examined for both types of device verification.

Note

"Hostile intent" could mean the installation of software to monitor the device, such as a rogue application or service, and "malicious intent" could indicate the installation of an application intended to destroy data.

Software Verification Software verification involves examining the data of a prepopulated device with the forensic software solution that will be used to perform mobile device examinations. Save the collected data in a format that can be easily analyzed and compared side-by-side to a secondary extraction. This comparison extraction should occur upon any minor and major updates to the forensic software used for examinations. The simple comparison collection should consist of all of the current user data capabilities supported by the software and should be a query of only the logical data contained on the device, not a physical collection.

Note

A "logical collection," in the context of a baseline, consists typically of only data that can be accessible by simple communication techniques; a "physical collection" uses more advanced communication that yields entire partitions from a device.

Once data has been prepopulated on the mobile device, the exemplar devices must not be altered after the baseline is created and not until the comparison collection is completed. By using this method, the examiner can determine whether changes have been made during the various updates to the forensic software to verify that the data collected is consistent with data that is within the baseline and to identify any variations to the output. Changes to software occur frequently, and using this method will assist the examiner in determining quickly whether the software conforms to the expectations of a similar collection.

Changes or differences to the collected data can be attributed to the addition of new scoped data artifacts, new capabilities, and alterations to data within an already scoped data set. An example of each of these changes is described next, along with recommendations for each when encountered:

- **New scoped data artifacts** These artifacts are additions to already defined data. This could include the addition of a status to an SMS message indicating a read or an unread message that was not indicated in the previous version, the addition of a work phone number to the contacts table, or the read date in a calendar entry. This should be documented and verified to ensure that the new addition is correct for the exemplar device. If new scoped data artifacts are located, an entirely new baseline should be created after the completion of the verification cycle.
- **New capabilities** These would include an entirely new set of data that was not in the previous release, such as the addition of memos or user applications when the previous forensic software version did not

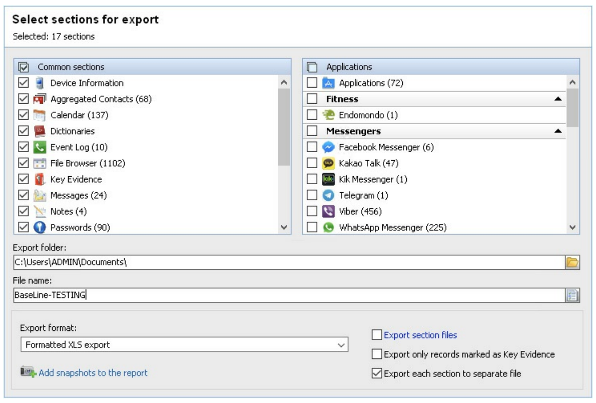
collect this app data. This addition should be documented, and each artifact in each category should be analyzed to ensure that the data conforms to the expected values. A new baseline should be created after the completion of the verification cycle.

- **Alterations to data within an already scoped data set** This alteration occurs to a value that existed previously in an earlier release. Because the actual value of the data has changed, the examiner must research whether the previous value provided by the software was incorrect or the release being evaluated is incorrect. This change is the most important to document and to understand because it can have the biggest impact on the software verification cycle. If you determine that the value was corrected in the software from a previous release, make sure the software company has documented this on release notes. A new baseline should be created using the new version after the verification process has been completed and all changes documented. If the value in the software that is being verified had been correct in previous versions and is now incorrect, immediately notify the software vendor of the inconsistencies. Then document this information and do not create a new baseline.

The baseline comparison method can be tedious. Because of the manual process involved in determining new capabilities and additional artifacts within each capability, it is important that you clearly document data contained on the exemplar devices. To obtain the needed baseline information, you should first export the logical data by data type. Then export each data type to a format that can allow for easy comparison by their hash values. If the hash value of the files is the same, then there is no additional change in values or capabilities. If the hash values do not match, this could indicate a change, and those files will have to be manually inspected to determine the changes that have occurred. If a change has occurred, create a new baseline, and the changes in the forensic software should be updated and verified in the release documentation.

Verification Workflow Example Following is the workflow for the verification of software. The tools used in this exercise are not the only tools available to accomplish this task. Mobile forensics tools such as Cellebrite UFED Physical Analyzer, MSAB XRY, Magnet AXIOM, and Oxygen Forensic Detective can all export collected data in various formats. The following procedure can be used with any of the mobile forensic tools that will allow collected data to be individually exported into single files.

For this example, the tools used are Oxygen Forensic Detective and hashdeep. Using Oxygen Forensic Detective, selected capabilities from the baseline image are exported into an Excel .xls file (see Figure 7-1). (Note that Oxygen Forensic Detective will also be discussed in Chapter 8.) Hashdeep is a free program that computes multiple types of hashes against a set of folders, files, and subfolders. It can also be used to compare a known set of values in a text file against a set of files, folders, or subfolders when performing an audit. Hashdeep was developed by Jesse Kornblum and can be downloaded from http://sourceforge.net. It is maintained at https://github.com/jessek/hashdeep.



Select sections for export
Selected: 17 sections

Common sections	Applications
✓ Device Information	☐ Applications (72)
✓ Aggregated Contacts (68)	☐ **Fitness**
✓ Calendar (137)	☐ Endomondo (1)
✓ Dictionaries	☐ **Messengers**
✓ Event Log (10)	☐ Facebook Messenger (6)
✓ File Browser (1102)	☐ Kakao Talk (47)
✓ Key Evidence	☐ Kik Messenger (1)
✓ Messages (24)	☐ Telegram (1)
✓ Notes (4)	☐ Viber (456)
✓ Passwords (90)	☐ WhatsApp Messenger (225)

Export folder: C:\Users\ADMIN\Documents\
File name: BaseLine-TESTING
Export format: Formatted XLS export
Add snapshots to the report
☐ Export section files
☐ Export only records marked as Key Evidence
✓ Export each section to separate file

FIGURE 7-1 **Oxygen Forensic Detective exports data into an Excel .xls file.**

This exercise involves comparing a baseline created with a previous version of Oxygen Forensic Detective with the current version of the software (10.1.0.54). This comparison uses the hashes created from exported files in an effort to identify changes immediately within the files. If a change is encountered, we will determine whether the change falls into the three categories discussed:

- New scoped data artifacts
- New capabilities
- Alterations to data within an already scoped data set

First, we export the data into a folder and, from the Windows command prompt, hash the contents using hashdeep with the following instructions:

```
C:\md5deep>hashdeep64 -c md5,sha1,sha256 -b C:\BaseLine-TESTING\*
```

The format for the command contains a call to the executable (`hashdeep64`), a flag (`-c`) to enable comma-separated output, the hash formats, a flag (`-b`) to enable bare mode and show only filenames of hashed files, and then the path to the folder with the files that will be hashed.

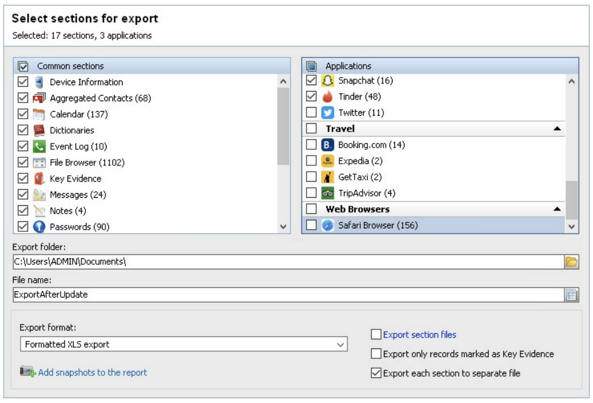
The result shows the hashdeep dialog, the result format, and the string used to invoke the application and the hashes:

```
C:\md5deep>hashdeep64 -c md5,sha1,sha256 -b C:\BaseLine-TESTING\*
%%%% HASHDEEP-1.0
%%%% size,md5,sha1,sha256,filename
## Invoked from: C:\md5deep
## C:\md5deep> hashdeep64 -c md5,sha1,sha256 -b C:\BaseLine-TESTING\*
##27648,e91455613980e2ada59ef6b38bbab27a,c34416f0a4223f59c61d254ff04af78d2
38b4c53,3f2e05a652722df0f865b0c84ec20b5e06f4601b7e39c9f5ac9d84d623f65662,
Calendar.xls
33280,1ce7e1f8d5816e2ed9bb10256e1e5b5c,cb540ee77e063b6eb3d614447cd0b3c5de9
4ea66,64b95732229aed1a861e14b5249c74841d0ab666bafe0ddb976e3740874d3460,De
vice Information.xls
3584,dfa346eaf9801d2cef2543b3d2e8abf0,a74facd934a9acea782348df78f53e883609
9d49,f70a41d9c8bf8bbca56677add89c3e13dd957a530a48e9db0abde68c19144015,Dic
tionaries.xls
10752,04f800671e40709ff028eced1925abec,dcd71d4c8d59220e9940ab9302dfd693dc4
05a6e,44e53f87959c7c9bf8b74f522993e4c62c7b7a84dfd052ce8ff61db0c83372d1,Ev
ent Log.xls
3584,26b22060028ea0254a58419f2a6d8219,10e5f9eefbb36ec6fe3757309f6cb985f667
d5b2,3fc2436b0f6af9d7969483da1345ee32c818ad809de9527b0c9265a11e86d8a0,Key
Evidence.xls
380928,ae0d9d052083775ab118a5c3f1ab6bf3,49674e699effd83ee91b09671d309a81ba
eebb8f,c60b56bbd18b3549ec6a66487efba95bd40d94afd8554017f09163b5418754ac,A
ggregated Contacts.xls
```

```
4608,551bb9be7a2498bcc305aa44ccd07f6b,5365d70aae216d86ae29144dcd07d7ca7f2
ce1b1,9b009f77944f9071476de1fdc61a1f4a317de9de8e4591c0fc717369afb6314a,No
tes.xls
58368,85ab27a0cc0616e4f2fba0cf7604a2c1,0b15507d52ea20519cc45bd996cec542873
73b7b,d3f531175a29d35a80f9dc4ab24a9665632eb9cd23b1406395db2b2d20d2078c,Pa
sswords.xls
17920,dc9625be91e64b0c80100b8c89f5a51a,d7e76ea488e03c751e716641b3cb3ae8dcc
18690,dd72c207afdc2c4451ee206fe2dc753ecc8d8cc3a1bdfd705f45c41492a12fca,Me
ssages.xls
71680,43c475af2a62b9aebbce843bc66f924b,0e5263184799b5acd39bb8fbf9d4c0a75ee
a22f7,65d7e4908e87a28cb2351ab0ee565af4d6b4fd65f55ec710a8496bc8be247808,Ph
onebook.xls
153088,6d5e4857833e30419fffeacd40881272,c0641b191ea1c467d4671f264146df9dca
39824e,e71f2a983dae86da59d37a08451fdddd0088b36daa6ec1a11b6ab1e8c9364fea,S
earch.xls
25088,d4731dba25f770796ac480cc74e9ba0a,8599cca3896cce52cd7757f9990b8622096
86a0f,5db41e59515be1bdde0a48bd5a0dda648bba36dcec7e8415ad68272b01b89f7a,Wa
tch lists.xls
10752,8605b9f87eb05ca9190e78f6d1750d47,6cf59ca3d9a6536dd097ac8caadeb54ff8f
6a431,8bc80324af48b5e066e75689ae28129629740fca1a06e9f3deec1b65c2020e34,Web
Connections.xls
10752,714802cf02d2299c1dcf3bf4f41544e8,6637ec5ab9ea0c214ce079550aaea2455d9
22233,a82e390597843e22ef551c6bf24e4f732101110099db3bc417df9eab99924cb3,We
bKit Data.xls
3088384,3f40a65db3058462c05ff7e45f287219,79333df89f165e71f4abdaee3e847319b
94ede63,b47636bd2d11bae473f6333d6f0e6b7024aea9b671fbdee1819a01000d0fb531,
Timeline.xls
13443584,3ba74639fa3b648327d9f45ac0e202e1,fa344dbf26d0c9857d363c7da0efbc9c
0455b3ba,9a0a14b27d092f071f4e0d1134785b8751f823b26c3592179622c2a6b6c63116
,File Browser.xls
```

The result, including the hashdeep dialog, is then saved to the file baseLine-knowns.txt. This file can be used to determine any differences in any image that will be obtained with the newer version of Oxygen Forensic Detective so that the values can be compared to ascertain whether changes have been made to the mobile device data's output.

Using Oxygen Forensic Detective 10.1.0.54, we make another collection on the exemplar device when an update is sent out to users. The capabilities are all exported to another .xls file (see Figure 7-2).


Select sections for export
Selected: 17 sections, 3 applications

- [✓] Common sections
- [✓] Device Information
- [✓] Aggregated Contacts (68)
- [✓] Calendar (137)
- [✓] Dictionaries
- [✓] Event Log (10)
- [✓] File Browser (1102)
- [✓] Key Evidence
- [✓] Messages (24)
- [✓] Notes (4)
- [✓] Passwords (90)

- [ ] Applications
- [✓] Snapchat (16)
- [✓] Tinder (48)
- [ ] Twitter (11)
- [ ] **Travel**
- [ ] Booking.com (14)
- [ ] Expedia (2)
- [ ] GetTaxi (2)
- [ ] TripAdvisor (4)
- [ ] **Web Browsers**
- [ ] Safari Browser (156)

Export folder: C:\Users\ADMIN\Documents\

File name: ExportAfterUpdate

Export format: Formatted XLS export

Add snapshots to the report

- [ ] Export section files
- [ ] Export only records marked as Key Evidence
- [✓] Export each section to separate file

FIGURE 7-2 **The second export of the same exemplar device after an acquisition with new version of Oxygen Forensic Detective**

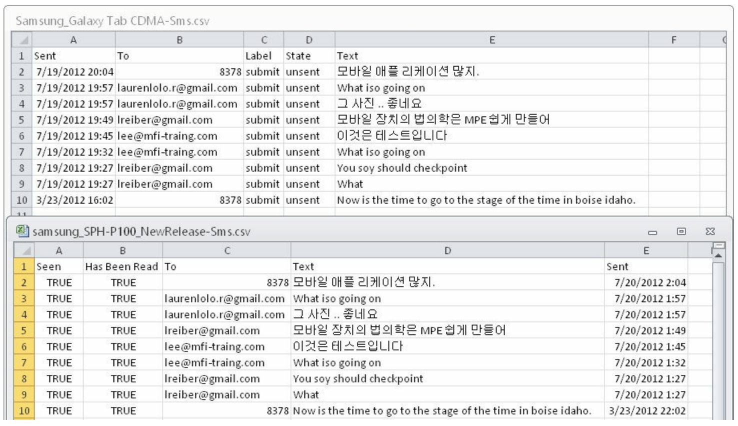
From the Windows command prompt, we use hashdeep again to compare the hashes to show those that match and those that do not match in the baseline hash file baseLine-knowns.txt. This command determines whether any files are the same based upon hashes, by using the `-m` and `-k` flags. The `-m` flag specifies that hashdeep should show only the files within the baseline that match the hashes that are within the file identified with the `-k` flag. It returns no matches, which indicates that all the files are different between releases of the forensic software:

```
C:\md5deep>hashdeep64 -m -k baseLine-knowns.txt
C:\BaseLine-TESTING\*
```

To identify which files are different based upon the known file hashes, we run another command using the `-x` and `-k` flags. The `-x` flag will identify which files within the baseline are different, and the `-k` flag identifies the file of known values. If files are returned, they can then be manually compared to the known files.

```
C:\md5deep>hashdeep64 -x -k baseLine-knowns.txt
C:\BaseLine-TESTING\*
C:\BaseLine-TESTING\Search.xls
C:\BaseLine-TESTING\Watch lists.xls
C:\BaseLine-TESTING\Webkit Data.xls
```

Several hash values have changed, and those files must be examined to determine the differences. This will help us differentiate between the types of changes that have occurred between releases in order to verify the forensic solution. The changes that are observed in Oxygen Forensic Detective has to do with new parsing algorithms for the WebKit data, new search terms used by the examiner, and additional watch list terms. Figure 7-3 shows several changes between versions of AccessData's MPE+. The large difference was a direct result of the manufacturer changing the application that queries an Android device. This information would not be known if it were not for the verification of the software. The image in Figure 7-3 depicts an additional column of data that indicates whether the SMS message has been seen or read, and also changes the format from local time to UTC time, which are both listed in the Sent column.



Samsung_Galaxy Tab CDMA-Sms.csv

	A	B	C	D	E
1	Sent	To	Label	State	Text
2	7/19/2012 20:04	8378	submit	unsent	모바일 애플 리케이션 많지.
3	7/19/2012 19:57	laurenlolo.r@gmail.com	submit	unsent	What iso going on
4	7/19/2012 19:57	laurenlolo.r@gmail.com	submit	unsent	그 사진 .. 좋네요
5	7/19/2012 19:49	lreiber@gmail.com	submit	unsent	모바일 장치의 법의학은 MPE쉽게 만들어
6	7/19/2012 19:45	lee@mfi-traing.com	submit	unsent	이것은 테스트입니다
7	7/19/2012 19:32	lee@mfi-traing.com	submit	unsent	What iso going on
8	7/19/2012 19:27	lreiber@gmail.com	submit	unsent	You soy should checkpoint
9	7/19/2012 19:27	lreiber@gmail.com	submit	unsent	What
10	3/23/2012 16:02	8378	submit	unsent	Now is the time to go to the stage of the time in boise idaho.

samsung_SPH-P100_NewRelease-Sms.csv

	A	B	C	D	E
1	Seen	Has Been Read	To	Text	Sent
2	TRUE	TRUE	8378	모바일 애플 리케이션 많지.	7/20/2012 2:04
3	TRUE	TRUE	laurenlolo.r@gmail.com	What iso going on	7/20/2012 1:57
4	TRUE	TRUE	laurenlolo.r@gmail.com	그 사진 .. 좋네요	7/20/2012 1:57
5	TRUE	TRUE	lreiber@gmail.com	모바일 장치의 법의학은 MPE쉽게 만들어	7/20/2012 1:49
6	TRUE	TRUE	lee@mfi-traing.com	이것은 테스트입니다	7/20/2012 1:45
7	TRUE	TRUE	lee@mfi-traing.com	What iso going on	7/20/2012 1:32
8	TRUE	TRUE	lreiber@gmail.com	You soy should checkpoint	7/20/2012 1:27
9	TRUE	TRUE	lreiber@gmail.com	What	7/20/2012 1:27
10	TRUE	TRUE	8378	Now is the time to go to the stage of the time in boise idaho.	3/23/2012 22:02

FIGURE 7-3 **Using Microsoft Excel, both .csv files from MPE+ 5.0 and 5.5.5 are compared, showing various differences.**

If any changes between the baseline and the secondary comparison image are detected, a new baseline should be created. In this example, a new image was created with MPE+ 5.5.5, and the files are all hashed. A subsequent extraction was completed with version 5.5.6 and the results compared using the steps previously covered with hashdeep. Here is the process with the new version:

```
C:\md5deep-4.4>hashdeep64 -c md5,sha1 -b
C:\Users\lreiber\Documents\SamsungBaseLineTest\*
%%%% HASHDEEP-1.0
%%%% size,md5,sha1,filename
## Invoked from: C:\md5deep-4.4
## C:\md5deep-4.4> hashdeep64 -c md5,sha1 -b C:\Users\lreiber\Documents\
SamsungBaseLineTest\*
##
8757,c98329f08a70bb3fb892b8e0d344a385,ebb565ac8430a3a10059cc0bf1fc9fb0ed06
66d9,samsung_SPH-P100-AndroidPackages.csv
739,17282d94f4b5d46fb79237ebe34d6389,1d462456000ecea6bfa2af2e79ff5188612c5
60d,samsung_SPH-P100-Bookmark.csv
2203,9d7e02d0a3eea068007d3b31077f0ff0,345c10e321e7514b52e11d1ecec3ad82a84c
2fc8,samsung_SPH-P100-Mms.csv
28189,2580444096d93033355cd2bf550a3792,0eb9c61d45aefae193faab283de78a3a314
a5b94,samsung_SPH-P100-Browser History.csv
895,e1e725d371b7fa47552236ead450a3a4,2bebef77324eb6e4e587a65d671029233cb7b
2e4,samsung_SPH-P100-Phone Info.csv
849,2e41f603991afab10c58e9d689c73b4e,4b8145ec8619c4615e0a859ac9d7e88734ed1
d9c,samsung_SPH-P100-Sms.csv
34172,34f054e3fe5ee65a9043ce7d114a6124,9f2f5d962cfa21efc1045966b0c3bba0228
5bf2f,samsung_SPH-P100-Phonebook.csv
194,b23190ac69277791f15cfc704a0840f7,d0078e800b464c119ded299440f0caa484c1b
950,samsung_SPH-P100-WifiHotspots.csv
```

Using the -m switch this time identifies all files that match within the original data set and the new version:

```
C:\md5deep-4.4>hashdeep64 -m -k baseLine-knowns.txt
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\*
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Bookmark.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-AndroidPackages.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Mms.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Browser History.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Phone Info.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Sms.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-WifiHotspots.csv
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\samsung_SPH-P100_
NewRelease-Phonebook.csv
```

To confirm, we check to see if any files do not match by using the -x switch. This command does not return any files:

```
C:\md5deep-4.4>hashdeep64 -x -k baseLine-knowns.txt
C:\Users\lreiber\Documents\samsung_SPH-P100_NewRelease\*
```

The result indicates that the new baseline files in MPE+ 5.5.5 are identical to the secondary image created with MPE+ 5.5.6. This proves that no changes were made to the extraction method and additional software capabilities have not been added between releases.

Device Integrity Verification The baseline steps involved in the verification of the software differ from the steps used for verification of changes to a device. The device integrity verification process does not determine whether the forensic software has made a change, but whether the device has been compromised.

Many global companies rely on mobile devices to keep their employees in touch with one another, to allow work to be completed remotely, and to keep production up—no matter where employees are located. An employee who is constantly tethered to a mobile device can instantly access files, documents, e-mail, and any other type of electronic data. With this mobility comes the price of data security, however. When you are confronted with the possibility of device compromise, you can take steps to ensure that the compromise can be recognized. You can create a baseline of the mobile device prior to its being deployed in an area where data could be compromised and later examined. This verification can identify malicious applications, injection of foreign data, and other types of espionage.

To complete this process, you must create a baseline of the device just prior to the journey. Of course, data will change in the course of the deployment, simply because the user will be using the device. But by creating a baseline for the device and its contents, you can quickly dissect information to determine whether anything has been compromised on the device. Once the device is obtained, a physical data collection should be completed. If a physical data collection cannot occur, for example, on a new iOS device, a full file system collection along with application files should be obtained. The files are needed in this type of verification because the addition or deletion of files that had previously existed or had not existed will be key to the process.

Note

Not all mobile forensic software can complete this process, but software such as Cellebrite UFED 4PC and UFED Touch Ultimate, MSAB XRY, and Oxygen Forensic Detective can create an image that can later be used to conduct a baseline analysis on most smart devices.

Once you have created a baseline for the device partition or file system, you must make an image of the device data along with hashing and indexing all the files within the created baseline image. Indexing involves categorizing, listing, filtering, and sometimes formatting the data to make it easy to search. Once all of the files in the collected image have been hashed and indexed to create each file's unique digital fingerprint, you create a *custom dictionary*, a list that contains all the file metadata along with the hash of each file in the image. Your forensic software should be able to create a dictionary containing all of the file hashes within the baseline image. Then save this dictionary along with the device baseline image to sort the known files against unknown files or alert files contained within the comparison device image once the device is back from the deployment.

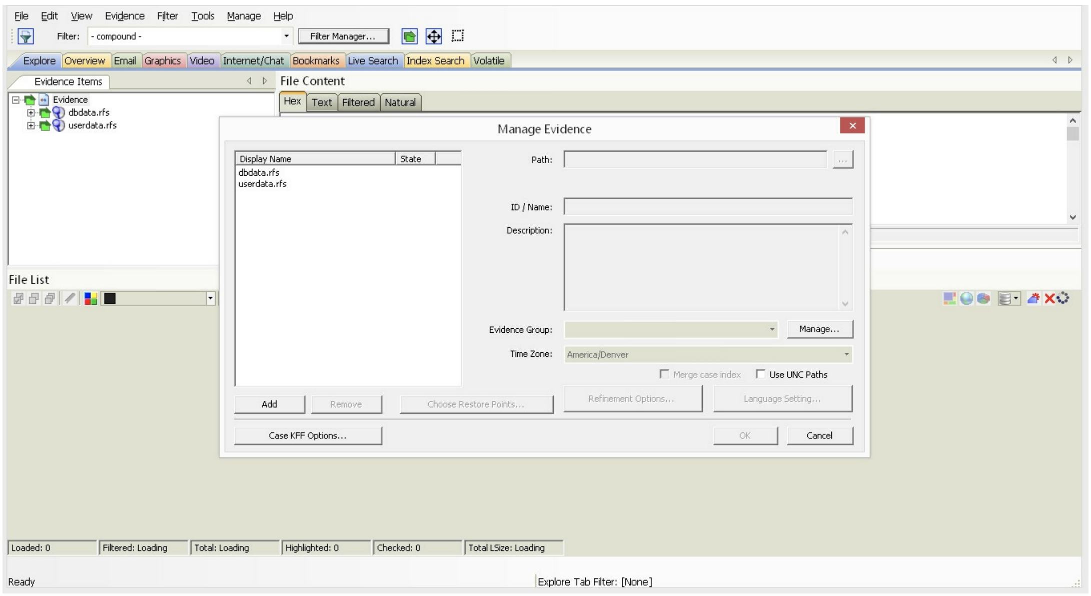
This technique is typically used by companies that are operating in different countries and by military employees or contractors. The following example information was culled from the employee travel policy of a major gas company. This policy requires that the security team take a baseline of the contents of the device and create an indexed list of the contents, which can be later compared to the contents of the device immediately when the device is returned:

> Immediately prior to the travel the mobile device designated to be used shall be forensically imaged by the device security team. The image that is created of the mobile device will be indexed and stored. … Upon return, the device will be immediately forensically imaged using the same technique used to create the baseline. The baseline dictionary will be imported and any delta analyzed immediately. Should the delta be identified as a security risk as defined in…the device should be removed from service and further response from the designated security team will be elevated based upon the reported compromise.

In today's world, entire company networks can be infiltrated with a simple rogue malicious app embedded on a user's mobile device. This technique should not be limited to travel policies, but can also be used to create a baseline of a device upon distribution to employees. If an employee is terminated, accepts a job from a competitor, or is subject to an investigation, the device data can immediately be compared to the baseline that was created when the device was distributed at hiring.

Using the following process, a device's integrity from a snapshot was examined to determine compromise, data changes, and malfeasance.

First, the images were collected using a Cellebrite UFED Touch device and then examined in AccessData Forensic Toolkit (FTK), as shown in Figure 7-4. FTK is a computer forensic product with built-in functionality for comparing known values that enables the examiner to flag files that match and do not match a known set of hashes.



Filter: - compound -

Evidence
dbdata.rfs
userdata.rfs

Display Name	State
dbdata.rfs	
userdata.rfs	

Time Zone: America/Denver

Loaded: 0 | Filtered: Loading | Total: Loading | Highlighted: 0 | Checked: 0 | Total LSize: Loading

Ready | Explore Tab Filter: [None]

FIGURE 7-4 **A physical image of the device partitions for an Android are imported as evidence into FTK.**

Note

X-Ways Forensics software can also create a set of hashes that can be used against an image created and exported by a mobile forensic tool. EnCase Forensic from Guidance Software can also allow for a set of hashes to be exported and then added to a hash database within the product. EnScripts are also available and can be used within EnCase to enable the examiner to export all files that either match or do not match a known hash within the hash library.

Next, a complete file list was created that contains all of the file hash values, which can be stored in a file and imported into a file database of known file hashes—in FTK, this is called the Known File Filter (KFF). The KFF stored the file hashes and used them to discern the known files from the unknown files when a second image of the device was created and imported into FTK. With this information, the examiner quickly identified which files had been altered and immediately began the examination into the cause of the change to the files.

After the baseline was created, the device was taken outside of the country for approximately three days, where it was used on various networks. The following process was used to determine whether data was altered while the device was out of the country.

1. A file list was created (by choosing File | Export File List Info, as shown in Figure 7-5) with the added evidence using the function within FTK to export file list information. Once exported, it was saved to a .csv file, which was modified to contain only the MD5, SHA-1, and SHA-256 columns.

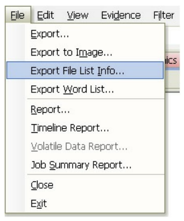


FIGURE 7-5 **Using FTK to export a file list to a .csv file**

2. This created a listing of 52,456 file hashes. The hashes were exported and modified, and then the user imported them into a file list to be used by FTK's KFF. (This file list can also be used in other tools such as EnCase and X-Ways Forensics.)
3. A secondary image of the same device was collected using the same software—in this case, Cellebrite UFED Touch. The same partitions used to create the known hashes were used—the userdata partition and the dbdata partition from a Samsung Fascinate. These partitions were again added as evidence into the FTK case.
4. The evidence was processed. Once complete, using the Filter Manager, all files that were identified as KFF Ignorable could be hidden. (KFF Ignorable equates to the files that are identical in hash from the baseline list.) After the files were hidden, only the files that were altered or created sometime after creating the baseline image were shown.

In this exercise, the baseline image contained 52,456 files, and all were hashed, creating a list that could be used to narrow down the number of files that needed to be examined to 64. The 64 files were examined for signs of malware or illegal access. The results were negative.

This technique can prove very valuable to a company's security. Using a mobile forensic tool first to collect the baseline image and then analyze with another tool that typically is used for computer forensics can be the only options when performing a similar device integrity verification exercise. Mobile forensic tools currently do not offer the examiner an option to filter user-defined values immediately, but computer forensic solutions such as FTK, EnCase, and X-Ways do offer this useful baselining feature.

Validating a Mobile Forensic Tool Using a Baseline

To validate a mobile forensic tool, you must first recognize that a second mobile forensic tool will be needed to perform this correctly. A single solution cannot be validated against itself, and if only a single solution is being used to perform all of the mobile forensic duties, then using the verification processes outlined previously will suffice.

To perform a proper validation of the software, you must employ a device that is used only as a mobile device baseline exemplar. Using actual evidence to perform the testing could pose problems should the software being validated fail testing. Having device exemplars of the most prevalent devices that are examined in the region will be the best use of the validation principles.

The object of the validation is to determine whether the software is modifying the device data upon extraction. It's important to understand that data on a mobile device will invariably change because communication continually interfaces with an active mobile device. By using one mobile device forensic application to create the baseline, you can use another application to test for possible data corruption. Perform these tests before using any mobile forensic software in an examination and also upon each major release update.

Note

As you know, a mobile device cannot be inhibited from being written to simply by employing a typical write-protection device. Furthermore, mobile forensics software interfaces with the device via protocols used to debug, back up, query, and manipulate the data on the device, changing the data in some way. Also, because the device is powered on, working processes such as system maintenance, system clocks, and other processes will continue to operate. It is important that you not get caught up in the changes to the various system processes but focus on the user data and user applications on the device.

To perform the validation of the software, you must collect a physical or file system image. Simply collecting the logical user data will not allow you to correlate and compare the two tools. You should also use an intermediary tool, as described earlier in the verification of software data. This tool should allow for the indexing, categorization, and hashing of each file in the mobile device partition or file system, similar to what was discussed earlier in the chapter. Then you can use this information to create a baseline dictionary of files, using their hashes, which existed on the mobile device at the time of collection. Next, you use the validated tool on the exemplar mobile device to conduct a logical extraction, selecting all capabilities that are supported for a forensic extraction. At the conclusion of the data collection on the mobile device, reacquire the mobile device with the initial mobile forensic software and recollect physical or file system data. It is important that you perform the same type of collection you used during the first extraction. You can compare this image to the initial data collection by using the dictionary created with the baseline data collection. If data has been added, deleted, or modified by the tool being validated, these files will be immediately identified upon completion of the dictionary comparison.

You must examine each file that diverges from the baseline image. These will undoubtedly include system files, which will commonly change because of the active state, and software protocols and changes the examiner may have made to complete a data collection of the device (such as placing an Android device into Android Debugging Bridge mode). Any files that contain user data should be scoured. The forensic software should not alter content in files that contain user data, such as SMS, contacts, call logs, browsing history, user applications, and so on.

If you determine that the mobile forensic software has modified a critical piece of user data, you must further examine the file to determine if the integrity of the user data has been compromised. At times, forensic software can alter a file's metadata during the extraction, which could change the overall hash value, but the data contained within the file has not been altered. If this is the case, although the user data has not been compromised, you should have a clear understanding as to why the file metadata was changed. If at any time you determine that the user data other than the metadata contained within the file has been altered, notify the forensic software company and discontinue use of the software until it is updated to address the problem.

Validation Workflow Example The following example shows a complete workflow using several products to address this type of software validation. This workflow was completed using Cellebrite UFED Touch, FTK, and Oxygen Forensic Detective.

1. Using the technique described earlier in the section "Device Integrity Verification," we created an image using a single forensic solution—in this case, Oxygen Forensic Detective. This was the baseline image—a physical image or image that contains a device file system.
2. We created a hash list of the files using FTK, as outlined earlier.
3. Using the software to be validated (UFED Touch) and the same device on which the baseline was just created, we conducted a full extraction of the device (this could be logically or physically or both).
4. After the collection was completed by the UFED Touch of the exemplar device, we performed another collection using Oxygen Forensic Detective, as with the initial baseline in step 1.
5. We compared the files obtained from the collection by UFED Touch in step 4 to the hashes obtained in step 2 in FTK.
6. We analyzed any files that were different as identified by the hash to determine whether they were a system file or a file that contained user data, which could have been altered by the software being validated. If any new files were identified that were not in the baseline image, we closely inspected the files to identify whether they were collected from the device or added by the software. The software should never add a file that was not extracted from the mobile device. If this occurs, the software should not be used until verified by the vendor software developer.

In this example, the UFED Touch did not add files upon the extraction, but six files showed different hash values. Inspection of the files showed they were system files, with no adverse changes to user data. This test validated the UFED Touch.

Using Multiple Tools to Your Advantage

This chapter described the ways that you can use one software application to complement another for verification and validation. Using multiple tools to perform this very important task is essential to ensure the validity of each data collection event and of the overall investigation. Using multiple tools is not just for verification and validation, however, but also for the actual analysis of data. This analysis typically cannot be performed using a single solution for many of the reasons already discussed, and most in the industry agree that there is no single solution that will perform every function an examiner will request or want.

The ultimate goal of any digital investigation should be to uncover and make visible all data available on the device should that data be desired during an investigation. If the data exists on the device and it can be recovered, that data should be presented in some way for further analysis. Simply using a single tool will not suffice because of the inherent limitations known to exist in every mobile forensic solution. In many cases, digital artifacts are easily seen in the data within the forensic tool, but the information cannot be visualized into a report or clearly represented in a way that could be discernible to an average observer. Using a secondary tool to support the analysis and display of the data in a form that can be evaluated and used to convey the investigative message is the ultimate goal of any digital investigation. The actual case example described in the sidebar shows the value in this type of collective tool analysis.

Case Study: Working the Case Using Multiple Tools

A law enforcement agency was involved in the investigation of a homicide that had occurred in a large city—a male had been shot and killed at an unknown residence. The responding officers knew very little since the deceased was found in a neighborhood park by a person walking his dog. A canvas of the area did not assist in gathering leads to the crime, but investigators identified the deceased as a gang member and began questioning known associates. After these people were questioned, limited additional information was collected, which is typical in investigations of homicides when a gang member is involved. Generally, people do not want to get involved for fear of being the next target. A break in the case came when one person came forward and identified a possible witness to the crime. Investigators interviewed the witness, who indicated that the homicide occurred because of an unpaid drug debt and provided the name of one of the people allegedly involved.

Police responded to the potential suspect's residence and began to interview the person they believed was involved in the homicide two days prior. During the interview, the person revealed enough information that he was taken into custody and transported to the detective annex to be further interviewed. A mobile phone was also in the suspect's possession, and this device was taken as evidence. A search warrant for the device contents was compiled, issued, and served.

A device physical image was obtained using Cellebrite UFED. The mobile device data was further analyzed in Cellebrite Physical Analyzer software, which revealed that the subject in custody was involved in the homicide—but, more importantly, another subject was identified based upon text messages that had been sent from the phone. No messages were received by that number on the subject's phone, however. Investigators were unable to locate a carrier associated with the phone number to which the messages were sent, but they did have a name and possible address. Officers responded to the location and took a second subject into custody. This person also had a smart device in his possession, which was seized and ultimately examined after completing a search warrant for the contents based upon probable cause to believe this person was involved in the homicide.

Officers examined the device and collected it with the same software used for the first device. What was puzzling to the officers conducting the examination was the fact that the phone number that was given to them by the first suspect and the one the message had been sent to was not the same phone number for this device. Officers believed that the subject had another device or had purchased a new device. Both of these theories were proved incorrect once they began examining the SMS messages, however. They found that messages were listed with date and time information referencing the time before, during, and after the date of the homicide. The text messages clearly indicated that the subject in custody was the person receiving the messages, which was evident by the use of his name in the conversation.

The mobile forensic examiners had multiple tools and subscribed to the multiple-tool approach. The image of the second device was imported into Oxygen Forensic Analyst in an attempt to uncover information not readily available in Physical Analyzer. Running simple search strings in Oxygen Forensic

Analyst for the text message they believed should be on the device as indicated by the original suspect, they immediately found the message within an application, Google Voice. Inside the Google Voice settings, they found the phone number that the first suspect was referencing and the number to which his text message had been sent. It was starting to make sense. Using a Google Voice number, the suspect was not just receiving voice calls but also was receiving and sending text messages. Using additional tools and functions not in the Physical Analyzer, the examiners were able to rebuild the message from the device, indicating the second person's involvement in the homicide, along with two others; the location where the victim had been killed; the motive of the crime; and geomapping evidence that placed the device in the area at the time of the crime.

Had it not been for the usage of Oxygen Forensic Analyst to complement the Cellebrite Physical Analyzer, the data that ultimately solved the case might not have been uncovered. The use of multiple tools in this instance was priceless.

The use of a single solution to handle all mobile devices and contingencies is something that most software vendors realize is not possible. Most practitioners also understand this and believe that more than one tool is needed, as indicated in the case study. What can be clearly observed, however, is that very few examiners actually use multiple tools.

Why is there no single solution that can cover every contingency? The many reasons involve too many mobile devices, too many mobile applications, lack of a specific analysis feature, missing analysis of a specific application, or lack of support for a specific file type. The reasons are many, the solutions are few, but what is clear is that no single tool can rule them all. For this reason, the word that should be used in mobile forensics is *complementing*. To complement one another is to "bring to perfection," as described in one definition of the verb. When you use a secondary tool to perform analysis of data that works differently from the imaging tool, the solutions truly complement one another and the investigation profits.

As outlined in the actual law enforcement case, the use of multiple tools can be a game changer. Adhering to the philosophy that a single tool is all that's needed is a recipe for disaster.

Dealing with Challenges

Most challenges to the examination of a mobile device will not come from *what* was collected from the device, but from *how* the data was collected. Also challenged will be the software used to collect the data. Questions abound about whether the tool wrote data to the device, whether the tool deleted data from the device, and whether the tool merged data from the previous collection into the current device's data. Also, a lot of challenges can come from questions about the inability of one tool to do one thing while another tool can accomplish the task.

Understanding the many ways in which the processes described in this chapter can be challenged will help you prepare for better examination techniques and for handling these common objections in court. What is quite evident when observing most individuals conducting mobile forensic exams is the fact that few examiners comprehend how important these challenges can be to their cases. Moreover, some current examiners believe that multiple tools are only for validation of the tools and fail to recognize that using multiple tools increases the chance of uncovering additional data from the device.

Overcoming challenges in verification and validation, along with the use of both single-tool and multiple-tool examinations, will forever be challenged.

Overcoming Challenges by Verification and Validation

The reasoning for conducting verification and validation of mobile forensic software is simple: to overcome a challenge before being challenged. Software is created by a software engineer who is a human being, and human beings can make mistakes. Software code is developed to do one thing, but it sometimes does another. Relying on what a software company has stated regarding how a tool should perform is not good practice. It's the responsibility of the examiner to maintain and test any software to be used in the forensic lab or in the field to perform digital data collections and analysis from mobile devices.

When it comes to explaining the various reasons behind the verification and validation of tools in the lab, an examiner must be prepared to support each challenge when presented. If an examiner cannot support the reasoning behind tool verification and validation with credible data, the collection and analysis of a mobile device should not be completed or even initiated.

There are various challenges to performing a proper baseline that mobile forensic tools can stifle. The greatest challenge is typically regarding how the integrity of the device was compromised due to the intrusive methods in the collection of the digital data from the device.

Some claim that mobile forensic software can change, alter, and modify data on the device. Mobile devices typically have their data collected while they are powered on, so data can change even without software interaction. When a device is powered on, its operating system is constantly changing its system files, just as a computer does. The device will update directories and index files and will allocate memory and monitor the system clock. This even occurs when the device is isolated 100 percent from all network sources.

When testing a forensic application by creating a baseline, you can show that no user data was altered in the collection and subsequent analysis of the data. In baseline verification and validation testing of the software used in the analysis, you can show that no user data was altered, changed, or modified. Based upon the testing of the software and passing results, you can conclude that during the digital forensic investigation, the software performed as described and did not alter, modify, or change the user data as represented in the case.

Typical mobile device collections are conducted via a USB cable connected to a computer and to the mobile device. Using device and manufacturer protocols that are unlike any other digital device collection methods, communication occurs between the software and the device. Unorthodox methods such as using protocols to initiate a data backup of the device, installing a small program and initiating the program to query the device's internal user data repositories, and placing a custom partition on a device to allow for the bypass of security are a few methods used in mobile forensic software. These methods do not alter, modify, or change the user data as determined by the baseline testing conducted using the mobile forensic software.

Overcoming Challenges for Single- and Multiple-Tool Examinations

Another, more personal, challenge can come in the form of an attack of the credibility of the examiner who conducted the collection and examination. Any change from the expected results is said to be caused by a lack of training of the examiner or the improper type of training received, along with the forensic tool used.

Training is necessary to mitigate challenges of the improper use of the forensic software or any device used in the forensic examination of the mobile device. Being certified in the forensic software is not essential to conduct an examination using it, but a certificate can show the examiner's competence to perform collection and analysis of the mobile device. Becoming well acquainted with all of the software that is currently in the mobile forensic toolbox can occur only with proper research, usage, and ultimately training.

Single-Tool Challenges

In many instances, even if the examiner has more than one tool available, he or she will use a single tool from the start to the finish of the examination. This tool will be used to collect the data from the mobile device, create an image, analyze the data, and create a report. This scenario, more times than not, is the norm. Sometimes an examiner has only one mobile forensic tool in his or her arsenal because of budget issues or for other reasons. In any case, the challenge often cited is that data output cannot be verified using the same tool that created the first forensic collection. This is primarily because an anomaly in the software's collection of data from a mobile device would, in theory, exist each time the collection was performed on an exemplar device, showing no difference.

If the lab has multiple tools, then using a single solution from start to finish can be justified so long as the verification and validation have occurred as outlined previously. What becomes critical when only a single solution is available will be the validation, testing, and identification that no user data was altered over subsequent extractions of a testing device. You must conduct the baseline testing as documented earlier and cover the steps on baselining devices when device integrity is required.

If only a single tool is available for the collection and analysis of data, the examiner should be diligent in every aspect of the examination. Verification of the data must typically be managed by visual verification on the actual exemplar device during verification testing. Challenges to "missing" data that is recoverable with another tool can also occur when using a single solution. When data is missed and consequently critical to the opposing counsel, the examiner who was not able to recover this critical piece is often questioned as to why this data was not disclosed, which questions the credibility of the examiner.

Dealing with this type of challenge involves simply explaining that all software is not created equally, with some having very robust analysis features while others simply report on limited data. In this instance, the software that was used has limited analysis features and does not support the recovery of the indicated information; ergo, the data was not located.

Multiple-Tool Challenges

When you use multiple mobile forensic products for a single device collection, various challenges can involve how the image was created, whether the integrity of the image was maintained, and whether changes were made to the actual data within the case. To overcome these challenges, you can follow several key investigative techniques.

To avoid problems with a challenge to the creation of the image and its integrity, use only one piece of software to create the data image of the mobile device. It is extremely important that you make every effort to extract the mobile device data only once from the evidentiary device. The more times that a device is hooked up to a computer, queried, extracted, and accessed, the more likely corruption will eventually occur. Sometimes collecting the digital data from a mobile device more than once is unavoidable—for example, when the device data is extracted at a scene and then again at the forensic lab. In these cases, to avoid a challenge, make sure you examine and compare the first extraction data to that of the second collection. If you note a difference in the data collected, clearly document all of the changes and the reason the data has changed. If proper procedures have been followed to isolate the device from network connections, the second collection should not contain additional user personal information, but should this occur, provide additional documentation stating how this information was written to the device. Such changes to data will undoubtedly be used later to discredit the data on the device and examination.

Also, challenges can be made against the integrity of the evidence by using a secondary tool when the secondary tool can import and analyze another tool's created mobile device image. It becomes very important that when you create an image with mobile device software, you also create an overall hash value of digital data collected. Most mobile forensic software applications create an overall hash of the evidence image, but for those that do not, the examiner should use a third-party tool that will create a hash value. Several free file hash software applications are available online and can be downloaded to perform this task.

If you're using an online software application, it is important that you create a text file that can be associated with the image file that contains the hash value, the date the hash was created, the person who hashed the file, and the software that was used to create the hash. With this information, if needed, a second person can duplicate the steps to validate the image as well. Once the hash value is obtained, it will be used at the conclusion of the examination to verify the integrity of the image after a secondary tool has been used to analyze the primary tool image. Some software titles will actually verify the image's hash upon import, and if it is not identical from the hash obtained at the creation of the image, the user will be immediately notified. Other applications, such as AccessData FTK Imager (Figure 7-6), can run a function to verify the integrity of images created by AccessData products. This hash, as shown in Figure 7-7, can then be used to compare against the image when requested at any time during the evidence life cycle.

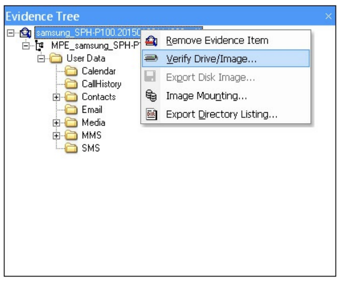
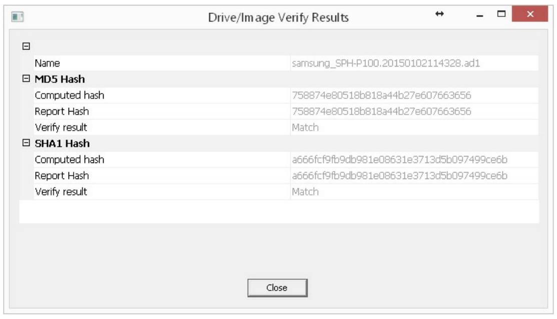


FIGURE 7-6 **Using AccessData FTK Imager, import the created image, right-click the device name, and select Verify Drive/Image.**



```
Drive/Image Verify Results

⊟
  Name             samsung_SPH-P100.20150102114328.ad1
⊟ MD5 Hash
  Computed hash    758874e80518b818a44b27e607663656
  Report Hash      758874e80518b818a44b27e607663656
  Verify result    Match
⊟ SHA1 Hash
  Computed hash    a666fcf9fb9db981e08631e3713d5b097499ce6b
  Report Hash      a666fcf9fb9db981e08631e3713d5b097499ce6b
  Verify result    Match
```

FIGURE 7-7 **The image will be tested and hashed and compared to the hash that was created when the image was made.**

Tip

When using a secondary tool to analyze a primary tool's image, make a copy of the actual evidence and analyze the copy of the image using the secondary tool.

Challenges when using multiple tools can also arise when a feature that is used to analyze data runs scripted code within the interface against the collected mobile device evidence. Scripted code could include code written in Python, C#, EnScript, or others. By using these features, the examiner can interact directly with the evidence; it is extremely important that you take steps to validate that changes are not being made to the actual evidence.

To overcome this challenge, test each feature that can act upon evidence with user-generated scripts. To test, create a simple rehash of the evidence containers after the use of the customized feature. If there is a difference in the hash values before and after running a script, you must document and research the issue to identify where the failure occurred and if the evidence was tainted. If the hashes match, as they should, after running the custom scripts, you should also document this so that if you're challenged, you can provide specific information regarding the testing completed to verify the function and image integrity testing.

Tip

When conducting any type of analysis, all work should be done on a copy of the actual evidence file. Evidence should not be used for testing or baselining purposes.

Chapter Summary

Data integrity is at the heart of a forensic examination. Maintaining confidence in the resulting output of a mobile forensic collection is critical. To do this, the examiner must be confident in the mobile forensic tools that are being used to perform the data collection and examination. By performing frequent verification and validation techniques, the examiner can be confident that his or her examinations will hold up when challenged. A great by-product of rigorous testing of the forensic tools in a lab will be tool knowledge. Expanding the examiner's knowledge of the tool from a traditional training curriculum or "I read the manual and web site" track with intimate hands-on work will not only benefit in court, but also in work production. Examinations of mobile data will be much more detailed and pointed, and will contain information that only an examiner very knowledgeable with a product could produce. The more that a tool can be dissected, tested, and used, the better the examiner will become at using the tool, and the actual investigation will reap the benefits.

Although there are many benefits to using multiple tools for mobile device examinations, not everyone has the luxury of using more than one tool. In these cases, you should ensure that the validation process is conducted vigorously. By using a single tool, not only will the examinations be limited to a single set of features and support, but the ability to verify and validate will be much more difficult. Using a single tool to conduct mobile device investigations can be accomplished, but there will always be a bigger risk of a challenge to the operation of the software and the examiner's methods. Also, the location of additional data could be jeopardized simply because the single tool does not support a feature, device, or data set. Selection of the mobile forensic tool, if allowed to have only one solution, is critical.

There are great benefits to using multiple tools to collect and examine mobile devices; nevertheless, with the use of multiple software applications comes risks. The many benefits range from verification and validation, to wider spectrum of device support, to the recovery of additional data by using features and controls of a secondary tool not supported by the primary tool. The risks come from the collection of the same device with different tools, ways in which tools validate the data with hashes, the inability to determine whether a tool altered the data, the integrity of the collected digital data, and the lack of correlation of the data between software applications.

Without the examiner having a firm understanding of the tools in his or her toolbox, there will always be risks. Overcoming any challenge to the collection and analysis of mobile device data comes in the form of understanding the operation of the mobile forensic software in the digital forensic toolkit.

8 Mobile Forensic Tool Overview

Selecting a mobile forensic solution can be an extremely difficult task. Trying to understand the capabilities of each tool can be confusing and at times cryptic. Although lots of mobile forensic software vendors list the vast number of devices that their products can support, you must remember that even if a product supports 18,000 devices, if it will not help with the single device that you need to collect, or the device is supported but does not extract the data that could be critical to the case, it's not the right tool for you. Vendors in the mobile forensic marketplace use the term "profiles" to describe their support and generally describe support for a "type" of device, loosely based on model. This enables the vendor to claim support for a subsection of the model or profile, even though not all of the device carrier's versions may be supported.

Quite honestly, the numbers game used by most mainstream vendors is just that—a game. The examiner should be aware of these types of nuances when selecting a solution that claims to support a specific model of device that will be examined. Most vendors post information online, where the examiner can go to determine whether a device is supported. But what does it really mean when a vendor says the software handles the "logical and physical collection" from an iPhone 7 running Apple iOS 11? Such information is often misleading, especially because logical and physical collections might mean one thing to one examiner or vendor and mean something entirely different to another.

In this chapter, several mobile device solutions are identified and the types of examinations are discussed—but in a more granular way that departs from the traditional nuances of logical and physical collections. Open source, freeware, and commercial tools all have benefits and limitations. Each piece of software covered here comes from my personal experiences as well as community experiences, but as with everything, one person's experience can differ from another's. Consideration of the level and type of examination required should be one of the deciding factors in the type of mobile device tools you choose to use for the collection and subsequent examination of recovered data. In some cases, the tool you use to collect a mobile device can be entirely different from the tool you use to complete the analysis of the data.

Just as confusing is the tool progression—the decision to move into another category of examination for a particular device or situation. Not all devices are supported with a simple connection via USB cable, and some are not supported by any traditional methods. Some devices must be examined after the removal of the memory chip, while others can be examined with a physical connection directly to the mobile device printed circuit board (PCB).

Mobile device forensics in today's world does not involve simply attaching a cable, clicking a Go button, and waiting for completion. To conduct a thorough mobile device examination when it counts is to comprehend the tool progression continuum, the risks involved, the training needed, and the tools available.

Collection Types

There is no more confusing discussion in mobile forensics than either listening to or speaking about a logical and a physical collection of a mobile device. This is one of the most danced around issues in mobile device forensics. What constitutes a logical collection and what constitutes a physical collection? If a file system was recovered and happens to contain files and folders, is that a logical collection? If the file system contains unallocated space, does it constitute a physical collection? To be honest, there is no clear definition for the difference between logical and physical with regard to mobile device collections, and there are too many variables involved. The following section discusses both terms, some historical data, and what today's working practitioner believes to be the definition.

Logical Collection

As defined in the 2007 publication, NIST SP 800-101, "Guidelines on Cell Phone Forensics," a *logical acquisition* implies a bit-by-bit copy of logical storage objects (such as directories and files) that reside on a logical store (such as a file system partition). In 2013, the Scientific Working Group on Digital Evidence (SWGDE) published the document "SWGDE Best Practices for Mobile Phone Forensics," which removed the bit-by-bit classification proposed by NIST and classified a logical collection in a different way. SWGDE stated that a logical acquisition implies a copy of logical storage of objects (such as directories and files) that reside on a logical store (such as a file system partition). In the 2013 publication "SWGDE Core Competencies for Mobile Phone Forensics," a logical examination is described as a process that provides access to the user-accessible files. Moreover, SWGDE indicated that the logical analysis process will "not generally provide access to deleted data." In a 2014 revision (currently being updated), NIST SP 800-101R, "Guidelines on Mobile Device Forensics," described a logical acquisition as capturing a copy of logical storage objects (such as directories and files) that reside on a logical store (such as a file system partition).

Note

The definition is getting closer between both groups, most likely because of the personnel who are now contributing both to the NIST and the SWGDE documentation.

Confusing, isn't it? There appears to be a common theme of files and directories from a mobile device, and honestly, the definitions have become better in describing the data available, but to a layperson reading this information and ultimately applying it to his or her job, it is cloudy at best, and almost mystical. The mysticism comes in the form of one of the terms used in SWEGDE's logical definition: "generally." It is because of this "generality" that the word "logical" has lost some credibility. The problem arises because, within a subset of devices such as iOS and Android devices, a logical file extraction can often contain deleted data. Simply stating that a logical examination is capturing the logical storage objects (such as directories and files) from the file system of a mobile device is *too general* and, quite honestly, in some situations inaccurate. The terms "logical" and "physical" acquisition are throwbacks to computer forensics parlance, and they are more often than not misinterpreted when it comes to mobile device forensics. Data that is not accessible to the user, such as system files, application stores, and deleted data, can often be collected from a device in a currently defined logical extraction. This would contradict some of the definitions mentioned.

A logical collection on its face should be interpreted as the extraction of user data from a mobile device without the collection of a device's file system. This data is extracted from the mobile device using proprietary protocols and queries and displayed in the software user interface. An example of a logical collection as just defined would be using a software tool on an Android device with an Android application package (APK) file.

Most forensic software applications use this method when conducting their version of a logical extraction. The APK queries the Android device's internal databases and returns the data to the software interface. The data is then displayed in the software's user interface. This method does not return a file system, but the data that is represented by the contents of the files on the device. This distinction is important, because the currently accepted definition makes an assumption that all logical collections and recovered data from a mobile device by software are similar, so long as the device is supported logically. Because of this often-used generality in describing a logical collection of a mobile device with a particular solution, a subcategory of a logical extraction should be discussed—a file system collection. A *file system collection* bridges the gap between what the mobile forensic community believes to be a logical extraction and a physical collection.

File System Collection

A file system collection contains much more information than the defined logical collection and should be considered a step up from a logical collection. A file system contains the files and folders that the device uses to populate applications, system configurations, and user configurations along with user storage areas. The following sections define and explore different types of file system collections.

MSC, MTP, and PTP Mobile devices such as iOS, BlackBerry, Android, and Windows Phones can have "points of storage," which could mean that the mobile device file system collection must occur in multiple places. A user storage area can be the location where images, videos, and audio are stored and accessible by the user via a computer and a cable. Another user storage area can be an internal storage point that also stores application data, system log files, and documents.

Mobile devices have long enabled users to transfer data to a PC using "USB Mode," or the USB Mass Storage Class (MSC). Legacy feature phones and devices, when removable media was added as a supported feature, would allow the user to move media from the device to a PC much like transferring data from a flash drive to a computer. In 2008, the Media Transfer Protocol (MTP) mode, originally part of the Microsoft Framework, became a standard by the USB Implementers Forum (USB-IF) as a USB type. This mode is recognizable when a device is plugged into a PC and is automatically mounted as a device, rather than a drive. This access occurs via MTP, a subset of Picture Transfer Protocol (PTP), which adds some enhancements and enables communication between the mobile device and the PC to copy, move, replace, and delete files from and to the mobile device. The move away from MSC to MTP was made in most modern mobile devices because if the device was in MSC mode, it could not store or communicate with the default storage point, making the device useless during an MSC connection. The device could not access applications, take pictures, or otherwise operate, so MTP mode was implemented, which enables the device to function even when tethered to a PC. Furthermore, the use of MSC exposed an entire partition to the computer system that would have to be properly formatted as FAT to mount and retrieve files. With the use of MTP, files can be retrieved at the file level. In the instance of Android devices using MTP, the device is able to move, store, and add a file without mounting a partition. Essentially, the device just acts as a file server to the client, and when files are requested, they are available no matter what underlying OS formatting is in use.

The "media" in MTP should not be confused with traditional media such as images, videos, and audio. In the MTP specifications outlined in Media Transfer Protocol v.1.1, the term "media" in Media Transfer

Protocol is used to identify any binary data and is not restricted to the audio/video formats to which it is commonly applied. This clearly indicates that any file that is stored can be recovered using MTP. Mobile devices running the Android system, Windows, BlackBerry, iOS, and feature phones of today allow for access to a logical file system via MTP and PTP.

The difference between the operating system, at a basic level, regards the extent to which files can be accesses and what types of files can be accessed in this manner. Apple iOS devices will allow access to the media (pictures and videos only) area via a file explorer, such as Windows Explorer and the Finder, using PTP. With an Android device, first by default in Honeycomb 3.0 and a standard in Ice Cream Sandwich 4.0 and later, the device can provide access to the internal storage area of the device, an external media card, or both, using MTP. The accessible data includes application files, user documents, and media and system files. A Windows Phone, like the Android, allows for the transfer of files from both the internal and external card (if available) using MTP. Windows Phone data from both locations can contain documents, images, audio, and video. BlackBerry devices starting at version 5.0 allow for MTP as well as MSC. If the BlackBerry is set as a portable device, access to the internal and external card storage areas is available. These storage areas can support multiple file types such as documents, audio, video, images, and system files. Access and recovery of these files should not be overlooked in the examination of a mobile device.

Table 8-1 shows the types of data available on basic mobile devices using MTP and PTP.

TABLE 8-1 **Mobile Device OSs and Data Accessible Using PTP and MTP**

Operating System	Type	Type of Data
Apple iOS	PTP	Images and videos
Android, BlackBerry 10, Windows Phone	MTP	Images, videos, and media (documents, files, other)

Figures 8-1, 8-2, and 8-3 show the various device types and how they are depicted within the desktop operating system.

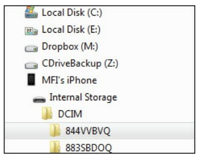


FIGURE 8-1 **File system view of an iOS device connected as a PTP device—DCIM (Digital Camera Image) folder is available**

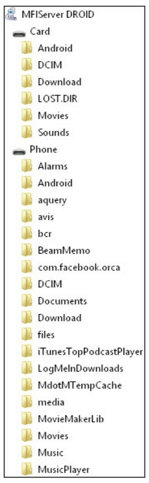


FIGURE 8-2 **File system view of an Android device connected as a MTP device—notice the internal and external file systems containing various files and folders**

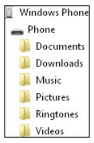


FIGURE 8-3 **File system view of a Windows Phone connected as an MTP device**

The files can be copied manually to an evidence location when the device is connected to the examination computer. Although this is not generally recommended, it is advisable if the forensic software does not support the collection of these files. To make these accessible to the computer, the device must be unlocked.

Internal System Collection and Display Some mobile device systems can also be accessed from a protocol level to display a represented file system. Feature phones using propriety file systems can have their file systems collected and displayed to show system files, user databases, media, user files, logs, user settings, and more, as shown in Figure 8-4. These files are not directly accessible to the user via the device interface, and inside of these file systems are artifacts that would not have been available without the use of specialized tools. These files are the actual containers queried and parsed by the logical software and displayed in the software interface. By having the actual file, you can conduct a more detailed analysis, which should be considered much more valuable than what "logical" defines. Similarly, mobile devices, such as Apple iOS, Android, and BlackBerry devices, also can have their underlying file systems collected to reveal application data, media, user files, system files, logs, user settings, and device settings.

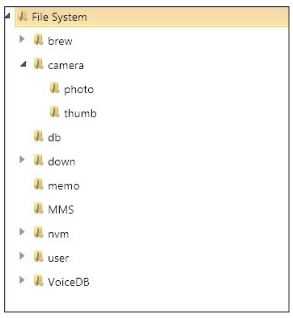


FIGURE 8-4 **In a CDMA file system, many files and folders can contain additional evidence.**

All three of the file system examples involve collections that can be lumped into the "logical collection" bucket definition. Because this level of collection is not available to all software solutions that identify logical support for the device, it makes sense to identify a secondary category or level for the tool used. For example, a vendor may describe providing "logical support" for an iOS device when the support is actually a query and return of the database content, which is then displayed via the user interface. In all actuality, an iOS device contains numerous files, as depicted in Figure 8-5, such as property lists (plists) and SQLite databases that contain user, system, and deleted data at the logical file system level, which is something that the vendor does not support but claims to. This type of recoverable data at the logical level clearly extends the definition of what is often described as a "logical extraction." Chapters 13 and 14 are dedicated to the analysis of these types of files, so you can clearly see that the term "logical" is often misrepresented by software vendors.

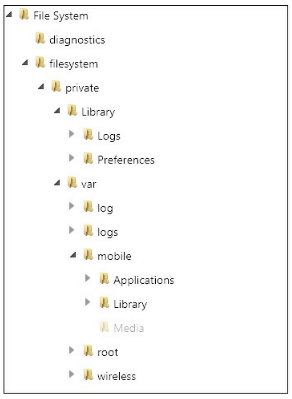


FIGURE 8-5 **In an iOS device file system, many files not represented by a superficial logical collection will be missed.**

When possible, the examiner should make a collection of a device's file system. The information contained far exceeds any data that is collected on the surface. Collecting the "surface" logical data along with file system recovery is what every examination should strive to accomplish. This type of collection should be referred to as a *file system collection*, not simply a logical extraction.

Physical Collection

To some, a physical collection includes application data, system files, and other information that is not available to the user via the handset. NIST SP 800-101 states that a "physical acquisition implies a bit-by-bit copy of an entire physical store (e.g., a memory chip)." In 2013, "SWGDE Best Practices for Mobile Phone Forensics 2.0" used a definition of a physical acquisition that was identical to that of NIST. In 2014, "Cell Phone Forensics in a Correctional Setting: Guidebook" was released and published by the Department of Justice, which described a physical analysis of a mobile device as a "digital forensic examination process that involves reviewing the data on a digital device by pushing a boot loader into the digital device and dumping the memory from it." Then, in 2014, NIST SP 800-101R described a physical acquisition as "extracting and recording a copy or image of a physical store (e.g., a memory chip)." Clearly, as with the definitions for logical collection, practitioners and researches interpret the definition of physical collection in various ways. In some instances, the definition is so vague that some software vendors are selling tools along with their own idea of supporting mobile devices “physically.”

If the NIST and SWGDE literal definition implies that a bit-by-bit copy of a memory chip is used, then software vendors providing “physical” support for legacy devices, or smart devices using a USB cable, are not compliant. Providing a bit-by-bit copy of mobile device memory would entail an interaction with the actual memory store using a tool that can read from block 1 to block *n*, where *n* represents the ending block of the flash memory chip. Inherently, a mobile device, when interacting using a USB cable, will be able to scan and collect data only in a way that is allowed at the operational level of the mobile device. In simple terms, a mobile device will enable the examiner to extract what he or she wants and only if the software can communicate in the way the device can understand. So a change is necessary in how an examiner should approach a physical examination of a mobile device and then define what is taking place.

The physical collection of a mobile device’s data should imply that direct communication with a device’s internal data storage is made to collect a representation of the data as it is stored on the actual device flash memory. This data is a snapshot of the area of the flash memory store that is accessible using specialized tools and methods. That being said, there are different levels of a physical collection that depend upon the type of specialized tools that are used. In “Best Practices for Mobile Phone Forensics 2.0,” SWGDE defines these as “non-invasive” and “invasive” physical collections, which are further explained in the following sections.

Non-Invasive Physical Collections

A non-invasive physical collection, according to SWGDE, “involves a process that provides physical acquisition of a phone’s data without requiring opening the case of the phone.” With this technique, the software must be able to communicate with the device to allow for a binary data “dump” of the device. In most cases, a collection using non-invasive techniques and software will not yield a physical image as defined by NIST (a bit-by-bit copy of the physical store). However, this method should yield a representation of the data area targeted by the software’s communication in the format in which it is stored on the device. The data can then be interpreted and displayed in the software as it was on the device, allowing for advanced analysis techniques.

An example of a non-invasive method is attaching a flasher box to a device’s USB port or FBUS connection and dumping the memory from predefined offsets known to contain user information. Another non-invasive example would be collecting an Android device using tools such as Oxygen Forensic Analyst or Detective, UFED, and XRY and selecting a physical option for a particular device that is not currently locked, with Android Debugging enabled. All mentioned tools communicate with the device to obtain partition information using the Android Debug Bridge (ADB), and they subsequently extract the returned partition table and partitions without altering the device partitions or operating system structure, which should be considered non-invasive.

Note

Communication does not always necessitate the loading of a boot loader, as some literature indicates, but elevated privileges to the device do need to be obtained by the software conducting the physical collection (such as HEX dump or data dump).

These methods, however, target only what is visible by the communication method. Various partitions are not enumerated by the Android device's operating systems by design and are not accessed or extracted by the forensic software. Again, the partitions that are available are extracted and in the format (file system) found within the actual device, but it is not a bit-by-bit representation of the entire device memory store.

Invasive Physical Collections

An invasive process, according to SWGDE, provides physical acquisition of a phone's data and requires disassembly of the phone for access to the circuit board. For example, JTAG (Joint Test Action Group) or ISP (In-System Programming) allows for communication with a mobile device using the device test access points (TAPs). JTAG is not a direct read of the actual memory module (flash), but a method to communicate with the device processor to access the NAND area of the device and obtain a binary file containing a representation of the partitions on the device. ISP differs from JTAG in that the processor is bypassed and the connection is made directly to the embedded Multi-Media Controller (eMMC) or embedded multi-chip package (eMCP) flash memory. Again, if the software is interacting with the device microprocessor, the microprocessor will dictate what memory stores are available and where to read from. The use of JTAG and ISP is classified as invasive primarily because the direct interaction with the mobile device circuit board is necessary when soldering to the TAPs or using specialized connections directly to the circuit board.

Note

Usually, a device is still functional after a JTAG or ISP procedure.

Another example of an invasive process is the removal of the memory chip from the mobile device, typically referred to as a "chip-off." Chip-offs are destructive methods, however, and generally the mobile device will not be functional after using this technique. A chip-off, however, will enable a direct read of the memory chip using specialized hardware and software. The examiner can create a full binary file of the device memory flash without limitations typically imposed by a device microprocessor. This physical collection method would conform to a bit-by-bit representation of the entire device physical store and equates to a traditional hard drive collection. The resultant data must also be interpreted by software and a represented file system compiled from the binary file in order to conduct further analysis. As devices progress along with more security-focused operating systems, the encryption of the device at the file system and file levels will hamper JTAG, ISP, and chip-off examinations as well.

Collection Pyramid

A "collection pyramid" was developed by Sam Brothers of the U.S. Customs and Border Protection and described in the SWGDE document "Best Practices for Mobile Phone Forensics 2.0" and also later published in "Guidelines on Mobile Device Forensics." This pyramid outlines a tool classification that can be used as a practitioner's approach when conducting mobile device examination; it is a departure from merely classifying a tool as logical and/or physical. Using this approach, the examiner can outline the analysis methods without identifying the differences between a logical and physical acquisition relative to the device or the data, but relative to the actual type of analysis and extraction methods used.

This ideology traditionally caused confusion in the mobile forensic community. Because of the various opinions as to what constitutes a logical examination versus a physical examination and the comparison to the definition imposed by computer forensic examiners and governing bodies of forensic practitioners, the use of multiple analysis methods is recommended. By describing methods of collection of a mobile device, an examiner can predict the outcome of using the available forensic tools and also the training and expertise required for each discipline.

The collection pyramid is visually represented as an analysis ranging from the most invasive and specialized tool to the least invasive and least specialized tool (see Figure 8-6). The tip of the pyramid, or smallest represented part, is the most specialized tool, and the largest and base of the pyramid is the least specialized. These methods are numbered from 1 to 5, with 1 being least specialized and 5 the most specialized. The following sections present the different levels listed by name, as provided by Brothers, with additional information to explain better what each will offer to the examiner when conducting each level of analysis.

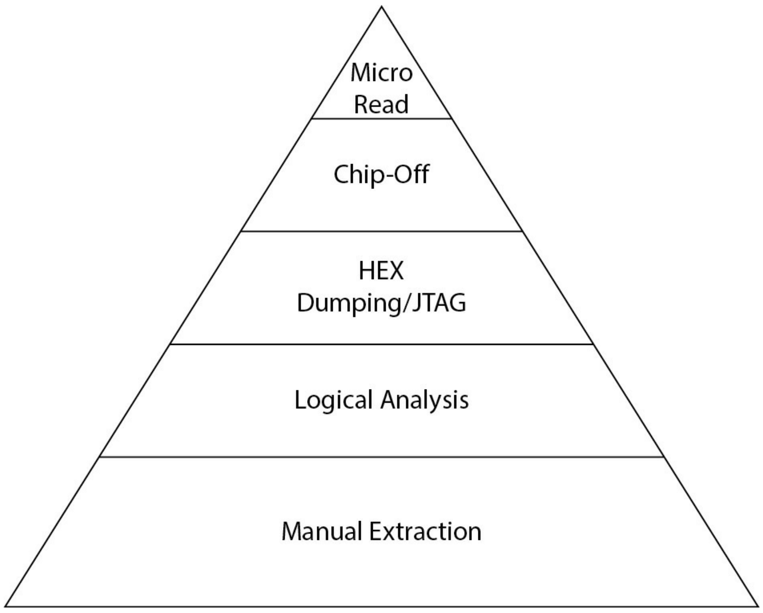


FIGURE 8-6 **Tool classification pyramid (copyright Sam Brothers, 2015)**

Level 1: Manual Extraction Manual extraction, or examination, involves capturing stored device information either by photography or written documentation. Photographing the information would be much more reliable in legal proceedings and therefore is the preferred method. Conducting a manual examination via the "commando method" or by "thumb jockeying" the device involves manual manipulation of the device to locate evidence stored inside. This method involves navigating the mobile device to the user stored areas and photographing or writing down the content observed in the device's viewing area. The first mobile examinations were conducted using the manual examination method, but mobile devices at that time were not capable of storing the amount of data that can be stored in today's devices. Conducting a manual examination of a smart device today is a considerable feat when prompted to document thousands of mobile device text messages, videos, or images!

Manual extraction may be the most cost-effective, and sometimes the only available, option, but it can be difficult at times and dangerous in others. If a mobile device is damaged or locked or data is stored in a foreign language, the documentation can be difficult, if not impossible. Manual examinations of a mobile device can be conducted by examiners without training and with only a digital camera, which involves more people in the examination process. Manipulating the keys and navigating a foreign device could expose the possibilities of unintentional data corruption at a later date, and there may be a complete disregard for the preservation of the data. Tools for conducting manual examinations are further discussed in the "Tools Available" section later in the chapter.

Level 2: Logical Analysis A logical extraction occurs using a built-in device transfer method (such as USB, Wi-Fi, IrDA, or Bluetooth) used by the mobile device. A connection is made with the device using a data transfer method, whereby the software can communicate using device protocols to extract data using commands comprehended by the mobile device. The data is returned to the software, which can be further analyzed and reported. This is the type of collection currently offered by most examiners as well as forensic software vendors.

Level 3: HEX Dumping/JTAG/ISP This level of collection uploads specialized software into the volatile memory of a mobile device. In doing so, it bypasses built-in security that would typically inhibit access to the device's internal memory store. However, devices that have chip-level encryption enabled will still pose problems for examiners.

Tip

This method could be used when conducting an analysis of an iPhone 4. These devices do not allow access to the internal protected stores that contain protected data (for example, Apple's built-in e-mail client) using any of the connection methods and communication described in a logical extraction.

A custom application or package is installed onto the device in an effort to act as the original application, package, or ROM on the device that contained the security measure. Once this vulnerability has been patched with the vendor's own application or package, allowing access to the device that was inhibited by the device file, the examiner can then access the files using commands and procedures used by the mobile device. Typically, a raw file system, represented in the format used on the mobile device, is extracted.

A subsection of this portion of the pyramid also belongs to flasher boxes and JTAG methods. Both will be described in the "Tools Available" section, but to help you understand their placement in this pyramid, they are briefly described here.

By using the JTAG TAPs in a mobile device, the examiner has access to the flash memory. Using specialized tools comprising hardware and software, the examiner uses software to communicate via hardware to the microprocessor of the device that interfaces with the flash storage medium. The examiner accesses the flash area, circumventing password security, to obtain partition information and user storage areas. Using JTAG or ISP can be invasive because the device is disassembled and in some instances leads are directly soldered to the TAPs on the circuit board, but in some instances preconfigured jigs can be used. The output when using JTAG or ISP is a binary file of the selected partition or memory area.

The output produced by most flasher boxes is represented by what the hardware flashing device has been configured to output. The output can be in an encrypted format, segmented, or altered (boot loader added to start of image), or it can be a flat binary file. This is truly a hexadecimal representation of the data living on a mobile device. Limitations to flasher boxes are numerous, ranging from proprietary output to flash area memory constraints. Data output produced with the use of JTAG or ISP methods offers a better representation of the data, with little interference with the digital data output—it is the preferred method, but it can be more physically destructive to the mobile device. The decision to use these devices should be left to examiners with appropriate training and an understanding of the various formats and implications of using the tool and its invasive nature. This type of collection not only involves specialized hardware, but also specialized training and skills.

Level 4: Chip-Off The chip-off examination involves the physical removal of the mobile device flash memory. Using specialized tools and techniques, the examiner disassembles the device and removes the flash memory from the circuit board. Once the flash module is removed intact, it is placed into a specialized component to read memory modules. These memory module adapters are specific to the type of flash memory and its configuration. A binary file is produced upon reading that must be interpreted by software that specializes in the decoding and interpretation of this type of file. An examiner who should be conducting these examinations should be well trained; otherwise, the evidence could be easily compromised. Needless to say, a chip-off examination is invasive, and once the chip is removed from the device, the chip would need to be *reballed* and then reinstalled into the device so it could operate as it had previously. This process is extremely labor intensive and equally expensive. Generally, once a device has been disassembled at the chip level, the device is inoperable.

Note

During the removal process of the BGA chip, you must remove the tiny pads that allow for the flow of input and output communication to the PCB; these must be reattached, which is called *reballing*. Reballing involves affixing tiny solder beads to each pad on the memory chip. This is no easy feat, as some flash memory chips have more than 200 pads! These beads, once affixed, are then reheated (reworked) to allow them to become permanently attached to the flash memory, which will allow communication with a chip programmer.

Level 5: Micro Read With this level of examination, the flash memory medium is read by an electron microscope. This type of examination is not only considered theoretical, but it has never been conducted publicly on mobile device evidence. The theory involves using an electron microscope to read and count electrons that occupy a cell on a flash memory chip. If electrons are present, a 1 is represented; if no electrons are present in the cell, a 0 is represented. This is often referred to as “gating.” After combining the binary data manually, it can be translated into raw data and interpreted. An example of the work involved to translate the information after reading from a flash chip is shown in this example:

```
01001101 01101111 01100010 01101001 01101100 01100101 00100000 01000100 01100101
01110110 01101001 01100011 01100101 00100000 01000110 01101111 01110010 01100101
01101110 01110011 01101001 01100011 01110011 00100000 01100110 01101111 01110010
00100000 01110100 01101111 01100100 01100001 01111001 01110011 00100000 01100101
```

01111000 01100001 01101101 01101001 01101110 01100101 01110010

What is incomprehensible would be the interpretation and decoding of more than 32GB of binary data from a standard Android device using this technique. Most examiners will never experience this form of examination and collection, but those who do this will probably not boast of their exploits, since this work would likely involve matters of national security.

Collection Additions

Although the collection pyramid does capture a vision of classification based upon collection of digital data from a mobile device with a tool, some additional considerations must be addressed and added to the continuum. If classifying a tool and a collection, you must consider a few other methods and subsections involved.

Descriptions of manual examinations using a specific tool or method should include a subset that describes using imaging (such as a photograph or video) or not using imaging. Suggested additions include manual extractions, manual with documentation, and manual with photography and documentation. There are obvious differences between simply pushing buttons and writing down the contents versus pushing buttons and then photographing the screen.

Pushing buttons on the device and writing down the contents displayed takes no tool training and is very labor intensive; it should be classified separately from more thorough methods. Methods progressing up the pyramid would involve manually pressing a device's buttons and then photographing or creating a video of the various screens of interest. By using video, the examiner is not only documenting the data on the screen but also the procedure he or she used. This can help to dispel any allegations of improprieties that may have occurred during the manual examination.

In the logical classification of the pyramid, a file system extraction should be included for defining a tool and the mobile device analysis. Some tools do, in fact, perform a logical extraction, but they fail to support the device file system and should be classified separately. A file system collection will contain data not accessible by the user as well as data that has been deleted by the user, unlike a traditional logical query. Analyzing a mobile device file system can be challenging and time consuming, and extensive examiner training is necessary, which also should be considered in the tool and analysis continuum. Because of the enormous amount of additional data that can be collected in a file system extraction, it is always the preferred method. However, most collections by mobile forensic tools and examiners are logical without an attempt to collect the file system because of a lack of understanding and expertise in the analysis of the resultant collected data.

Note

SWGDE identifies a file system level of collection in "Best Practices for Mobile Phone Forensics 2.0."

At the pyramid's HEX Dumping and JTAG collections level, the term "boot loader" is identified as a method that satisfies a higher level of tool collection and analysis. Because a boot loader is not always needed to obtain a raw binary dump of a mobile device, a better classification should be used. A new tool and collection pyramid can be created to indicate not only the types of tools that can offer the type of collection required but also the level of complexity of each tool, as shown in Figure 8-7.

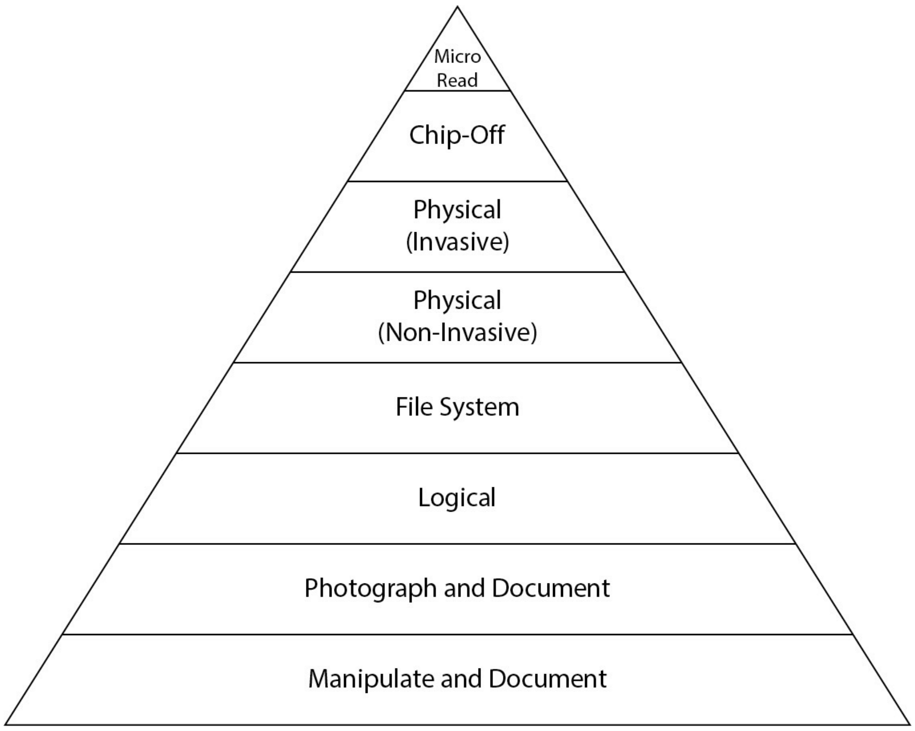


FIGURE 8-7 **A clearer representation of both tools and analysis, including portions of SWGDE and the tool classification pyramid**

To understand these new categories, you need to understand boot loaders.

Boot Loaders

A boot loader is code that loads in a runtime environment or operating system. Boot loaders can be used in nearly all digital devices that have an underlying operating system. With mobile devices, a boot loader can change depending upon the hardware as well as the device's service carrier. If the boot loader code becomes corrupted, the device either cannot be started or will continue to restart over and over—this is referred to as a "boot loop."

The use of a boot loader to obtain a physical collection of a mobile device is classified in the "SWGDE Best Practices for Mobile Phone Forensics 2.0" as a non-invasive physical classification. Considerable operational differences are involved between using or not using a boot loader. Tool classification should be further expanded to describe that not all collections of binary data from mobile devices will involve using a boot loader, and in some instances, altering a device's boot loader can have dire consequences.

To use a custom boot loader in mobile forensic software, the examiner places the device into a particular mode—for iOS devices, this is called Device Firmware Update (DFU) mode; most Android devices use a recovery, download, EDL (Emergency Download), or fast boot mode. This can occur operationally by the software or hardware intervention, but the examiner typically places the device into this state with a combination of key presses. Once the device is in the correct mode, the examiner selects the device make and model, and the software then replaces the device boot loader, unlocks the boot loader, and in some instances of Android replaces the ROM with a custom recovery ROM. The software then begins the process of calling instruction code to complete the collection of the mobile device's data store. The customized version of the boot loader and/or recovery ROM loaded onto the device has been designed to allow full access to the device's memory store and additional settings relating to data transfer. With a customized version in place, communication can occur with the software to obtain unadulterated access to the device.

Typically, on completion of the data collection, if the boot loader or ROM was exchanged, the original boot loader or ROM is returned to the device. In many cases, the customized boot loader is installed in the volatile area of the device's memory and will be released upon restart of the device to remove it. The use of a custom boot loader or ROM is specific to each device, and if the custom boot loader is used on another device, corruption of the device boot process can occur and render the device useless. For this reason, a classification for "physical non-invasive" should include a subset that describes tools and analysis if you are using a custom boot loader; otherwise, these techniques should be classified as "physical invasive."

Note

Tools such as Cellebrite UFED Touch 2 use a modified USB cable to place a device into device mode without requiring a keypress pattern.

Not all binary collections described as "physical non-invasive" have boot loaders or ROM altered to obtain a hexadecimal dump of the memory store. Some techniques used in mobile forensic tools also allow physical access to a device memory store by the addition of a code set not available in the device's original codebase. The action places a file or files into the temporary space of the device's logical file system. This code then adds commands that are not installed by default to be executed to elevate privileges on a mobile device. With elevated privileges, the software can communicate with the mobile device and obtain access to areas of the memory store that are typically inaccessible. Once these areas are accessed, the memory storage can be recovered from the mobile device.

The classification "HEX/JTAG" and even "physical non-invasive" should include these possibilities and not assume that this classification applies only to tools and collections that use a boot loader to access a device memory store. This classification should indicate that the collection involves using customized techniques to enable access to the mobile device memory store that are not accessible ordinarily, because some methods might be non-invasive to the physical device but invasive to the internal operation. This classification would apply to boot loaders, rooting software, customized files, and customized ROMs.

Nontraditional Tools

Other tools can be used in many investigations, but because of their limitations or other concerns, they are considered "nontraditional" tools. Some tools are considered non-forensic, and others are extremely expensive. Having a thorough understanding of how these tools can be used is important because these tools are often discussed throughout the digital forensic community. Likewise, in some situations, nontraditional tools may be the only means of recovering the data.

Manual Examination Tools

A manual examination of a mobile device can include photographing a mobile device's onscreen digital content using a tripod and a digital camera. The examiner will move from screen to screen, photographing the various pieces of evidence. This technique can be accomplished without specialized training in digital data recovery.

Tip

For a manual examination, use a digital video camera at the beginning of the work and allow it to run continuously during the entire process. This entire work can be reviewed for court, final reporting, or both. Stills from the video can be captured when needed for inclusion in a written report.

Some commercial products enable you to capture images of mobile device evidence. Using these tools, you can immediately document images or video in associated software for later review and reporting. There are, of course, limitations to the types of information that can be obtained using these tools, but little to no training is needed to operate them.

The following solutions are similar and enable the examiner to obtain information from the mobile device that may not be possible using traditional mobile forensic tools. Also, by using a manual solution, the examiner can allow for the documentation of the manual manipulation of a mobile device prior to conducting a traditional examination to show the actual device during testimony or presentation.

- **Paraben Project-A-Phone ICD8000 and Paraben Project-A-Phone Flex** These camera setups allow for both HD video and 8-megapixel pictures. The ICD8000 uses a clamping mechanism that can inadvertently press buttons on the side of the mobile device, including the power button on the right side of most Android devices, and improper clamping could change settings or power off the device. The Project-A-Flex does not use a vise but instead uses a mat on which the device can be placed to photograph the evidence.
- **Fernico ZRT3** This tool combines a camera and HD camera along with materials to hold the device and cameras in place. The device connects directly with software installed onto a PC to capture the photographs, and it uses technology such as Object Character Recognition (OCR) to translate images containing text to searchable text within the report. The device includes a mat onto which evidence can be placed to conduct the manual interrogation of the device.
- **Teel Technologies Eclipse 3 Pro** This tool is similar to the ZRT3 and combines a camera, mount, and platform, with a software solution to capture and document the images collected during the manual examination.

Flasher Boxes

A flasher box is a service tool that is typically used by mobile device technicians to fix a nonresponsive device, add features, or unlock a device for unrestricted access with any carrier. The tool's name is derived from the action that the device is built to perform—it flashes a new version of the device firmware, ROM, OS, or settings. The flasher box could also be used to add language packs and even to change the serial number of the mobile device. The altering of a mobile device's serial number (IMEI [International Mobile Station

Equipment Identity] for GSM and ESN/MEID [Electronic Serial Number/Mobile Equipment Identifier] for CDMA) is illegal, however, and by changing the serial number, some devices can operate on a network blacklisted by their original serial number. Device manufacturers do not endorse the use of these types of hardware tools for these reasons.

These hardware devices were never designed for mobile forensics, but they are frequently used, and their technologies were used to formulate a lot of the current mobile forensic product capabilities. Each flasher box functions, programs, and repairs a specific device—sometimes a specific model of mobile device. Because of the various differences in device hardware and programming, many hardware flashing tools are available to fulfill this need. Not only are multiple boxes available, but these boxes come with multiple cables. Typically, each mobile device must have its own specific cable, and hundreds of cables are available for these boxes.

Because the devices are developed to repair and alter a mobile device's software, the software used to control the flasher box can read and write to the attached device. Because of this ability, most examiners will not use a flasher box to conduct a physical examination of a mobile device; in addition, there is little to no documentation regarding how to use the hardware and software. This type of analysis should be classified as invasive, and the examiner needs specialized training before using this technique.

A flasher box will communicate with the mobile device using serial and USB protocols to recover areas of mapped memory. Memory offsets must be known if not identified by the software to recover the area where user data is stored. The resultant output is a binary file that often must be manually parsed and interpreted to locate evidence artifacts. In some cases, the data that is output is encrypted in a format known only to the flasher software. In this way, the flasher box manufacturers hide methods and procedures from other flasher box manufacturers. This limitation makes it difficult to examine the produced file. Table 8-2 represents some of the common flasher boxes and a general idea of the device support.

TABLE 8-2 **Common Flasher Boxes Used in Mobile Forensic Device Collections**

Box	Support
Ns Pro	Samsung
Z3x	Various Samsung devices (Agere, Sysol, Swift, Infineon, OMAP, Qualcomm)
Octopus Box	LG and Samsung
SHU Box	Nokia and Sony Ericsson
Advance Turbo Flasher (ATF)	Nokia Legacy, Nokia Lumia Series (SL3)
Vygisoft Toolbox	LG
Infinity-Box	MTK, ZTE, Huawei
IP-BOX and IP-BOX2	iOS PIN unlock

JTAG

The Joint European Test Action Group (JETAG) was formed in 1985 in Europe as a standard for boundary-scan testing. Boundary scanning in the context of mobile devices efficiently tests connections on the printed circuit board in an effort to program or debug the device without needing physical access to the flash. In 1986, members from North America joined, and the group became the Joint Test Action Group (JTAG).

Between 1986 and 1988, the group proposed and published a series of proposals to the IEEE Testability Bus Standards Committee (P1149) with the final JTAG Version 2.0 being accepted. This standard (IEEE Std. 1149.1) was accepted and published in February 1990 and has been updated several times with the current specification identified as IEEE Std. 1149.1-2013. At its core, JTAG is the standardization of TAPs and boundary-scanning architecture.

Note

The word "JTAG" has a variety of meanings, from directly programming systems to debugging others, from Xbox hacking to forensics. In the context of this book, JTAG is described as the process of setting and reading values on the test pins accessible on the PCB of the mobile device. By using the TAPs, communication can occur via the boundary-scan path, interfacing with the Boundary Scan Registers (BSR) that interface with components on the PCB. These components can be programmed or read without removing, independently reading, or programming each separately.

In order for communication to occur with components, IEEE Std. 1149.1 indicates at a minimum that three input connection ports and one output connection port must be on a PCB. Furthermore, the TAP is a multipurpose port that enables access to test support functions built into a component, and the standard outlines that the TAP shall include TCK (Test Clock), TMS (Test Mode Select), TDI (Test Data In), and TDO (Test Data Out) as connections. An optional input port, TRST (Test Reset), can also be used and is covered in the IEEE document. A basic understanding of each TAP is outlined next in an effort to help you visualize what is taking place when conducting a collection using JTAG, which is also represented in Figure 8-8.

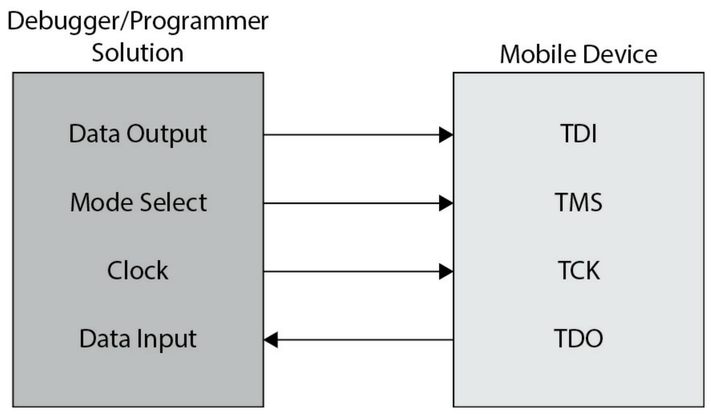


FIGURE 8-8 **A simplistic communication model from a software debugger/programmer to and from a mobile device TAP group**

- **TCK (Test Clock)** This port enables the synchronization of the internal state of the device between components. Devices are made of many components that could be using different forms of timing. The

TCK maintains a standard across the components during a test.

- **TMS (Test Mode Select)** This port controls the TAP controller and relies on the TCK to determine the state of the process.
- **TDI (Test Data In)** This port accepts the data from the software debugger/programmer and sends it to the target.
- **TDO (Test Data Out)** This port accepts the data from the target and sends it to the debugger/programmer software.
- **TRST (Test Reset)** This port is optional but can be used to reset the TAP.

The TAPs for mobile devices are not readily documented, and manufacturers are making it more difficult to locate them on the device PCB. Some manufacturers, such as BlackBerry, have gone to massive lengths to hide the TAPs or place them where any access destroys the device. Some software and hardware vendors do have the devices that they support listed within their software showing the TAP locations, and some software can determine the correct TAPs automatically.

JTAG hardware for mobile devices is just another type of flasher box, but the point of communication and interaction is different from that of the typical box. The difference is the fact that serial communication occurs to and from TAPs located on the mobile device PCB, unlike most flasher boxes that communicate using the traditional USB connector pin-outs on the device.

Note

Some flasher boxes used for Nokia models communicate using the Fast Bus (FBUS) or Meter-Bus (M-Bus) connectors on the PCB, which are early forms of TAPs. These connections were exposed and available under the battery, as shown in Figure 8-9.

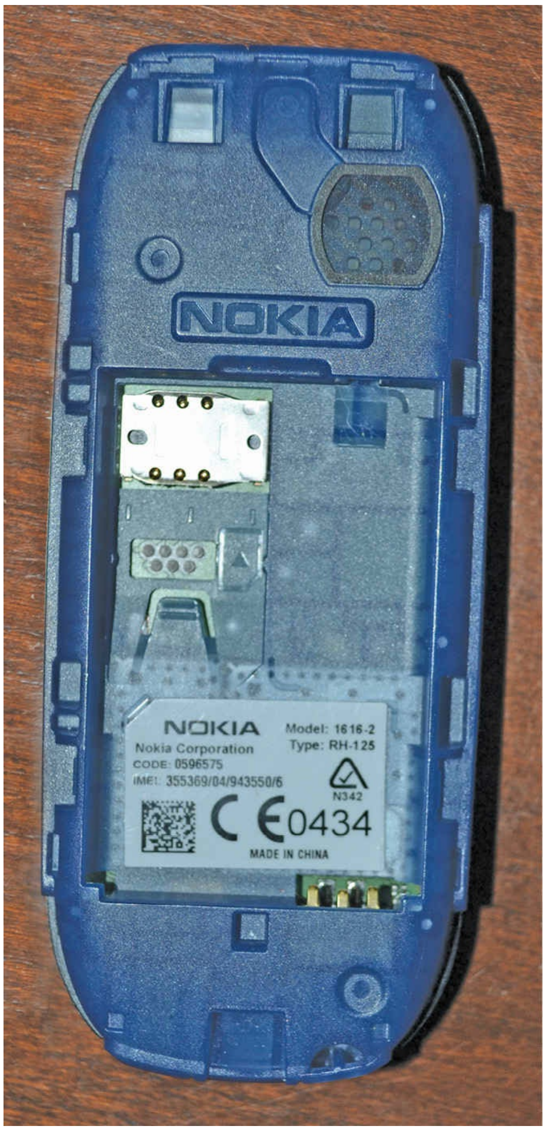


FIGURE 8-9 **FBUS connections on a Nokia 1280**

JTAG hardware communications occur from the software to the hardware box through a cable or wire attached to the TAPs on the mobile device PCB. The wires to the device TAPs must be in contact for communication to occur. You must locate the TAPs and determine the type of test port in use. Because some JTAG boxes do not auto-detect the port type, this process can be time consuming. To determine to correct test port, you can use a secondary tool to scan the TAPs and identify TCK, TMS, TDI, TDO, and TRST. One such tool, the JTAG Pin Finder from 100RandomTasks, enables the examiner to attach wiring from the JTAG Pin Finder to the TAPs on the PCB and use associated scanning software to determine the correct ports. Once the correct ports are identified, the wire can be attached to the JTAG hardware to conduct a collection. Both the scanning of the port and subsequent collection involves soldering the wire to the appropriate TAP, as shown in Figure 8-10. However, in some instances, special Molex connectors can be used that snap directly onto a female Molex connector located on the mobile device PCB (Figure 8-11). Using a Molex connection would mean there is no need to solder the wires directly to the TAPs on the device PCB, which can be advantageous when attempting to determine the level of invasiveness that will be acceptable.

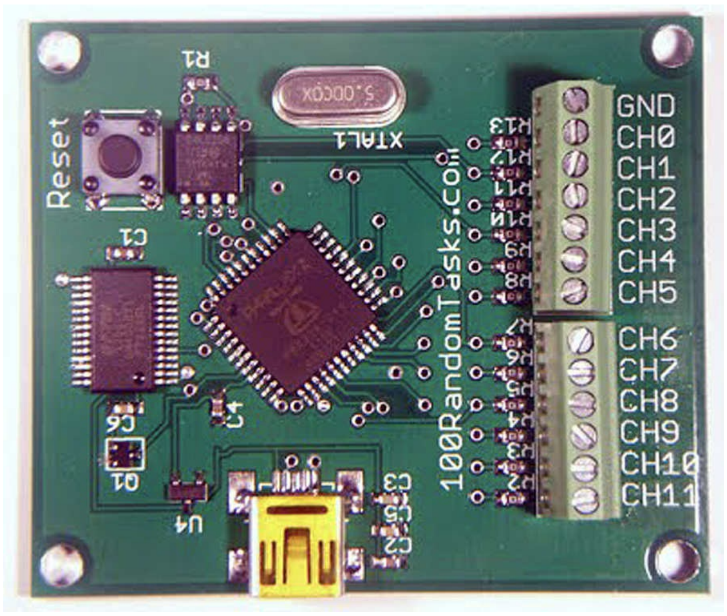


FIGURE 8-10 JTAG Pin Finder for scanning JTAG TAPs

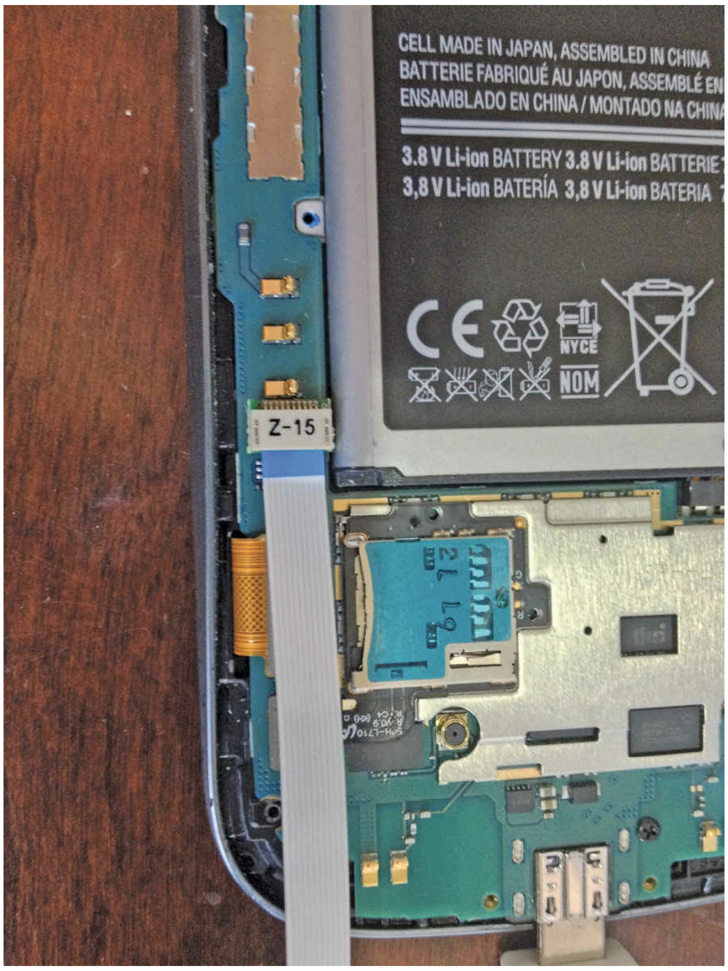


FIGURE 8-11 Molex connector attached to a Samsung S3 JTAG TAP

JTAG software undergoes communication with the mobile device, and the examiner identifies the type of data to collect. When used correctly, the software can bypass all security of a mobile device and obtain entire device images in the file system format that exists in the flash memory. The JTAG software does not interpret the data to allow for a forensic analysis and the rebuilding of the file system, but several tools such as Cellebrite Physical Analyzer, MSAB XRY, and Oxygen Forensic Detective can import and analyze such images obtained from a JTAG collection.

Again, like using flasher software, JTAG hardware and software was developed to flash, alter, and repair a mobile device. It was never the intention of the JTAG hardware and software designers to offer features that would be used by mobile forensics examiners. Because of this, examiners must be properly trained in using these tools. All associated JTAG software can destroy evidence if not used correctly in a controlled environment, but when used properly, it can yield magnificent results. Table 8-3 lists several JTAG hardware and software combinations with supported device information.

TABLE 8-3 **Commonly Used JTAG Boxes in Mobile Device Collections**

Box	Support
RIFF/RIFF 2	Samsung, HTC, Nokia, Huawei, LG, ZTE, and others
Medusa Box	HTC, Huawei, LG, Samsung, Sony Ericsson, ZTE, and others
ORT JTAG	Samsung, LG, HTC, Huawei, ZTE, SKY, SE, and others
GPG JTAG	Google, HTC, Dopod, and others
Z3x	Samsung, LG, Benq-Siemens, and others
ATF	Samsung, Huawei, Nokia/Microsoft Lumia

ISP

In-System Programing (ISP) is a method similar to JTAG, but as mentioned previously, using the ISP technique will communicate directly with the flash memory and not through the processor. As with JTAG, the TAPs will be utilized to connect to the PCB and then to a hardware device to conduct the communication directly to the eMMC/eMCP and transfer the data to the software interface. The benefit of using ISP rather than JTAG is there is no dependence on the processor of the device—only the flash memory. Acquisition is not limited to only mobile devices but anything that is utilizing eMMC or eMCP flash memory, if the TAPs can be located and accessed. The TAPs identified by flash manufacturer Kingston Solutions, Inc., in the document "Embedded MultiMedia Card EMMC04G-M627-A01," that can be utilized for ISP are the CMD, DATA0, CLK, VCC, VCCQ, and GRD. Listed in Table 8-4 are some hardware devices commonly used for ISP collection.

TABLE 8-4 **Commonly Used ISP Boxes in Mobile Device Collections**

Box	Support Needed
ATF	eMMC adapter
RIFF Box and RIFF Box 2	eMMC adapter
Z3x Easy	eMMC adapter

Chip-Off

The removal of a device's flash memory module and analyzing in a chip-off procedure is quite labor intensive in both the removal of the actual embedded flash memory chip and reading of the stored data. Because of the various differences between phone models and storage types, which could range from Thin Small Outline Package (TSOP) to Fine-pitch Ball Grid Array (FBGA), the purchase of the equipment to read the information from the embedded chips and the time investment required can be expensive. However, conducting the imaging of the memory chip using this method is the closest to the bit-by-bit collection a forensic examiner would expect and is similar to imaging a computer's hard drive.

To perform a collection using the chip-off method requires that the examiner have specialized skills in disassembly of mobile devices, repairing of mobile devices, desoldering and soldering of fine electrical components, and forensic procedures. This technique is commonly referred to in electronics as "rework." Depending on the type of chip, TSOP or FBGA, the removal and preparation of the memory chip will require different skills and time investment.

For TSOP chips, the pins that attach the chip to the PCB are exposed and can be easily removed by heating the solder joints. BGA chips must be heated to the correct temperature to remove the solder joints and adhesives and then be carefully removed from the PCB because their solder joints are not accessible along the exterior, as are TSOP chips. Both chips must be removed with extreme caution, because they cannot be reattached to the mobile device after removal. Upon successful removal, both TSOP and BGA chips must be thoroughly cleaned, examined, and inspected. For BGA chips, once they have been examined, they must also be reballed.

Once properly prepared, both TSOP and BGA chips can then be attached to the appropriate adapter and read in the designated chip programmer.

Tip

TSOP chips are not as common as BGA chips in today's devices and are used in mobile devices on limited runs, prototypes, and smaller manufacturers. BGA chips allow for larger data capacities and faster input and output (I/O).

A *chip programmer* tool will allow for the collection of the raw data from the memory chip. The programmer itself can also be a detriment, much like a flasher box or JTAG box, because it can write to and even erase a memory chip if it's used incorrectly. Examiners must follow proper procedures when using such equipment. Also, typically, more than one programmer should be available to an examiner because there are many different types of embedded flash memory types with the many mobile devices, adding even more complexity. In addition, prior to placement in the chip programmer, the memory chips are affixed to unique adapters, and each of these can cost upwards of $1800! These adapters must be compatible with the memory chip type—with the many differences in BGA-style chips, this can become extremely expensive.

Once the chip has been placed into the correct adapter and into the chip programmer, the examiner must select the identical chip and model as printed on the memory chip to begin the reading process.

Tip

The exact chip manufacturer and chip module must be selected and supported by the chip programmer. This step helps to identify unique preferences, algorithms, and settings that might be needed to perform a collection.

The analysis of the file produced by reading the memory chip is often the most tedious part of the process. Because a mobile device memory chip is based on flash memory (such as NOR or NAND) chips, the inherent advantage of I/O when compared to a traditional spinning hard drive might be the biggest detriment to forensic examination at the chip level. NOR flash memory is older technology that allows for high read performance, but it does not allow for high capacities. NAND flash memory allows for both faster programming and erases, but it can consume more power because of its higher functioning and complicated I/O interface. To understand some of the problems that could be encountered, you need a better understanding of flash storage.

How Flash Memory Works

Flash memory in NAND arrays is stored in a series of *blocks*, which are the smallest erasable entities on a NAND memory chip. Within these blocks are *pages*, which are the smallest programmable entities on a flash chip. Pages include sectors, or chunks. Pages contain a data area and an area for memory management called the out-of-band (OOB) data. There is OOB data for each sector or chunk within each page that can contain metadata specific to the page's status (such as valid, invalid, or bad). The OOB area can also contain metadata on the associated page and block and does not have to be immediately following the sector (adjacent), but all can be at the end of the page (separate).

NAND memory chips do not last forever: their lifespans are measured according to the number of erases that occur. Typically, the time it takes to reach the maximum number of erases that can occur without memory chip failure far exceeds the lifetime of the mobile device. This measurement is very important with regard to the way the data is often written to the flash memory. Data is written to a flash cell as 1's and 0's: 1 represents empty and 0 represents full. So if data was written to a block, a 1 can be replaced by a 0, but a 0 can never be changed to a 1 to alter the data. The entire block must be written to another block, and the previous block is erased during another background process. This technique is a function of the flash memory and is typically called "wear-leveling."

Wear-leveling is a way for flash memory to ensure that areas of memory are not exceeding the number of erases too quickly—in other words, it "evens out" the erases over the surface of the flash chip. In turn, when a file is updated on the flash, because it is not possible to program the current page, it is completely rewritten to another location (a page, pages, a block, or blocks). This location does not even have to be in the same block or blocks. During this process, the OOB area is marked as active for the new page and the old page is marked as inactive. Another flash mechanism, called "garbage collection," is a function of reclaiming entire blocks if the number of inactive pages exceeds a given threshold. If this occurs, the entire active page or pages are written to a new block, and the entire old block is then erased to allow for new data to be written and become available for new data. These processes in turn extend the life of the flash memory.

Various types of traditional file systems (such as FAT, FAT32, VFAT, HFS+, EXT, EXT3, EXT4, and NTFS) and actual flash file systems (such as JAFFS2, YAFFS, and UFS) can be observed in the data from a chip-off collection. The difference between the two types is whether or not the file system needs a transition layer or whether it uses a system of databases to manage flash memory. For traditional file systems, there must be a Flash Translation Layer (FTL) that will operate as the interpreter for the file system and allow it to act as a block file system and emulate a flash file system. Different FTL interfaces use different specifications (such as MMC, eMMC, MSD, and ATA). As data is written to various areas of the flash to conduct wear-leveling and other procedures, the FTL presents the information to the traditional file system as though it had been written to a static location, so the traditional file system operates normally. On the other hand, a flash file system handles all the wear-leveling and creates its own data structures without the need of an FTL. These data structures are generally mounted into RAM and will contain flash information such as bad blocks, block erases, and pointers to files for the mobile device. This structure is written to the flash upon shutdown and then rewritten into RAM on startup.

When you put together the ways in which a NAND chip's memory is constructed—the built-in structures, wear-leveling, and garbage collection processes—and the ways other various file system types are constructed, it is no wonder that reconstructing the data from a chip-off examination is so difficult. Data collected by a chip-off examination can look disjointed and spread across the entire image. When this type of data, commonly referred to as "non-contiguous" data, is used, it makes recovering the data extremely time intensive on some mobile device file systems.

It's important to recognize the way flash memory moves data on the chip surface to prolong the life of the medium. With the function of traditional file systems and the flash translation layer, along with flash file systems, the fact that data is often written multiple times across the medium provides more opportunities for the examiner to recover forensic artifacts.

Traditional Tool Matrix

Table 8-5 indicates logical, file system, physical (non-invasive and invasive), and limited support features of several forensic tools. Limited support means only a small number of devices (such as Android and iOS only). For brevity, only the most popular tools currently in use are listed. NIST maintains a large list of forensic tools on its web site, including most mobile device forensics tools on the market (www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt/cftt-technical).

TABLE 8-5 **Traditional Mobile Device Forensic Tool Classification**

Tool	Logical	File System	Physical (NI)*	Physical (I)*	Limited Support
BlackLight/Mobilyze	X	X			X
UFED 4PC	X	X	X	X	
Device Seizure	X	X	X	X	
MOBILedit Forensic	X	X			
Magnet AXIOM	X	X	X	X	
Oxygen Forensic Detective	X	X	X	X	
XRY	X	X	X	X	

*(NI = non-invasive; I = invasive)

Tools Available

A plethora of mobile forensic solutions are available today, from open source software, to freeware, to commercial tools. As previously discussed, the tools used in the forensic laboratory and in the field are those that can be used on the majority of devices observed in the area of operations. The tools discussed in the following sections cover the majority of devices, have the most robust feature set, and complement one another. These tools will provide the capability and versatility you need when you're conducting mobile forensic examinations.

Open Source Tools

Open source tools allow for the viewing, changing, altering, and sharing of the source code by anyone. Any restrictions depend upon the licensing that accompanies the distribution. Most open source mobile device forensic tools are geared toward smart devices, iOS, and Androids. BitPim and TULP2G, for example, handle feature phones, but both tools are no longer being updated.

Let's begin by discussing the actual usage and philosophy of open source tools. There has always been a rift in the forensic community regarding the use of open source tools versus commercial and freeware (closed source) tools. In 2003, Brian Carrier published "Open Source Digital Forensics Tools: The Legal Argument," which discusses the forensic community debate that is still ongoing today and hinges on the integrity and reliability of the data and support for the software. There is no clear winner in this debate. Many examiners believe strongly one way or the other and cannot be swayed no matter the argument. However, both types of software can coexist in today's mobile device forensic toolkit.

Using an open source tool for the collection of a mobile device can be of great benefit because the collection of the device is 100-percent verifiable and transparent. That is difficult to say about a closed software solution, which is often an Achilles heel for the examiner who has no idea how the tool extracted the data from the mobile device—it just simply happened.

Tip

At times, the use of open source tools is advisable simply because the mobile device image creation can be closely observed. Because device image creation is often challenged, having the source code to gather specific intelligence as to what is "under the hood" can help you dispel any allegations to discredit the image creation.

iOS Devices

iOS devices can be examined using several open source tools that work for both Windows and Linux as well. These tools all use the Apple Backup service to initiate a local backup of the device, just as a user would if using iTunes. Some tools also use additional Apple services during the collection of mobile device data. All of the tools covered in this section allow access to the device without enabled security—that is, the passcode for the handset as well as the iTunes password must be known, if enabled.

Inflatable Donkey This software is used to extract iCloud backups up to and including iOS 11. This software does not parse and decode the software, but it downloads the specified databases contained in the user's iCloud account. These databases can then be viewed in many commercially available tools as well as some freeware. As of November 2017, development of the code ceased, possibly (rumored so) because of pressure by Apple on the availability and usage for nefarious purposes. The codebase is a fork of the iloot code that many current commercial tools' iCloud extraction is based on. The most current build is available on GitHub (https://github.com/horrorho/InflatableDonkey).

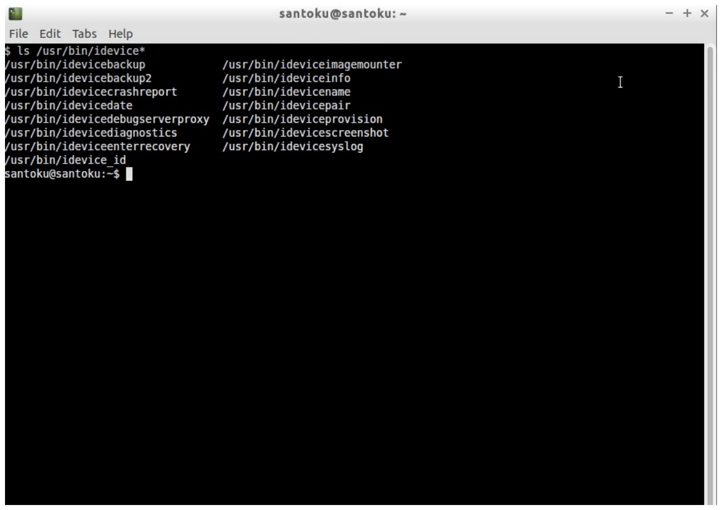
The following features are supported for extraction:

- SMS/iMessage
- Call history
- Address book
- Notes
- Media (video/images)
- Voicemail

Santoku Santoku is a suite of tools used for mobile device forensic investigations, mobile malware analysis, and mobile security assessment, all rolled into one interface using a Linux virtual machine (VM). Santoku has not been updated since September 2014 and will not work well with newer devices for both iOS and Android. However, there are tools within the kit that will offer assistance with mobile security testing and other app analysis methods. Santoku can be added to a Mac partition to dual-boot, or to use the preferred method and only suggested method for Windows, create a virtual machine.

The Santoku web site offers online documentation that walks an examiner through the installation of the materials and setup for both Mac and Windows machines. Using the tool, however, requires a lot from the examiner: to use most of the tools in the suite, the examiner must have knowledge of the Linux command line. Because of the degree of user interaction required, use of this tool is not recommended for those not familiar with command line–based programming. The suite of tools is compiled and all tools are installed, which allows for easy setup and usage, but, as indicated, it can be intimidating.

Both iOS and Android device collections are supported in the Santoku interface. (Android collection is discussed next.) For iOS devices, Santoku can collect, parse, and carve data. Collections occur via an iOS backup, using an open source library called libimobiledevice, shown in Figure 8-12, and then use iPhone Backup Analyzer to conduct the parsing of the iOS backup. iPhone Backup Analyzer is the previous version of iPBA2.



```
$ ls /usr/bin/idevice*
/usr/bin/idevicebackup            /usr/bin/ideviceimagemounter
/usr/bin/idevicebackup2           /usr/bin/ideviceinfo
/usr/bin/idevicecrashreport       /usr/bin/idevicename
/usr/bin/idevicedate              /usr/bin/idevicepair
/usr/bin/idevicedebugserverproxy  /usr/bin/ideviceprovision
/usr/bin/idevicediagnostics       /usr/bin/idevicescreenshot
/usr/bin/ideviceenterrecovery     /usr/bin/idevicesyslog
/usr/bin/idevice_id
santoku@santoku:~$ 
```

FIGURE 8-12 **Using the Santoku interface to start libimobiledevice to create an iOS backup**

Additional tools for parsing device data also are installed on the virtual machine. Both Scalpel and The Sleuth Kit are included (https://github.com/sleuthkit), which can be used to investigate the file system and conduct data carving methods. Both tools are also command line tools but can allow for analysis of the file system to locate files by type, create a timeline based on identified files, and check hashes of files against known hash libraries and custom libraries.

Android

Many more open source tools are available for Android devices, probably because Android source code uses some open source language and access to the Android development SDK is available online (http://developer.android.com). Because of the variety of devices on which the Android OS operates and variants of firmware on top of the various builds of the operating system, open source tools are sometimes more reliable for the collection of the device data. Quick code changes and frequent releases, along with high community involvement in addressing specific problems, also make open source tools popular. These frequent updates and code modifications are necessitated by the openness of the Android device OS. If a carrier uses a specific codebase or the device does not operate as the user wants, the user can change it! The "modding" community for Android is larger than any other mobile device community, with a mission to make the phone do what they want. This, in turn, offers open source developers looking to collect data, instead of change data, an opportunity to use source code already produced in an entirely different way. The following sections cover some of the tools that are used frequently in the forensic community.

Santoku As mentioned, Santoku is a Linux-configured VM and is a self-contained mobile forensic, mobile security, and mobile malware suite of tools. For Android devices, many more options are built into the application. For basic logical extractions, AFLogical Open Source Edition (OSE), developed by ViaForensics (which changed to NowSecure in 2015), is included in the Santoku edition. This tool is not longer updated and many scripts are outdated for extraction and parsing. I mention this tool simply for educational historical purposes.

Also included in Santoku are developmental tools that can be used to obtain a non-invasive physical device file system by using exploits such as Odin, Heimdall, and built-in fastboot methods. Note that these methods are extremely risky when they're used without proper training and guidance and can have catastrophic consequences to the mobile device data and the device itself. Santoku also contains several tools that enable the analysis of mobile device malware, including Androguard and Apktool to decompile Android application files, along with versions of Wireshark that can assist with network analysis of Android applications during dynamic malware analysis. Again, this virtual environment is not for examiners just starting out in mobile forensics, because it requires that the user interact with command line utilities.

All the tools that have been compiled under the Santoku environments are available for native installation and do not need to be run under the prebuilt environment. However, having the system already set up and configured is a plus, especially since control of the built-in applications, device collection, and data analysis often must be configured to run correctly.

Android Backup Toolkit The Android Backup Toolkit (https://sourceforge.net/projects/android-backup-toolkit/) is a suite of tools, including some that were developed by Nikolay Elenkov, who contributed to Santoku tools, for the backup and restoration of Android backup files.

Within the Android Backup Toolkit is the android-backup-extractor, which enables not only the extraction of the Android device using ADB but also deflates the backup into a tar (tape archive) file. Once the backup is converted to a tar file, the examiner can view and open the database files of applications and other files using freely available database browsers and file viewers.

A second application within the toolkit is the no-adb-backup-lister. As you should be aware, an app developer can place a flag (`"FALSE"`) in the manifest file for the application. This indicates that the database files will not be included when a backup is initiated. This information can be helpful in explaining why a tool did not extract information from an installed application. WhatsApp and Facebook Messenger are prefect examples of the developer using this flag.

A third tool bundled in the toolkit is android-timestamp-keeper. Unbeknownst to many examiners, and tool developers as well, is the fact that MTP mode, when extracting files from the SD card (internal/external), will alter the date/time of the files. Using the android-timestamp-keeper script will enable the modified time of the mounted partition files to be backed up and restored correctly.

ANDROPHSY ANDROPHSY (https://github.com/scorelab/ANDROPHSY) is a tool set that runs on Linux Ubuntu, much like Santoku, that utilizes several different dependencies including Java, Android SDK with Eclipse, and MySQL.

This tool set was developed as a forensic tool that contains case and evidence management, mobile device acquisition (logical and physical), analysis, and reporting. The software was first developed in 2015 and has not been updated. It can provide logical extraction of current devices but does not provide exploits above Android 4x, per the commitments on GitHub. This tool will enable logical extractions of current Android devices, however, as long as the current version of ADB is obtained and utilized.

Caution

Take care when running these tools, because some run directly on the device and may cause undesirable effects, especially if you're processing evidence. It is important that examiners test and become familiar with the underlying code within each of the modules.

BlackBerry: MagicBerry

MagicBerry is an open source tool (https://code.google.com/p/magicberry/) that allows for the parsing of both IPD and BBB files, which are created for data backup using BlackBerry Desktop Manager software. Newer backups that are produced by BlackBerry Link software are not supported by the MagicBerry code or any open source tools. The MagicBerry compile program can be downloaded from the MagicBerry web site (http://menastep.com/pages/magicberry.php). The software supports the parsing of the IPD and BBB file to extract several types of user data, including SMS, contacts, call logs, service book, tasks, memos, calendar, and media (images and audio). The information can then be extracted and exported in various formats. This software was not created with the forensic examiner in mind, but it does enable the recovery and parsing for presentation. The MagicBerry software has not been updated for several years, and with the change in format along with encryption, this software appears to be at the end of its life.

Freeware Tools

Freeware is software that is essentially free to use, but the source code is often not available. Most freeware code cannot be modified by the user, viewed by the community, or updated without the direct involvement of the freeware project software development team. Some of the following tools do have open source components, however.

iBackup Viewer

iBackup Viewer (www.imactools.com/iphonebackupviewer/) is a freeware tool that allows for data extraction and offers some reporting capabilities. The free version does not support the decryption of encrypted backups, though the company offers this in its paid version. iBackup Viewer supports devices up to and including iOS 11. The data that is parsed and shown in the interface includes the following:

- Contacts
- Call history
- SMS/iMessage
- Calendar
- Notes
- Voice memos
- Internet history (bookmarks, history)

- Media (phones/videos)
- App data
- Filesystem view
- WhatsApp messages

iBackup Viewer also enables the decoding and viewing of the iOS property lists (binary and XML).

iFunbox

iFunbox (https://i-funbox.en.softonic.com/) is a freeware tool for iOS devices. Apple iTunes must be installed on the device that is running iFunbox because the software uses Apple's device services and device drivers to communicate with the software. iFunbox enables access to the iOS device applications and media area for all iOS devices, and with devices prior to iOS 8, it enables access to the raw file system. If an iOS 8 or later device has been jailbroken, a raw file system is also available. This tool is especially useful for gathering the file system data to be examined later using another forensic tool. iFunbox does not conduct any analysis of the device contents; it simply enables the internal contents to be displayed and copied to a storage location without the use of iTunes. iFunbox is also very useful if iTunes cannot recognize the device to conduct a backup. This software is not a standard forensic tool and should be used only after testing and validating the processes used by its communication. All iOS devices are supported, and access to the device and its file system are available without a jailbreak. iFunbox's last update was in March 2017, which enabled support for the iOS 10 file system changes within the iTunes backup files.

Commercial Tools

A *commercial tool* is a closed source application that is available for purchase. Commercial tools covered in the following sections contribute both to the collection and the analysis of mobile devices in a single solution. A lot of commercial tools that can be used for mobile device forensics perform several tasks, although an examiner might need to use a second tool to complete the analysis if a particular feature is not available. (This was discussed earlier in the chapter as a basis for a multitool approach to mobile forensics.) An example would be the collection and viewing of property lists from an iOS device. One tool may collect the files, including plist files, but these files cannot be viewed within the application, so a second tool would have to be used for that. The commercial tools discussed here can also be used generally as a single solution if multiple tools cannot be purchased based upon the supported devices and feature set.

Note

The tools discussed are those that I am well versed with and have personally relied upon for both professional and research collections. The following information is not intended to outline how the tools should be used, provide guidelines on collections using the tools, or describe specific instructions in the analysis of mobile device data. These sections provide an overview of the tools, their features, and their general use for mobile device forensics. Also, the devices are listed alphabetically, not in order of preference. These tools will also be used throughout the processing, artifact finding, and analysis discussions within this book.

Cellebrite

Cellebrite (www.cellebrite.com/en/home/), located in Petah Tikva, Israel, produces several tools—UFED 4PC (Logical and Ultimate), UFED Touch 2 (Logical and Ultimate), and (Logical and Physical) Analyzer software—for the collection and analysis of mobile devices. The UFED 4PC, UFED Touch 2, and Physical Analyzer software perform differently depending upon the version that the examiner is using. If the examiner is using the Logical version of both the UFED 4PC and Touch 2, the collection support is for UICC (Universal Integrated Circuit Card), feature phones (GSM/CDMA), smart devices, and mass storage devices. The collected data types are logical only, without file system support. If the examiner is using the Ultimate version of the UFED 4PC or Touch 2, logical, logical file system, non-invasive, and invasive physical collections are possible. Both the UFED 4PC and Touch 2 Ultimate solutions come with the Physical Analyzer (PA) software, which is used to conduct the decoding and analysis of the collected data from either tool.

The PA software enables importing and decoding of mobile device images from JTAG, chip-off, and GPS devices, and it can decode and parse collected images created using the UFED 4PC and Touch 2 Ultimate. PA software also includes visualization of timelines, link analysis, communication analysis, malware analysis, and watch lists. PA can search globally using strings, text, proprietary device formats, regular expressions, and many other data types across the entire device image and can conduct data carving across the device image to uncover additional data from unallocated and logical file system areas. Custom scripting is via Python, and a built-in shell and plug-in feature are available. PA also has integrated malware scanning using Bitdefender.

Caution

Malware analysis using one vendor's product can be limited, because this type of signature analysis can be easily duped by malware developers. It would be much more prudent to scan the evidence using several types of malware-scanning software to maximize signature analysis.

MSAB

MSAB (www.msab.com) produces several mobile forensic products to include XRY and XAMN. The company is based in Stockholm, Sweden, and develops mobile forensic software to include XRY, XAMN, and XEC. XRY, the collection product, is sold as part of a modular suite, much like the tools from Cellebrite. They types of extractions you will be able to perform are based upon what modules are included in your version of XRY (Logical, Physical, or Cloud). MSAB takes it a step further, however, by including modules within modules!

XRY is divided into two types of products: XRY Logical and XRY Physical, and these divided into other modules. XRY Logical enables the user to conduct quick extractions (of iTunes backup, Android backup, Android Agent) and is geared toward "pump-and-dump" examinations. The XRY Physical add-on to XRY Logical enables the user to conduct password bypass of some Android devices, on-board memory chip reads, and other advanced mobile forensic tasks. If a user wants to extract data from the Cloud or from Chinese devices, for example, he or she would need to purchase the additional license.

XAMN is similar to Cellebrite Physical Analyzer and enables the user to look into the actual extraction from XRY for analysis as well as get into the hex level of binary physical extractions. The product is also divided into modules, called Spotlight, Elements, and Horizon, that are available via licensing.

XEC is the new module focused at centralized storage, control of extractions, administrator rights, and control of workflow. XEC is not included with the purchase of XRY or XAMN and contains additional modules: Director, Export, and Express. Each module in XEC is designed for the required workflow for administration.

Oxygen Forensics, Inc.

Oxygen Forensics (www.oxygen-forensic.com/en/) is based in Alexandria, Virginia, and develops both Oxygen Forensic Analyst (OFA) and Oxygen Forensic Detective (OFD). Both applications support the collection of UICC and legacy CDMA/GSM devices along with smart devices. Logical, logical file system, non-invasive, and invasive methods are supported.

Images from other tools such as the UFED series, Lantern, and XRY can be imported into both OFA and OFD to analyze the information further. Analytics, password analysis, malware analysis, and mobile app support are available within Oxygen Forensics software as well, making these applications good choices for examiners gathering group intelligence. File carving as well as deleted data and file searching over multiple images is also a supported feature.

Support of JTAG images from Android devices can be parsed for valuable user data within OFA along with a powerful SQLite database viewer. Oxygen Forensic Detective also contains a tool (Oxygen Forensic Cloud Extractor) to obtain information from cloud services to include Windows Phone, iOS, Android devices, and today's most popular apps such as WhatsApp, Dropbox, Twitter, Facebook, Instagram, and many more. All data recovered from any mobile device via the Cloud or directly from the device can be visually represented within a timeline, social graph, and aggregated view, along with a native user data view. Oxygen Forensic Detective also supports today's most popular recreational drones for extraction along with decoding, parsing, and mapping.

Magnet Forensics

Magnet Forensics (www.magnetforensics.com), located in Quebec, Canada, is the newest company in the mobile forensics world. The company focuses primarily on Internet artifact recovery and introduced Internet Evidence Finder (IEF) as a supplemental tool for computer forensic examinations. The tool was designed for computer examiners to find Internet evidence quickly—something that's often lacking in regular computer forensic tools such as FTK and EnCase. Banking on the explosion of mobile device forensic evidence, Magnet first gave away a free extraction tool for mobile devices. The output produced from the tool was then analyzed in IEF.

Recently, Magnet released a forensic tool called AXIOM. AXIOM is an upgrade from IEF, and allows for the extraction and analysis of mobile device data from the most popular smart devices, plus the import of file systems from other tools and binary images produced by invasive and non-invasive extractions (such as chip-off, JTAG/ISP). AXIOM recently upgraded the product to include cloud data extraction from within the interface. AXIOM enables the concurrent examination of data from computers, mobile devices, and the Cloud within the interface.

Chapter Summary

The mobile forensic examiner must have a critical understanding of the different types of collections and what can be collected using each method, and at what level, for success. This becomes clear when considering the multitude of tools that are available. Although tools may indicate the ability for logical collections of thousands of devices, upon closer inspection they often do not support a logical file system collection. This limitation can be a detriment when requested to unearth additional data from the mobile device that might be contained only in the mobile device file system. Being prudent in what a mobile forensic tool can accomplish and knowing its limitations before making a purchase or considering it for the toolbox are important. Understand your mission and choose the appropriate tool to accomplish it, but also reflect on what might be needed later—such as file system or partition collection. Remember that doing one collection at the onset of the investigation is preferred instead of doing multiple collections because of tool limitations.

The tool classification outlined by NIST is only a guideline, and the recommendations outlined in this chapter should be considered in the same way. Obtaining a device file system is paramount, and a subset of logical examinations that enable obtaining a file system should be included. Additionally, HEX and JTAG should be more clearly defined, as listed by SWGDE, in declaring non-invasive and invasive methods. Furthermore, the terms "non-invasive" and "invasive," as defined by SWGDE, should not describe only physical methods, as collecting data by means of removal of the device coverings and internal components, but they should also reflect the exchange and replacement of internal software ROMs. Performing a replacement and exchange of a portion of the device operating system to circumvent security and permissions to gain a physical binary image can be just as invasive as performing a chip-off, JTAG, or ISP procedure.

What is clear in the tool classification pyramid and in the text is the fact the progression up to the top of the pyramid should be directly related both to the technical training and the knowledge needed to perform the analysis. What is also apparent in the tool progression pyramid is the relational size of each level, which can also visually represent the number of examiners conducting the specific types of examination. In addition, the need for training as the examiner climbs to the top is also important. For an examiner to be successful at the level of chip-off, for example, he or she must undergo a sufficient amount of training and have sufficient knowledge in the area of device disassembly, flash memory physical architecture, flash memory internal architecture, and digital file systems. Without a level of competency that matches the level of examination, the examiner's success rate will be unimpressive, while the possibility of catastrophe is increased.

Many tools can be used in a mobile forensic examination, and the use of multiple tools is a necessity. Tools that can be verified and validated more easily include open source tools that enable the examiner to peer into the actual code to verify what is happening while creating an image or parsing the data. The use of open source tools has always been controversial, but usually the controversy is encouraged by commercial tool vendors. As long as the tool can be verified and validated, be it open source, freeware, or closed source, it will be usable in mobile device examinations and will stand up to court challenges. Use of commercial tools is a necessity, however, because these tools cover more devices than typical open source tools. This is primarily due to the concentration of the open source tool on a single operating system and sometimes a single device. This is not due to a lack of understanding in the open source community, but due to a particular developer or group concentrating on the device subset that they actually own! Development time costs money, and commercial tools will cover more device profiles and file systems.

Several tools were discussed at the end of the chapter that will be used for further collections and analysis throughout the book. These software applications are not the only tools available to the mobile examiner, as is plainly evident in the tools table. These tools cover the largest number of devices with the most built-in features, however. Obtaining a single mobile device solution set that can conduct collections, analysis of internal and external data, and complete reporting is often the preferred solution.

Your understanding of the levels of examinations, along with the levels the tools will support, will help you make the best decision as to which tools will best suit the examination at hand.

9 Preparing the Environment for Your First Collection

If your collection and analysis environment is set up properly, you are far more likely to extract and analyze forensic data successfully. If the environment is set up improperly, however, you can be confronted with many problems throughout an examination. Perhaps, for example, the collection device or computer is different from the analysis computer, and the optimal configurations on the two may be different as well. In a poorly configured environment, software may be ineffective not only at connecting to the device but also at extracting valuable data, leading to frustration.

Typically, software vendors include specifications on release notes or user guides that detail the computer configurations required to run their software properly. Sometimes, however, vendors' configurations are unrealistic for a working environment in which several different types of software are stored on the computer, and their minimum requirements are insufficient for running all the software. In addition, because forensic software solutions often write critical data temporarily as well as permanently, you need to know the location of this data, especially when you're conducting sensitive investigations.

Device drivers are also critical to a mobile device examination, and when properly installed, they provide the conduit to the device's digital data store. The mobile device examiner is often responsible for setting up the system at a scene location, so it's important that he or she get to know the devices. What happens when a driver is not installed properly? Obviously, communication to the device will not occur. But do all the drivers have to be installed? Many devices can also mean many drivers, which can hinder the computing environment's performance, especially in the Windows environment.

For the many reasons covered in this chapter, examiners often create virtual machines to operate each type of mobile forensic software in their toolboxes. This often eliminates the configuration problems that can be associated with each piece of software. Both NowSecure (which is built on Santoku Linux) and Santoku are Linux virtual machines that have been configured and fine-tuned to help eliminate some configuration problems, but they cannot solve all problems. Subsequently, Santoku Community Edition, sponsored by NowSecure, has not been updated since September 2014. In addition, some forensic software solutions cannot be run in a virtual machine and must be installed in a traditional, statically installed operating system. Not only do some mobile forensic solutions inhibit their software from running within a virtual machine, but devices such as iOS do not work well with virtual machines because of driver issues. When an Apple iOS device is plugged in, a driver installs, but when you're conducting a non-invasive physical collection of the device, additional drivers for Device Firmware Update (DFU) and iBoot must be installed on the fly. This installation flow gets caught up when the device drivers first attempt to connect with the host operating system before the virtualized system can take hold, often causing errors. Furthermore, additional drivers are required for the acquisition of Android devices at the physical level, but drivers are also required during the extraction process. For this reason, and others, some devices simply cannot be collected using a virtualized environment.

Note

Mobile forensic solutions that offer software installed on hardware, such as Cellebrite UFED Touch 2 and other embedded mobile forensic units, are not covered in this section, because these devices are self-contained hardware units that come preconfigured with device drivers. Other solutions from Cellebrite include the UFED 4PC and Physical Analyzer, which do require that device drivers be installed.

Creating the Ideal System

Computer specifications typically depend upon the analysis functions of the software that is installed on the system. If a software solution contains features such as data carving, database processing, searching, or other hardware-intensive functions, it would stand to reason that a more powerful computer solution should be used. Some software applications that extract data from a mobile device have little or no analysis capabilities, and a less powerful computer solution would be needed. The particular situation or mission will generally dictate the type of hardware and software configuration that a mobile examiner should use.

Most software vendors list the minimum specifications required for the base operating system, but few take into account the installation of other mobile solutions on the system. Experience has shown that the minimum specs that are listed for many vendors are too low for today's devices and, if used, can have negative effects on the overall collection and analysis process—even without the installation of another mobile forensic solution.

For example, consider the minimum specs listed for MSAB XRY Logical on its web site (www.msab.com):

- **Processor** Intel dual-core processor 1.6 GHz dual-core (Recommended 2.0 GHz dual-core)
- **RAM** 4GB
- **2GB hard drive space for the application**
- **500GB for the storage of data**
- **USB ports** Minimum two ports
- **Operating system** Microsoft Windows 7 64-bit (recommended Windows 10 64-bit)
- **Windows .NET 4.5**
- **Screen resolution of 1366 × 768**

The software will function and run with these specifications, but other processes must take place during the collection of the device, and with the low specs indicated here, operating system performance will be affected and the process can become frustrating.

The following sections offer general recommendations for building a mobile forensic environment that will efficiently run most mobile forensic software without issue. These are recommendations based upon a typical install running three mobile forensic tools simultaneously. Each additional software solution installed on the system will generally degrade the performance of the computer, so you should make adjustments as necessary.

Note

These recommendations are based upon multiple tools installed on the system per the guidelines outlined in this chapter.

Processor (CPU)

The processor, or central processing unit (CPU), type is integral to the success of your forensic software. The CPU runs the computations, functions, and I/O specified by the system, and its performance, measured in gigahertz (GHz), depends on several factors. The higher the *clock speed rate* of the CPU, the more functions it can perform each second. For example, a 3.4 GHz processor can perform 3,400,000,000 cycles per second, which is obviously more computations than a 1.8 GHz processor that performs 1,800,000,000 cycles per second. Although a higher clock speed rate means more cycles are performed, this should not be your only consideration in choosing a comparable processor, especially when it includes multiple *cores*. A CPU can contain multiple cores that enable simultaneous computations *n* times, where *n* equals the number of processors. So a computer with more cycles per second (GHz) but with a single or dual core may handle less information and run more slowly than a computer containing a processor with four cores and a slower clock speed.

Another important consideration in processor selection is choosing a 64-bit (x64) versus a 32-bit (x86) processor. A processor that is capable of x64 operations can handle twice as many bits (64) of information with the same clock speed as an x86 processor. A computer with at least a dual core capable of 64-bit processing should be the minimum recommended for mobile forensic software.

Note

Even if the mobile forensic solution can run on a 64-bit machine, that doesn't necessarily mean it operates in 64-bit mode. Some solutions can run as a 64-bit solution but operate only in 32-bit mode. Be sure to check the documentation to gauge your expectations regarding performance.

RAM

Random Access Memory (RAM) enables the processor to run computations on the flash memory rather than on the actual storage disk of the computer. When the processor places data into a computer's temporary memory, faster input and output result, which means data analysis can be completed more quickly. RAM is typically measured in gigabytes (GB) in today's computers. Mobile forensic software running in a Windows environment should have a minimum of 8GB of RAM, based on a single running instance of software at a time. Anything less than 4GB of RAM will not corrupt or invalidate the device collection, but it can cause software solutions to crash from out-of-memory exceptions.

By Microsoft's recommendations, Windows requires at least 1GB of RAM to run on an x86 system and at least 2GB to run on an x64 system, which gives the forensic software approximately 3GB of RAM to conduct the collection, analysis, or reporting—or all these functions simultaneously. If Windows cannot access the required RAM or is left with less than the required memory resources, the operating system will display the familiar "Blue Screen Of Death" (BSOD). If a BSOD occurs during a collection or an analysis, the information will be lost and the collection or analysis will most likely have to be redone.

Most software vendors in the mobile forensic field recommend that a minimum of 2GB of RAM should be used, but with only 2GB of RAM, even a small device containing approximately 10,000 records will have noticeable limitations. With most smart devices having hundreds of thousands of records that need to be processed using RAM, less than 4GB of RAM is not recommended, hence having 8GB will be sufficient.

Input/Output (I/O)

A computer's I/O throughput can also have a dramatic effect on the success and overall presentation of the forensic software. I/O involves not only the physical USB hub that is part of the computer but also secondary hubs attached to the computer. I/O is directed by the processor, and it's important that you understand how the controller and processor work together.

The USB controller and its USB hubs on a computer come in three types: 1.0, 2.0, and 3.0. USB 1.0 is the slowest I/O and USB 3.0 is currently the fastest. USB 1.0 is seldom found in newer computers these days, but many mobile devices communicate only at USB 1.0 speeds. It is recommended that at least three USB ports be available in a forensic workstation, including at least one USB 3.0 port. Table 9-1 shows information specifications from the USB Implementers Forum (www.usb.org).

TABLE 9-1 **USB Speeds as Determined by USB-IF**

Type	Speed
USB 1.1 (low bandwidth)	1.536 Mbps
USB 1.1 (high bandwidth)	12 Mbps
USB 2.0	480 Mbps
USB 3.0	5 Gbps
USB 3.1	10 Gbps

Simply having a USB 3.0 port does not necessarily mean that information will flow at a higher rate from a mobile device to the software during collection. The mobile device and the software conducting the collection will govern the speed of connection and data transfer. As specified, most devices communicate using USB 1.0 protocols. USB 3.0 and USB 2.0 are backward-compatible, so the connection between device and computer is not much different from that of USB 1.0 when the device is connected to one of these USB drivers. However, the speed of data output from the device to the collection computer can be noticeable. Obviously, collection speeds using USB 3.0 will be faster than those of USB 1.0 and 2.0, but if the device cannot output at a higher rate, the speed of the computer's USB hub will not matter. The speed is dictated by the device and software, not the computer's port.

Storage

Storage space and storage media are often underestimated by examiners seeking to collect and examine a mobile device, perhaps because the physical size of the device is misleading as to the amount of actual storage available and the number of computations and calculations completed by the mobile forensic solution. Today, the most popular mobile smart device's (iOS and Android) on-board storage averages 32GB. Most mobile forensic storage solutions need to be twice the size of the evidence when processing a single device because of the storage of the collected image and any temporary storage used during examination, processing, and analysis. You must consider portability, segregation of forensic tools, and I/O speed when choosing a storage medium for the mobile forensic solution and device processing. The use of multiple hard drives for the forensic analysis of mobile devices is not required, but it is often critical because of the many variables, processing overhead, and growing size of today's mobile devices. Apple iPad devices now come with up to 512GB of storage, and Android devices routinely carry 64GB SD cards and even up to 2TB cards, so having additional storage available for these devices is necessary.

The operating system, utility software, and forensic software should be installed on a hard drive devoid of any evidence or temporary data collected from a forensic tool. This drive should be used only to run the computer and associated software. By segregating the entire OS and software to a single drive, you can clone the drive and restore a pristine copy without compromising evidence or the integrity of the OS and software. In this way, forensic software and other related software can be maintained, changes and updates to the operating system or restores can be made without compromising evidence, and disk capacity does not shrink due to saving evidence images, temporary files, or other files relating to the processing and analysis of mobile devices. The operating system and software drive can be a traditional spinning hard drive (HDD) or solid-state drive (SSD) with a minimum of 250GB of storage.

A secondary drive should be used to store all temporary data, both from the operating system and forensic software. If using Windows, moving the location of the Windows pagefile
.sys file to another drive can also increase software performance. The Windows pagefile.sys file is used as a virtual memory swap file when a computer's RAM is exceeded and is often set to the same size as available physical RAM installed to the computer. If the pagefile.sys file is moved to an external drive, the operating system could crash if the drive is removed from the system, so moving the pagefile.sys file to an internal drive is recommended if the move is warranted.

The location of pagefile.sys file differs between Windows 7, Windows 8, and Windows 10. The following instructions show how to change the location of the pagefile.sys file for Windows 7, 8, and 10.

For Windows 7:

1. Press the Windows key.
2. Right-click Computer and then choose Properties.
3. Click Advanced System Settings.
4. On the Advanced tab, under Performance, click Settings.
5. In the Advanced tab's Virtual Memory section, click Change.
6. Clear the Automatically Manage Paging File Size For All Drives checkbox.
7. Under Drive [Volume Label], click the drive where the page file should be stored.
8. Check the Automatically Manage Paging File Size For All Drives checkbox.
9. Click OK.

For Windows 8:

1. Press Windows key-X, and then choose System in the pop-up menu.
2. Click Advanced System Settings.
3. On the Advanced tab, under Performance, click Settings.
4. In the Advanced tab's Virtual Memory section, click Change.
5. Clear the Automatically Manage Paging File Size For All Drives checkbox.
6. Under Drive [Volume Label], click the drive where the page file should be stored.
7. Check the Automatically Manage Paging File Size For All Drives checkbox.
8. Click OK.

For Windows 10:

1. Press Windows key-R, type **SystemPropertiesAdvanced** in the RUN box, and then press ENTER.
2. Click the Settings button in the Performance section.
3. In the Advanced tab's Virtual Memory section, click Change.
4. Clear the Automatically Manage Paging File Size For All Drives checkbox.
5. Under Drive [Volume Label], click the drive where the page file should be stored.
6. Check the System Managed Size checkbox and select the button to enable the OS to decide the correct size.
7. Click OK and restart the PC.

Note

Moving the pagefile.sys file is not mandatory, and not moving the file will not adversely affect the evidence you obtain during a mobile device collection and analysis. The recommendation to move the file is based on storage and I/O speed when the file is stored on the same disk with the operating system and forensic software.

The effect of temporary storage is not limited to the operating system, because it affects forensic software as well. Most software solutions use temporary storage to conduct different transactions against the evidence being examined to maintain a usable level of memory for the solution and the operating system. When the transactions are completed by the software solution, the temporary data is supposed to be removed from the disk by the software using that space, but this is not always the case.

Storing temporary data on a secondary drive enables the examiner to control the data, such as removing and purging sensitive data (company intellectual property, financial records, criminal case evidence, or other data), which is often part of the forensic examination. The drive should be dedicated to the temporary storage of the data from the Windows temp folder and pagefile.sys along with any forensics tools. This area can be sanitized after each forensic examination without compromising the operating system or installed applications on another drive within the forensic workstation. Because this secondary drive will be used by both the operating system and forensic software to store large amounts of data temporarily, a drive that is three times the size of the largest device that will be processed is required. Also, an SSD drive, if possible, should be used, which will provide a noticeable increase in both read and write speed during the collection and analysis of the mobile device data, as analysis tools used by mobile forensic software require large amounts of temporary storage to complete transactions.

A third drive should also be used for the storage of the evidence files from mobile devices either temporarily, semi-temporarily, or permanently. This drive is used only for the files and images classified as evidence: the original mobile device image, the output proprietary image from a mobile forensic solution if applicable, any generated reports, exported native files, case notes, case folders, and anything else that will be needed to complete a final report presentation and later archiving. This drive can be an external drive or internal drive with an HDD or SSD configuration. The appropriate size of this disk depends on its intended use: If it will temporarily house the evidence until the case is completed, the size is determined by the number of cases that will be typically concurrently worked. If the drive will store all of the evidence permanently until it can be deleted, a RAID solution can help protect against failing hard drives and data loss and allows for more efficient I/O functions.

External Storage

External storage is not attached to the device that is conducting the collection, computation, and evidence. It has two specific storage options: temporary and semi-permanent. Both are often required, but this is generally determined by the type of software being used and its features.

Using an external device for temporary storage is usually needed when the tool exporting the collected data sends it to an externally attached drive for later analysis, when the temporary data used by the device is later removed by the tool when the tool has completed its work, or in both cases. Tools such as Cellebrite UFED Touch specify that the examiner plug in a removable drive to complete the extraction, which then deposits the collected image and data to the attached removable storage. UFED Classic, which has now reached its end of life and is no longer sold (but still available on online outlets), was able to extract only to storage drives formatted with a File Allocation Table (FAT) file system, which often led to a large number of files for high-capacity devices because of the file size limitations for FAT-formatted devices. The UFED Touch and UFED4PC and Oxygen Forensic Detective can perform writes to NTFS drives, which allows for large file writes.

Note

Both UFED Touch 2 and Oxygen Forensic Detective can also export the data to a network or mapped drive if needed.

Once the data has been collected, it will be available at the location indicated by the examiner. Also, as discussed, having a secondary drive to write temporary data to is advised, and a removable drive can be used much like an internal drive. However, the I/O of a removable disk will be slower than that of an internal drive, and when a removable drive is used as a temporary location, there are noticeable limitations.

Tip

Some examiners also use the removable drive to store temporary data and export collected image files. Using this method is not advisable for obvious reasons, however, such as possible evidence contamination or inadvertent deletion when the software solution decides to purge the temporary data.

External storage can also enable archiving. When an image of a mobile device is created, it should be stored separately after a copy has been made. The copy image will then be analyzed. Typically, the storage of any electronic evidence should be on RAID to ensure redundancy and safeguard the image should a hardware malfunction occur with the external storage.

Operating System

Mobile forensic tools usually run on some version of Microsoft Windows. BlackBag BlackLight runs on macOS, but BlackLight can also run on Windows.

Note

Mac setup is not covered here because most examiners who use Macs for mobile device forensics use a type of virtualization that runs Windows. For those using Macs for mobile forensics, the information in this chapter can be used to set up the Windows virtualized environment, just as a user would on a system running Windows.

Some forensics software still indicates that Windows XP is a supported operating system, but using Windows XP is not recommended simply because newer devices and their associated drivers cannot run on XP and this version of the operating system is no longer supported by Microsoft. Also, software built using Microsoft .NET Framework versions later than 3.5 will not run on XP. Most mobile forensic software supports Windows 7, 8, and 10, with a recommendation by most to run Windows 10 (Microsoft extended support of Windows 7 will end in 2020). Experience conducting hundreds of examinations has shown that Windows 10 is much more efficient at communicating with mobile devices and both setting up and then removing drivers as necessary. Previous versions, such as Windows 7, did not remove device drivers from allocated COM areas, and some mobile forensic software had difficulty with communication. However, some device drivers, such as MediaTek, can be installed more easily in Windows 7 simply because of the driver enforcement rules in version 7 versus the newer Windows operating systems.

As mentioned earlier, with today's forensic processing and its demands on the operating system, if possible, use a 64-bit version of Windows, as long as the forensic software will support it. This will allow for more information to be processed and will address more memory than a 32-bit system.

Allowing automatic updates will also benefit the mobile device examiner; be sure to enable this within the software as recommended. Prior to installing any mobile forensics software, you should install all Windows updates and security patches.

Power functions can also cause difficulties while processing a mobile device. Most communication between a mobile device and forensic software solutions occurs via the USB port and a cable. If the power to the USB port is regulated by the operating system, the forensic solution cannot communicate properly and collection cannot occur. The power to the USB ports for both desktops and laptops is regulated by default in Windows. This enables Windows to turn off the USB ports when the OS believes that there is a need (such as during inactivity, low battery, or hibernation). In theory, Windows should wake the USB ports when a device is plugged in, but experience shows that this is not always the case, depending on the version of Windows installed in the forensic environment.

Tip

Troubleshooting this problem within Windows is covered in the "Troubleshooting the Device Connections" section later in the chapter.

Once the operating system has been updated and configured and all supporting software has been installed (but before any forensic software is installed), you should create a restore point. This restore point will enable the user to return to a baseline if needed. The following procedures show how to create a restore point in Windows 7, 8, and 10.

For Windows 7:

1. Press the Windows key.
2. Right-click Computer, and choose Properties.
3. In the left pane, click System Protection.
4. Click the System Protection tab, and then click Create.
5. In the System Protection dialog, type a description, and then click Create.

For Windows 8:

1. Press Windows key-X, and then choose System in the pop-up menu.
2. Click Properties.
3. In the left pane, click System Protection.
4. Click the System Protection tab, and then click Create.
5. In the System Protection dialog, type a description, and then click Create and OK.

For Windows 10:

1. Press Windows key-X, and then choose System in the pop-up menu.
2. Click Properties.
3. In the left pane, click System Protection.
4. In the System Protection dialog, type a description, and then click Create and OK.

Tip

If possible, use software such as Symantec Ghost to return the forensic computer to a point in time identical to when the image was created, which is unlike the Windows built-in Restore function. I suggest that you create a ghost image after all software is installed but before the first examination. This way, the machine can be brought back to a "clean" restore point after each examination to ensure a clean version before each new collection.

Device Drivers and Multiple-Tool Environments

Using multiple tools is important to achieve success in the world of mobile forensic investigations, but there can be consequences, particularly issues that occur with the unique drivers used by the various types of software to communicate with different mobile devices. Software vendors, at times, develop their own mobile device drivers that do not work with every piece of software that might also be installed on the system. As a result, when a mobile device is plugged in and a software-specific device driver is installed, other software solutions on the computer will be unable to communicate directly using a proprietary driver. This is immediately evident when an error is returned by the software solution indicating that the software was unable to communicate with the attached device. A quick check of the Windows Device Manager can identify the driver and the manufacturer. If the driver is not the original equipment manufacturer (OEM) version of the driver (for example, the equipment is Motorola but the driver is from and XRY or Cellebrite), you could encounter problems with all other software titles that use the OEM driver to communicate with the mobile device. Changing from the proprietary device driver to the proper OEM driver will typically solve the problem.

Some examiners create independent virtual machines for each software product and then run all the virtual machines on a single machine to combat a software-caused driver conflict. This enables all software to run independently and conflict-free using software-specific drivers. This can be a good solution, albeit a more advanced one, but unfortunately not all software will be able to run within a VM. To complicate matters, some mobile devices, such as iOS devices, which disconnect frequently within the VM, attach to the host operating system, which causes a collection failure.

The key to running a multitool environment, or even a single-tool environment, successfully is to understand how to control driver problems.

Understanding Drivers

Mobile device drivers are the gateway that allows hardware or software to communicate with the mobile device. If a device driver is not installed or is improperly installed, communication is generally impossible. Device drivers are not unique to mobile devices; they are also used for printers, scanners, cameras, motherboards, CD/DVD drives, USB controllers, monitors, and other computer peripherals and software. Anything that is not a part of the original operating system code base will need a driver to interact properly with the operating system. Device drivers are used in every operating system available and are undoubtedly one of the most common problems encountered in mobile device forensics.

Device drivers are themselves software applications that enable an operating system or another application to use hardware or other devices with common code and procedures to perform I/O. However, drivers are hardware dependent and also specific to an operating system. This means that, for example, a driver developed for Windows XP will generally not work if it's installed in any other version of Windows, and a driver developed for an LG device will not work with an HTC device.

A driver is like a switching station, where data from the operating system or application is interpreted upon arrival and translated to communicate to the device by way of the device driver. The device sends data back to the application and operating system as requested. A device driver allows a generic set of instructions from the operating system or application to request to read data, perform functions, and send data independent of knowing what device the software will be communicating with.

Here's an example dealing with mobile device forensic tools. Many different mobile devices are supported by a mobile forensic tool, but a single set of instructions is used to read data from a mobile device. Consider, for example, processing an Android device logically, when there are more than 18,000 different Android mobile phone devices on the market. How could a development team write specific code for each type of hardware? It is possible, but this would be an enormous task! By using a device driver unique to the Android hardware, the communication code is interpreted and translated to a format specific to the hardware connected by the driver. By using the device driver, virtually all the data collected will use the same communication code transmitted from the forensic solution for any Android device attached. The driver allows the forensic software to "speak" to the device using a single set of commands to enable the support of thousands of devices by way of the single driver.

Now that you have a basic understanding of what device drivers are and how they operate, your next challenge is finding the correct driver for the mobile device that needs to be processed.

Finding Mobile Device Drivers

When a mobile device comes into the lab for processing and has not been pre-installed into the mobile forensic solution, the examiner must locate and install the driver manually on the system before collection can occur.

Note

Some manufacturers, such as Cellebrite (UFED Touch), have pre-installed the device drivers directly to the hardware device they ship, so generally device drivers are not needed.

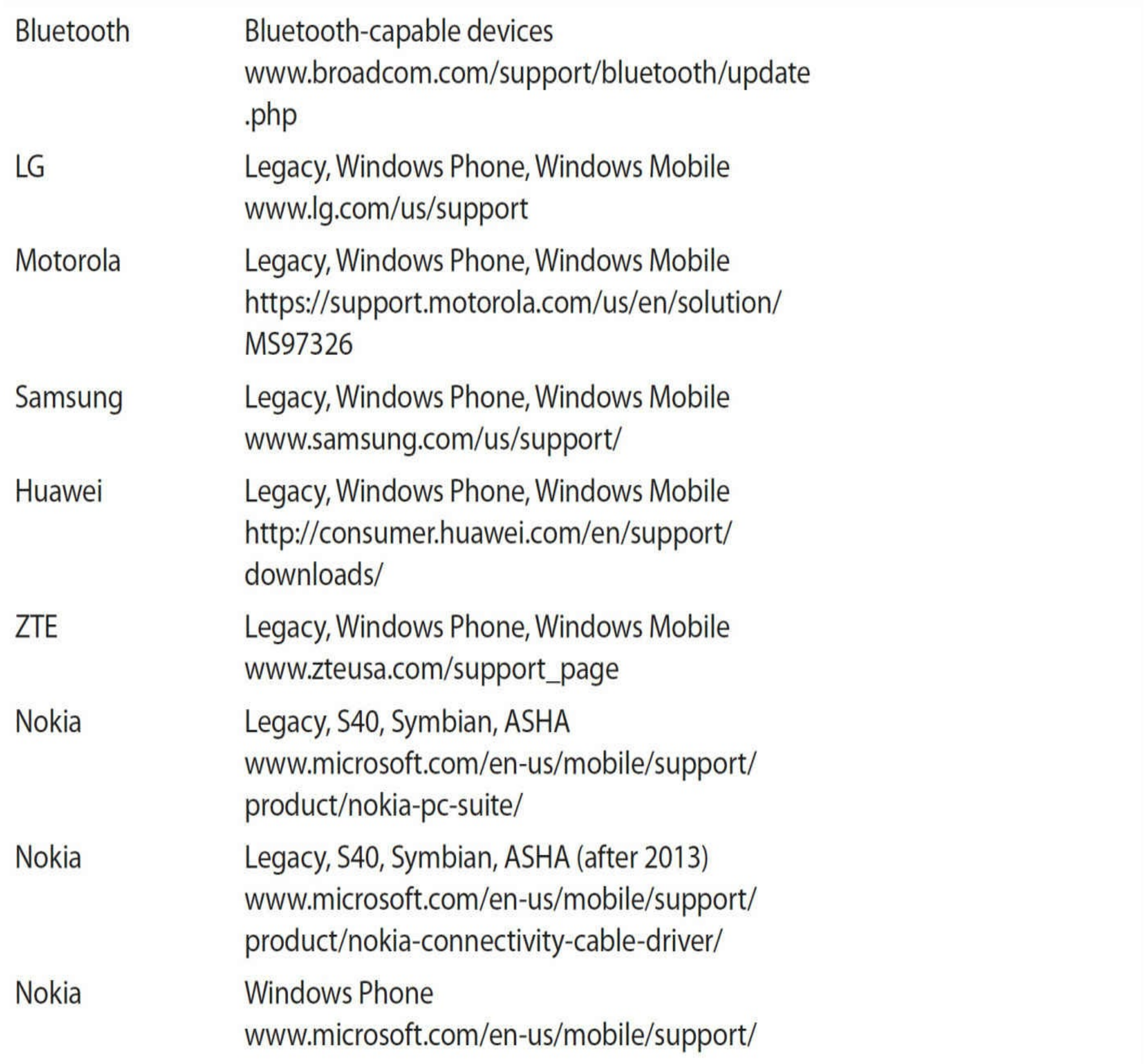
There are times when trying to locate mobile device drivers is like trying to find a needle in a haystack. Typically, mobile devices do not ship with a driver on a CD, as most other hardware devices (such as printers and cameras) do, and sometimes drivers are not readily available on the manufacturers' web sites. Nonexistent or missing drivers are more common with older feature phones, but even some newer Android devices do not have available drivers on the manufacturers' sites. Often, missing or nonexistent drivers are encountered with newer prepaid devices. In an effort to subsidize the cost of the device, manufacturers will allow the communication port to be disabled. By disabling the port for connections to a personal computer, a company can then offer additional services to transfer media or user data via online services, thus making additional revenue on data transfer that would not be possible if the user could plug the device into a computer and do the same thing. Because some of these drivers are needed and are not available on the manufacturer's web site, a hunt for the driver ensues.

Where does an examiner go to locate drivers if the solution does not have one? The answer is "it depends." It depends on several factors, ranging from the type of device, the manufacturer of the device, and the type of connection required to transmit the data to the operating system. Table 9-2 is not exhaustive, but it contains information on where to find drivers for the majority of devices encountered.

TABLE 9-2 Mobile Device Drivers for Device Manufacturers and Models

Manufacturer	Device Type and Driver Web Site	Credit
Acer	Android www.acer.com/ac/en/US/content/drivers	developer.android.com
Alcatel One Touch	Android https://www.alcatel-mobile.com/car-en/support/faq/usbdriver.html	developer.android.com
Asus	Android https://www.asus.com/support	developer.android.com
Dell	Android www.dell.com/support/home/us/en/19/products/?app=drivers	developer.android.com
Foxconn	Android http://drivers.cmcs.com.tw/	developer.android.com
Fujitsu	Android www.fmworld.net/product/phone/sp/android/develop/	developer.android.com
Garmin-Asus	Android https://www.garminasus.com/en_US/support/pcsync/	developer.android.com
Hisense	Android https:// https://www.hisense-usa.com/support/models	developer.android.com
HTC	Android www.htc.com	developer.android.com
Huawei	Android http://huaweiadvices.com/download-huawei-usb-drivers/	developer.android.com
Intel	Android www.intel.com/software/android	developer.android.com
KT Tech	Android www.kttech.co.kr/cscenter/download05.asp	developer.android.com
Kyocera	Android www.kyocera-wireless.com/support/phone_drivers.htm	developer.android.com
Lenovo	Android http://developer.lenovomm.com/developer/download.jsp	developer.android.com
LGE	Android www.lg.com/us/mobile-phones/mobile-support/mobile-lg-mobile-phone-support.jsp	developer.android.com

Motorola	Android http://developer.motorola.com/docstools/USB_Drivers/	developer.android.com
MTK	Android http://online.mediatek.com/Public%20 Documents/MTK_Android_USB_Driver.zip	developer.android.com
Oppo	Android www.oppo.com/index.php?q=software/view&sw_id=631	developer.android.com
Pantech	Android www.isky.co.kr/cs/software/software .sky?fromUrl=index	developer.android.com
Pegatron	Android www.pegatroncorp.com/download/New_Duke_PC_Driver_0705.zip	developer.android.com
Samsung	Android www.samsung.com/us/support/downloads	developer.android.com
Sharp	Android http://k-tai.sharp.co.jp/support/	developer.android.com
SK Telesys	Android www.sk-w.com/service/wDownload/wDownload .jsp	developer.android.com
Sony Mobile Communications	Android http://developer.sonymobile.com/downloads/drivers/	developer.android.com
Teleepoch	Android www.teleepoch.com/android.html	developer.android.com
Toshiba	Android http://support.toshiba.com/ sscontent?docId=4001814	developer.android.com
Yulong Coolpad	Android www.yulong.com/product/product/product/ downloadList.html#downListUL	developer.android.com
Xiaomi	Android www.xiaomi.com/c/driver/index.html	developer.android.com
ZTE	Android http://support.zte.com.cn/support/news/ NewsDetail.aspx?newsId=1000442	developer.android.com



Bluetooth	Bluetooth-capable devices www.broadcom.com/support/bluetooth/update .php
LG	Legacy, Windows Phone, Windows Mobile www.lg.com/us/support
Motorola	Legacy, Windows Phone, Windows Mobile https://support.motorola.com/us/en/solution/ MS97326
Samsung	Legacy, Windows Phone, Windows Mobile www.samsung.com/us/support/
Huawei	Legacy, Windows Phone, Windows Mobile http://consumer.huawei.com/en/support/ downloads/
ZTE	Legacy, Windows Phone, Windows Mobile www.zteusa.com/support_page
Nokia	Legacy, S40, Symbian, ASHA www.microsoft.com/en-us/mobile/support/ product/nokia-pc-suite/
Nokia	Legacy, S40, Symbian, ASHA (after 2013) www.microsoft.com/en-us/mobile/support/ product/nokia-connectivity-cable-driver/
Nokia	Windows Phone www.microsoft.com/en-us/mobile/support/

Note

URLs in Table 9-2 continually change. It is entirely possible the URL will not be correct or will be different from what has been documented at the time of publishing.

Installing Drivers

Once the examiner has located a driver, the next step will be installing it. Most of the drivers located on a manufacturer's web site include an installation program that makes it easy to install the device and then register it on the operating system, which enables the OS to place it into the proper Windows driver location so that when a device is plugged in, it is installed without user interaction. The method used by Windows when searching for and installing a driver for a new device is outlined in Table 9-3.

TABLE 9-3 **Windows Installation Path for Device Drivers (Microsoft Developers Network)**

Search Phase	Windows 7 and Later	Windows 8 and Later
Without User (Auto)	Windows Update, driver store, Device Path	Driver store, Device Path, Windows update
With User Interaction	None	None

When a device is plugged into a Windows computer running Windows 7 or later, Windows will automatically check the Microsoft Update Services, and if it cannot locate the driver, Windows will check the driver store. If the driver is not located in the driver store, Windows will look to the default system location of the INF (device information file) as specified in the registry HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion.

Note

Other operating systems are not covered here simply because most forensic software operates with the Windows OS, which subsequently experiences the most driver issues.

If the driver is located from the Windows Update query or from the default system location for INF files, it is then moved (*staged*) to the driver store and installed from that location. The driver store, introduced in Windows Vista, is a two-part process comprising staging and installation. To be installed, a driver must be staged, which involves moving the INF file and all the files referenced in the INF file and then verifying and validating the package. If all checks pass, the driver is copied into the store and installed without user interaction. If the driver does not pass (for example, files are missing from INF, the user does not have correct permissions, or the digital signature is not correct), the driver is not installed, and manual installation of the driver must then be handled by the examiner.

Note

Manually installing drivers for all versions of Windows involves locating the problem driver via the Device Manager and then updating the device with the proper driver.

Windows Device Manager

All examiners should be familiar with the Windows Device Manager, an application within Windows that enables the user to view and update device drivers on the system, check to see that the hardware is functioning properly, and modify the hardware's settings. How you navigate to the Device Manager depends upon the version of Windows in use.

Note

Only Windows 7, Windows 8, and Windows 10 are covered here because Microsoft no longer supports Windows XP.

For Windows 7 using Control Panel:

1. Click Start, and then click Control Panel.
2. Click Hardware And Sound.
3. Click Device Manager.

For Windows 7 using the command line:

1. Click Start.
2. In the search box, type **mmc devmgmt.msc**.
3. Press ENTER.

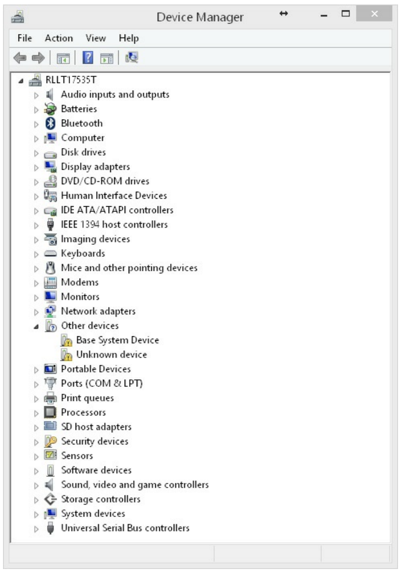
For Windows 8 and 10 on the main Start screen:

1. Press Windows key-C, and then click Search in the pop-up menu.
2. Enter **Device Manager** in the search box.
3. Select Apps and then click Device Manager.

For Windows 8 and 10 on the desktop:

- Press Windows key-X, and then click Device Manager in the pop-up menu.

Once you access the Device Manager, you'll quickly see any existing driver problems. Typically, the yellow exclamation prompt or question mark makes it clear that there is a problem with the installed driver or that a driver is not found. If a driver is not found for a device, the device will be installed in the Other Devices section, as shown in Figure 9-1. If the driver is installed, Device Manager will identify whether a problem exists with it. In both cases, you must install the correct driver before the forensic software can communicate properly with the device. To diagnose the problem and determine what steps will be needed, the Device Manager takes a closer look at what Windows indicates as the problem and the device as described by the operating system.

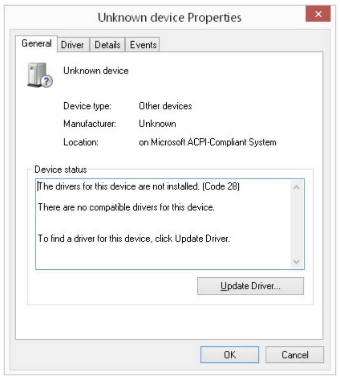


```
RLLT17535T
    Audio inputs and outputs
    Batteries
    Bluetooth
    Computer
    Disk drives
    Display adapters
    DVD/CD-ROM drives
    Human Interface Devices
    IDE ATA/ATAPI controllers
    IEEE 1394 host controllers
    Imaging devices
    Keyboards
    Mice and other pointing devices
    Modems
    Monitors
    Network adapters
    Other devices
        Base System Device
        Unknown device
    Portable Devices
    Ports (COM & LPT)
    Print queues
    Processors
    SD host adapters
    Security devices
    Sensors
    Software devices
    Sound, video and game controllers
    Storage controllers
    System devices
    Universal Serial Bus controllers
```

FIGURE 9-1 **If devices are not recognized or the driver cannot be found, they are located in the Other**

Devices section of the Device Manager.

In the Device Manager, locate the device with the exclamation point or question mark—for mobile devices, these are typically located under Ports, Modems, ADB, USB Controllers, and Portable Devices. There are always exceptions, but the most important information will be listed in the properties for the troubled device: Right-click the device and select Properties. A new dialog will open (Figure 9-2) with several tabs and a description of the device that you selected. Directly in the center of the first page under Device Status is the first clue to a problem or problems with the device. The displayed message will provide a short description of the problem and the associated code: in the figure, the message "The drivers for this device are not installed. (Code 28)" is displayed.



Unknown device

Device type: Other devices
Manufacturer: Unknown
Location: on Microsoft ACPI-Compliant System

Device status

The drivers for this device are not installed. (Code 28)

There are no compatible drivers for this device.

To find a driver for this device, click Update Driver.

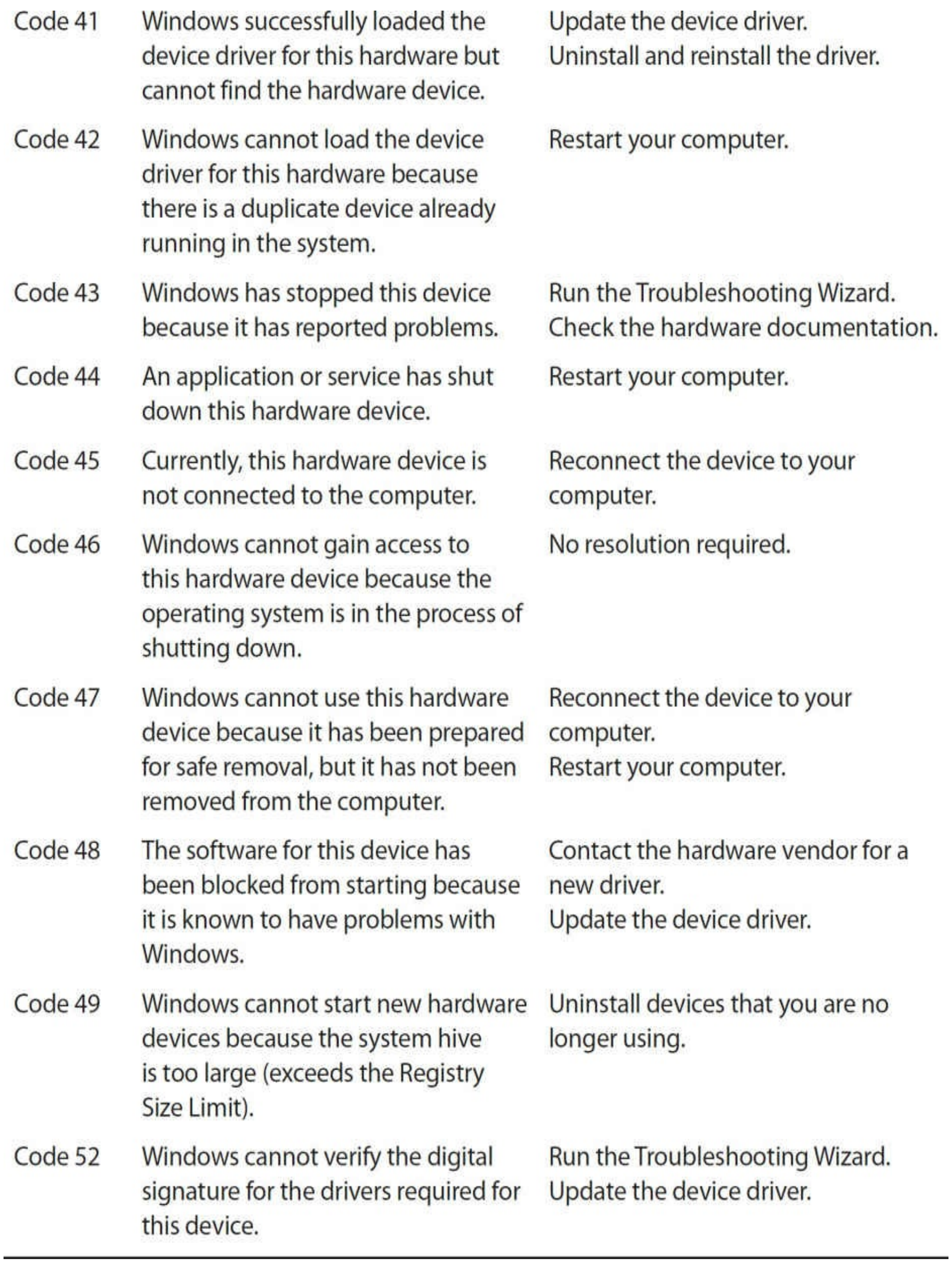
FIGURE 9-2 **The Properties dialog for a device in the Device Manager details the device and associated drivers.**

Many codes can be listed here, but typically the codes displayed for a mobile device are Codes 10 and 28. Table 9-4 shows a listing of all the codes used by Microsoft to describe problems in the Device Manager. This is not an exhaustive list, but it includes the most prevalent device problems.

TABLE 9-4 **Device Dialog for Each Device Describing the Status, Drivers, and Compatibility (http://support.microsoft.com/KB/310123)**

Code	Display Status	Fix
No Code	This device is not configured correctly.	Update the device driver.
Code 3	The driver for this device might be corrupted, or your system may be running low on memory or other resources.	Close some open applications. Uninstall and reinstall the driver. Install additional RAM.
Code 10	This device cannot start.	Update the device driver. Driver is for x86 or x64 only.
Code 12	This device cannot find enough free resources that it can use. If you want to use this device, you will need to disable one of the other devices on this system.	Disable the conflicting device.
Code 14	This device cannot work properly until you restart your computer.	Restart your computer.
Code 16	Windows cannot identify all the resources this device uses.	Assign additional resources to the device.
Code 18	Reinstall the drivers for this device.	Update the device driver. Uninstall and reinstall the driver.
Code 19	Windows cannot start this hardware device because its configuration information (in the registry) is incomplete or damaged.	Uninstall and reinstall the driver. Revert to the most recent successful registry configuration.
Code 21	Windows is removing this device.	Refresh the Device Manager view. Restart your computer.
Code 22	This device is disabled.	Enable the device.
Code 24	This device is not present, is not working properly, or does not have all its drivers installed.	Update the device driver. Remove the device.
Code 28	The drivers for this device are not installed.	Install the device driver.
Code 29	This device is disabled because the firmware of the device did not give it the required resources.	Enable the device in the BIOS.

Code 31	This device is not working properly because Windows cannot load the drivers required for this device.	Update the device driver.
Code 32	A driver (service) for this device has been disabled. An alternate driver may be providing this functionality.	Uninstall and reinstall the driver. Change the start type in the Registry.
Code 33	Windows cannot determine which resources are required for this device.	Configure or replace the hardware.
Code 34	Windows cannot determine the settings for this device.	Manually configure the device.
Code 35	Your computer's system firmware does not include enough information to configure and use this device properly.	Contact your computer manufacturer to obtain a firmware or BIOS update.
Code 36	This device is requesting a PCI interrupt but is configured for an ISA interrupt (or vice versa).	Use the computer's system setup program to reconfigure the interrupt for this device. Change the settings for IRQ reservations.
Code 37	Windows cannot initialize the device driver for this hardware.	Uninstall and reinstall the driver.
Code 38	Windows cannot load the device driver for this hardware because a previous instance of the device driver is still in memory.	Run the Troubleshooting Wizard. Restart your computer.
Code 39	Windows cannot load the device driver for this hardware. The driver may be corrupted or missing.	Uninstall and reinstall the driver.
Code 40	Windows cannot access this hardware because its service key information in the registry is missing or recorded incorrectly.	Uninstall and reinstall the driver.



Code 41	Windows successfully loaded the device driver for this hardware but cannot find the hardware device.	Update the device driver. Uninstall and reinstall the driver.
Code 42	Windows cannot load the device driver for this hardware because there is a duplicate device already running in the system.	Restart your computer.
Code 43	Windows has stopped this device because it has reported problems.	Run the Troubleshooting Wizard. Check the hardware documentation.
Code 44	An application or service has shut down this hardware device.	Restart your computer.
Code 45	Currently, this hardware device is not connected to the computer.	Reconnect the device to your computer.
Code 46	Windows cannot gain access to this hardware device because the operating system is in the process of shutting down.	No resolution required.
Code 47	Windows cannot use this hardware device because it has been prepared for safe removal, but it has not been removed from the computer.	Reconnect the device to your computer. Restart your computer.
Code 48	The software for this device has been blocked from starting because it is known to have problems with Windows.	Contact the hardware vendor for a new driver. Update the device driver.
Code 49	Windows cannot start new hardware devices because the system hive is too large (exceeds the Registry Size Limit).	Uninstall devices that you are no longer using.
Code 52	Windows cannot verify the digital signature for the drivers required for this device.	Run the Troubleshooting Wizard. Update the device driver.

Your next steps will be determined by the code and/or the description of the error listed in the Device Manager. This could involve finding the correct driver package for the device and running an executable program that installs the driver and reattaches the device. Or it might involve you manually installing the driver when an installation package cannot be located. Whatever the course, becoming familiar with the Device Manager in Windows should be one of your priorities.

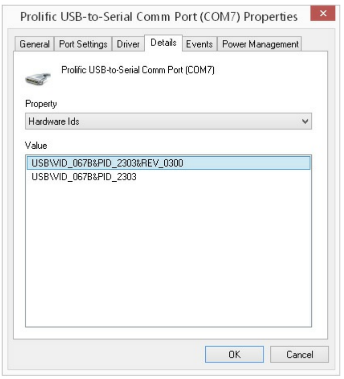
Tip

Something that has always been suggested, but rarely implemented, is placing a shortcut to the Device Manager on the desktop or taskbar of the analysis machine. By creating a shortcut, the examiner can quickly navigate to the Device Manager if any issues need to be investigated during a mobile device examination.

Identifying the Driver

If a problem exists with the driver and it is determined that the version of the driver is compatible with only a certain operating system version (such as x86 or x64), then generally the device properties will indicate, "This device cannot start. (Code 10)," but it will install. On the other hand, if a driver could not be located by the operating system and the installed device does not have a driver, it will be listed under Other Devices in the Device Manager with a message, "The drivers for this device are not installed. (Code 28)." In either situation, to find the correct driver, you can determine the device information by consulting the device properties, but this time using the Details tab.

The Details tab lists all the information about the device that is connected to the computer system. The important sections and tags for determining the correct driver are Hardware IDs and Compatible IDs. Both will help you locate a suitable driver for the attached device, and both contain valuable information in determining exactly what device is attached to the computer. An example in Figure 9-3 shows a Hardware ID with several lines that are important to the overall meaning; each will need to be broken down. Every USB device has two codes that distinguish it from other USB devices: the Product ID (PID) and the Vendor ID (VID).



Prolific USB-to-Serial Comm Port (COM7)

Property: Hardware Ids

Value:

```
USB\VID_067B&PID_2303&REV_0300
USB\VID_067B&PID_2303
```

FIGURE 9-3 **The prolific cable properties table under the Details tab identifies the assigned PID and VID for the device.**

- **PID (Product ID)** This is an essential part of every driver and must be used. This indicates the type of product or a family of products (such as modem, serial port, human interface device). The PID does not have to be unique and can be anything that a vendor assigns.
- **VID (Vendor ID)** This is also an essential part of every driver and must be used. A VID must be granted by the USB Implementers Forum, which assigns and maintains all VIDs as part of the compliance program. Sometimes this may not be the vendor of the mobile device. The VID could identify the chip

vendor that is either embedded into the cable itself or the silicon chip embedded in the mobile device, but a manufacturer has cooperated with the vendor and conformed to the driver specifications.

Tip

You can determine whether a driver is needed for a cable by plugging in the cable without the mobile device attached. If a driver is installed, or a driver install is attempted but fails, then hardware is embedded in the cable. The typical chips embedded into cables are Prolific or FTDI. If you do not see drivers install when you plug in the cable, check the cable; it may be for charging only and not for data transfer.

Once you've obtained this information, you can search both the VID and PID on the Internet to identify the correct driver needed for the mobile device. When you locate the package, you must download and install the INF and associated files manually.

Troubleshooting the Device Connections

At times, a device will not communicate with the forensic software, even when the correct driver has been installed. This might be encountered in the following situations:

- Multiple tools accessing the same device are running concurrently.
- The computer is allowed to manage the power to the USB hub.
- The computer is using a USB hub with multiple devices attached.
- The mobile device does not have enough power.

When multiple software processes are accessing a device, the device will be able to respond only to one at a time. I call this "COM envy"—if a communication port (COM) is open by one piece of software, it will not be released until that software either completes the tasks or releases the device or the software and processes are stopped and closed. Because of this limitation, another software solution cannot connect to the mobile device. When collecting mobile device data, you must have only one software solution running while initializing a connection and conducting an extraction.

Power management of USB hubs can also cause device connection problems, often disabling the connection and causing a collection error. Windows 7, by default, turns power off to the USB hub when the system senses that power needs to be managed. This is particularly noticeable with laptops that are not plugged in because of their power management profiles. There are two ways to manage the power management, but in all instances, the ability for Windows to manage the USB hubs should be disabled.

For Windows 7, 8, and 10:

This should be done for all USB root hub listings.

1. Click Start, or press Windows key-X for Windows 8.
2. In the search field, enter **mmc devmgmt.msc**.
3. Navigate to the Universal Serial Bus Controllers section at the bottom of the device listings.

4. Locate all USB Root Hub listings.
5. Right-click and select Properties.
6. Select Power Management.
7. Uncheck Allow The Computer To Turn Off Device To Save Power, and click OK.

A better way for both Windows 7, 8, and 10 will be to change the power settings via the power management console:

1. Click Start, or press Windows key-X for Windows 8.
2. In the search field, enter **edit power plan**.
3. Click OK.
4. Click Change Advanced Power Settings.
5. Scroll to USB Settings.
6. Expand USB Selective Suspend Setting.
7. Switch both On Battery and Plugged In to Disabled, and click OK.

Power to a USB device can also be inhibited when the power is distributed across multiple devices attached to a single external hub. When you're processing mobile devices, if you use an external USB hub that is attached to the processing computer, drivers may not be installed, and inconsistent connections can occur to an attached mobile device. If possible, attach a mobile device directly to the computer when conducting the mobile device collection, not through a USB hub.

Lastly, the power of the mobile device being examined can also cause problems with device driver installation. If the mobile device does not have at least 50 percent power, when you attach it to the collection computer, a failure to recognize and install the device driver is possible. During all collections, a mobile device should have at least 50 percent power before you connect the mobile forensic solution and throughout the entire extraction process.

Cleaning the Computer System of Unused Drivers and Ports

Windows, in an effort to be efficient, will cache all devices that have been attached to a Windows computer at one time with its assigned COM port number. What is considered efficient by Windows can have dire effects on mobile forensic software, however. This problem has been improved with the release of Windows 8, but it's still a problem. It boils down to *phantom devices* (or non-present devices)—hidden ports (devices) that are not shown in the Device Manager because the actual device is no longer connected with Windows. By default, users cannot see these hidden devices and will not even realize there is a problem. Even when an error is encountered by the mobile forensic solution because of the presence of a phantom device, most examiners are still unaware of the root of the problem.

The issue that arises with a phantom port and mobile forensic software is called *port enumeration*. Mobile forensic software enumerates through each port number that is used, looking for an attached device, and it will submit a challenge to the port to identify the device so communication can begin. When a computer running Windows has been used for mobile forensics, many devices get plugged into the USB port. When a device gets plugged into the computer and the driver is installed, a COM port is assigned to the device. When the device is unplugged, the device information, including the COM it was attached to, is saved so that when the device is plugged back into the same USB port, Windows will have already identified it and the same COM number can be assigned. This all means a couple of things to the examiner: One, if the device is plugged into any other USB port, it is assigned another COM number. Two, when any other device gets plugged in, a new unique COM number is assigned to that new device even if it had been plugged in to the same computer before. This means some computers can soon be operating with hundreds of assigned COM numbers, which can cause forensic software to time out while looking through all of the COM ports for a valid device.

Remove these phantom devices so problems do not arise when you're attempting to connect to a mobile device with a Windows-based forensic solution. You can manually remove the ports from the Windows command line, invoking a Registry change, by navigating to the Device Manager to show hidden devices and then removing each device by right-clicking and choosing Uninstall. They also can be removed with several open source automated tools that are currently available, along with CPD PortScrubber and Micro Systemation XRY, which include a port cleaner as part of their solutions.

Removing Non-Present Devices (Phantom Devices)

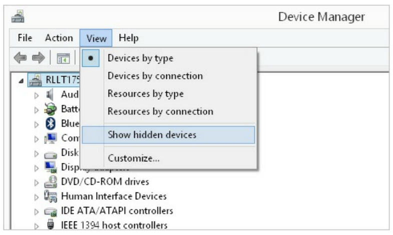
The manual method is covered here in an effort to help you understand what Windows is doing or what process is undertaken by the automated tools when they remove the phantom devices.

For Windows 7:

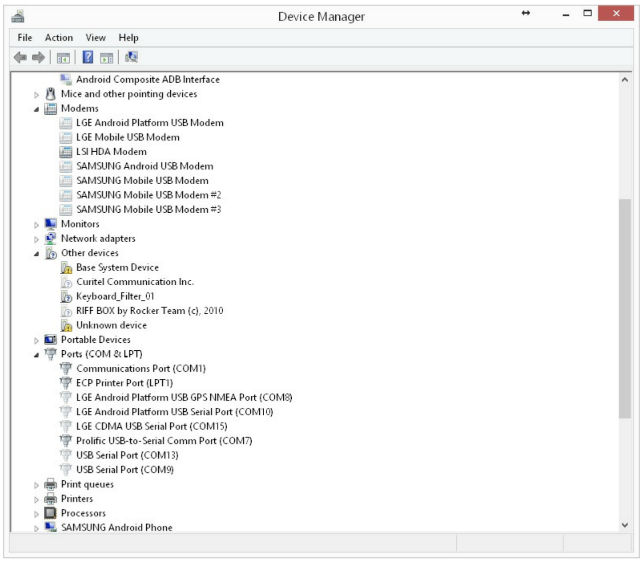
1. Click Start.
2. Search for **Command** and select Command Prompt.
3. At the command prompt, type the following, and then press ENTER:

   ```
   set devmgr_show_nonpresent_devices=1
   ```
4. Type the following command, and then press ENTER:

   ```
   start devmgmt.msc
   ```
5. Choose View | Show Hidden Devices, as shown next:



6. Navigate to Ports, Modems, USB Controllers, ADB, and other device areas where mobile devices have been installed.
7. Right-click any device that is "ghosted," or faded, as shown in Figure 9-4, and select Uninstall.



```
    Android Composite ADB Interface
  Mice and other pointing devices
  Modems
    LGE Android Platform USB Modem
    LGE Mobile USB Modem
    LSI HDA Modem
    SAMSUNG Android USB Modem
    SAMSUNG Mobile USB Modem
    SAMSUNG Mobile USB Modem #2
    SAMSUNG Mobile USB Modem #3
  Monitors
  Network adapters
  Other devices
    Base System Device
    Curitel Communication Inc.
    Keyboard_Filter_01
    RIFF BOX by Rocker Team (c), 2010
    Unknown device
  Portable Devices
  Ports (COM & LPT)
    Communications Port (COM1)
    ECP Printer Port (LPT1)
    LGE Android Platform USB GPS NMEA Port (COM8)
    LGE Android Platform USB Serial Port (COM10)
    LGE CDMA USB Serial Port (COM15)
    Prolific USB-to-Serial Comm Port (COM7)
    USB Serial Port (COM13)
    USB Serial Port (COM9)
  Print queues
  Printers
  Processors
  SAMSUNG Android Phone
```

FIGURE 9-4 **Several devices are shown as phantom devices under Modems and Ports (here, for example, Android Composite ADB Interface, Samsung Mobile USB Modem, LGE Mobile USB Modem).**

8. Do not check the box to uninstall the driver unless the driver is causing a conflict.
9. When finished, close Device Manager.
10. Type **exit** and close the Command window.

For Windows 8:

1. Press Windows key-X.
2. Select Command Prompt.
3. At the command prompt, type the following, and then press ENTER:

   ```
   set devmgr_show_nonpresent_devices=1
   ```
4. Type the following command, and then press ENTER:

   ```
   start devmgmt.msc
   ```
5. Choose View | Show Hidden Devices.
6. Navigate to Ports, Modems, USB Controllers, ADB, and other device areas where mobile devices have been installed.
7. Right-click any device that is "ghosted," as shown in Figure 9-4, and select Uninstall.
8. Do not check the box to uninstall the driver unless the driver is causing a conflict.
9. When finished, close Device Manager.
10. Type **exit** and close the Command Prompt window.

For Windows 10:

1. Press Windows key.
2. Type **Command Prompt** and press ENTER.
3. At the command prompt, type the following, and then press ENTER:

```
set devmgr_show_nonpresent_devices=1
```

4. Type the following command, and then press ENTER:

   ```
   start devmgmt.msc
   ```

5. Choose View | Show Hidden Devices.
6. Navigate to Ports, Modems, USB Controllers, ADB, and other device areas where mobile devices have been installed.
7. Right-click any device that is "ghosted," as shown in Figure 9-4, and select Uninstall.
8. Do not check the box to uninstall the driver unless the driver is causing a conflict.
9. When finished, close Device Manager.
10. Type **exit** and close the Command Prompt window.

Several automated tools are available to conduct this process simultaneously on multiple phantom devices, and some are freely available. Phantom device removal applications are generally based upon the Microsoft DevCon (https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/devcon) utility. DevCon, or Device Console, is a command-line utility that lists the devices on a computer running Windows. The utility can show phantom devices and remove them using several built-in commands and controls. By using the basic functioning of the DevCon source code, GhostBuster (http://ghostbuster.codeplex.com/releases/view/566397) enables the enumeration of phantom devices and aids in their removal with a GUI. GhostBuster is available for both x86 and x64 Windows systems. Prior to connecting a device to the forensic workstation, make sure that phantom devices are not consuming available COM port numbers. CPD PortScrubber is available from CPD Forensics. This is a commercial tool, but it works with Windows x86 and x64 systems up to and including Windows 10.

It is also recommended that you use the same USB port when conducting a mobile device collection. This will ensure that the driver that was installed previously for the same device is not installed again. By using the same port every time, Windows will already have reference to the device and will use the same driver that was used successfully the previous time the device was attached to the computer. If another port is used, Windows will search for the driver using the same process, depending upon the OS version, described in the section "Installing Drivers," and then install the driver to the device store.

Chapter Summary

How a computer system has been set up prior to installing mobile forensic software can ultimately determine the success of the software solution. If the hardware, such as processors, RAM, and hard drives, is not set up correctly, forensic collection and analysis will suffer no matter what type of software is running on the computer system. You can consult the ideal specifications outlined on the vendor's site and then follow the guidelines listed in this chapter to ensure that the software solution will perform well.

Simply using the minimum specifications listed on a vendor's site will generally not allow the software to run at peak performance, typically because the vendor tests when only a single solution—their solution—is running on the machine. A rule of thumb is to multiply the component capacities and requirements by two to get an idea of the ideal size system that should efficiently run the vendor software, but also any other solution that may be used in a multitool environment. By multiplying the specifications by two and allowing for additional software solutions, along with providing enough storage space for each solution's temporary storage and evidence storage, all solutions should operate well.

One of the most important, and often overlooked, problems with mobile device forensics is the device driver. By first understanding how device drivers are used, the examiner can make reasonable decisions as to whether the software solution simply does not support the device or the device is not installed correctly. Too often, examiners blame the software solution for failing to work, when the problem is actually based on a conflict between the operating system and the device. Because some device manufacturers update their device drivers regularly, it is a good idea to check regularly with a manufacturer's web site to make sure that the current driver used by the forensic solution is the newest driver for the device. Using Table 9-2, the examiner can at least have a starting point to identify the problem and then locate a device driver if communication to the device and the software becomes an issue. By understanding that some devices do not have an OEM driver and that each is identified by a PID and VID, an examiner can search for the driver using other methods. This can help uncover drivers for devices that could not be previously located by simply searching a device manufacturer's site.

When troubleshooting, you may find communication problems between software and devices, faulty drivers, software conflicts with other software, or an overabundance of COM ports. The examiner's ability to identify exactly where the problem lies takes a keen sense of understanding the different fault points. For example, when drivers do not install correctly, the forensic solution will be unable to communicate with the device. The examiner must be able to locate and update a driver manually. If multiple software solutions are installed on the same machine, and one uses a custom-built driver for a mobile device while another uses the OEM driver, an error will be caused in communication if the wrong driver is installed. Realizing this, the examiner can switch to the compatible driver instead of dismissing the failure as a problem with the forensic software. Lastly, COM ports are a way of life for communication to a connected mobile device. Clearing COM port numbers that are not being used and using the same port for all mobile device connections will substantially reduce the time Windows takes to install a driver and can also fix problems encountered by software solutions that iterate over all COM ports whether the devices are present or not.

Troubleshooting expertise is an important step in becoming a mobile forensic expert and can be a common practice when conducting hundreds of examinations using multiple forensic solutions. Understand a device connection, the proper setup for forensic tools, and common pitfalls associated with drivers, and you can minimize many of the frustrations involved in conducting mobile device forensic collections and examinations.

10 Conducting a Collection of a Mobile Device: Considerations and Actions

Now that the environment has been set and you understand some of the problems that may result, let's get into the actual processing and collection of the mobile device. As covered previously, connecting to the device to complete an extraction can be difficult at times, so starting with a clean system and arming yourself with knowledge of troubleshooting problems can keep your frustrations to a minimum. Once a device successfully connects, you'll find that collecting the digital data from the mobile device will be one of the more straightforward procedures in mobile forensics. Before we get into the connection and extraction of the digital data, let's look at some initial considerations.

As discussed in Chapters 2 and 5, the status of the device when it is received and how the request to process the device is made will often steer the type of examination that will occur and all subsequent steps thereafter. If the device is powered on when it comes to the lab or is located at the scene, the initial steps in the examination as well as the workflow could be much different than they'd be if the device were powered off.

The type of collection (such as logical or physical) of a mobile device is another important consideration. The type of collection required is generally determined by the type of investigation underway, the time needed or available, the device to be examined, the training and expertise of the examiner, and the tools available for the job. No matter what type of collection is used, proper isolation and documentation of the device and its components are necessary, because the processing order can be dictated by both. The type of mobile device that is to be examined can also determine how the collection should be approached.

Feature phones, also referred to as legacy devices, add a different dimension regarding preparation before the collection, but the actual extraction of the digital data from these types of devices is no different from that of the latest smart device. The setup of these devices prior to device collection is also important to the overall success of the digital data collection.

Alternative methods for collecting mobile device data are available for some devices such as BlackBerry, Android, Windows Phone, and iOS, and these should also be considered if traditional collection methods do not work or are unsuccessful; however, the initial preparation of the device prior to the data collection should be no different.

Initial Considerations

Before beginning the collection phase on the mobile device, the examiner must reflect on the various aspects of the collection, such as whether the examination will take place on the scene or in a lab. Additionally, the state of the device is important. Most devices arrive at the lab powered off. Sometimes, while on scene, the device is powered on and the examiner must immediately take steps to examine it.

Note

In most instances, whatever the state of the device at the time of collection, it must be powered on to be processed.

Prior to processing a mobile device of its digital contents, an examiner must consider other aspects of the examination. From documenting the state of the device in written form and in photographic form, to processing associated evidence, there are many nuances to a mobile device collection, and each must be considered. Also, the type of examination (whether logical or physical, as discussed in Chapter 8), must be considered. Conducting a physical collection outside the lab is not recommended, but it may be necessary based on the situation. To determine the need for and urgency of a specific type of collection, the examiner must consider how much time is available to complete the collection along with the type of data that is expected by either the requester or examiner.

Adhering to a sound methodology for all mobile device encounters will help solidify the resultant data and overall examination. The specifics are outlined in detail in the following sections of this chapter.

Isolating the Device

The examiner conducts the initial observation and subsequent processing according to the device state—powered on or off, or even in pieces. Whether the device is brought to a lab or the examiner conducts an on-scene examination, his or her first step should be to make sure that the device is isolated from any network connections. Device access to a network (such as cellular, Wi-Fi, or Bluetooth) can corrupt and even destroy the data.

If the device is powered on, it must be isolated immediately. Methods and appliances to isolate include but are not limited to mesh, bags, boxes, tents, rooms, and even signal disrupters (illegal in the United States, with exception of specific federal agencies). After the device is properly isolated, the examiner must be sure to keep the device isolated during the entire examination process.

Note

This section is not intended as a guide for the examiner in isolating the device; that information was covered in Chapter 6 and is continued later in this chapter.

If the mobile device is powered off at the scene or when brought to the lab, the examination should commence immediately using a systematic approach. The physical characteristics of the mobile device should be documented, and all peripheral evidence (such as a Universal Integrated Circuit Card [UICC] or media card) should be forensically collected.

Tip

It is good practice to keep a device in a signal isolated bag, even if it is powered off. Some devices can turn power back on if an alarm has been set, which could lead to a connection to a network and possible data loss or contamination.

Whether the device is powered on or off, the examiner must research to determine whether the device can be examined physically using a non-invasive method, which could include both software and hardware solutions. Armed with this information, along with the level of the examiner’s expertise, the current status of the phone (on, off, or locked), and the software that will be used, the examiner can decide on the next step.

Device Collection Type: Logical or Physical

The status of the mobile device upon reception or discovery often determines whether the device should be processed logically or physically. The type of examination is contingent on the device's power status and whether the device is locked, password protected, or disabled (disassembled or broken). Prior to conducting the initial collection of the device, the examiner must consider operational speed, available tools, and his or her level of expertise and training.

Note

Not all mobile devices can be collected physically. An iPhone 5 to X, for example, cannot be collected physically using non-invasive methods. The following suggestions are offered to the examiner with the full understanding that each type of device encountered must be approached in a unique way.

In general, a physical collection should occur first to provide a representation of the overall data on the mobile device and can validate a logical collection. During a physical examination, any information collected can be used as a foundation for the data from a logical extraction. This foundation is extremely important, because this is the information on which a logical collection will stand. With a physical collection, an examiner can clearly show where the information "lived" when it was extracted during a logical collection.

Tip

If a physical collection is not achievable, a logical file system extraction can suffice.

By the examiner first obtaining the foundational information from a mobile device with a physical collection, any information that subsequent tools or logical extractions produce can be directly compared to the device file system. Comparing the data collected from a logical examination with the data from a physical collection is powerful evidence, especially if data recovered corroborates or contradicts either collection. Having this type of information is extremely powerful to an examination and to the overall outcome of the investigation.

Conducting one type of collection *before* the other generally does not constitute a disaster or epic failure, but in most situations, conducting one *without* the other can jeopardize the investigation. What will determine the order of the collection type (physical or logical) are the circumstances dictated by the scene, investigation, time allotted, or request.

Note

Some examiners say that a physical collection should always precede a logical collection because of *wear leveling,* which arranges flash data so that write/erase cycles are distributed evenly among all of the blocks in the device. Simply reading data would not corrupt or overwrite existing data in a forensic examination, however, so this theory should not be the sole basis for conducting a physical collection of a mobile device prior to a logical collection.

Time Issues

Investigations that are not time sensitive or that have been submitted to a forensic lab should obtain a physical collection prior to a logical collection, when possible. As mentioned, by conducting a physical collection first, the examiner can rely on the underlying data to verify any information collected with the logical collection.

If an investigation depends on the examiner accessing the information quickly and in a format that can allow immediate follow-up, a logical examination should be first. With a logical collection, the investigator can immediately follow up on the information that is extracted from the device. A logical collection is also beneficial if the device loses power or is powered off and a lock is enabled. This quick collection can also assist if the device does not support a physical collection or if a physical collection is not possible for other reasons; the extraction can occur later either when time allows or upon arriving at a location that can conduct a physical collection safely and efficiently.

Device Power and Security Status

The power status of the mobile device should also be considered when determining whether to perform a logical or a physical collection first.

Tip

The type of investigation should be the primary directive in determining the type of collection, and the power status of the device should be secondary. If the investigation dictates immediate action, then whether a device is powered on or off does not matter. If the device is already powered on, process it logically first.

A physical collection should be attempted first if the mobile device is powered off when it is acquired. This can be accomplished with many feature phones if the connection to the device is via a USB or FBUS cable. These feature phones are often powered off while the device is placed into a special mode (such as flash, download, or bootloader) and cannot connect to a network. Also, if a device is powered off and it is a GSM (Global Systems for Mobile Communications) device containing a UICC, the card should be removed. When the card is removed from an iOS, Android, or Windows Phone, the device will be isolated from the cellular network and the examiner can attempt a physical collection using a USB and other non-invasive methods. With devices such as a Samsung SPH-D710, for example, an examiner can conduct a non-invasive Joint Test Action Group (JTAG) collection relatively easily with the device powered off. In this case, researching the device and the various ways to complete a physical examination are important before the examination begins. The physical collection should be a non-invasive physical technique, but the device can still be examined logically later.

If the device is powered on and unlocked, it should be processed logically first, because if the device is processed physically first and the device battery is removed or the device is turned off, the device can lock and a logical examination could be difficult. After the logical collection, the mobile device can be powered off for a physical collection and a locking mechanism will not be an issue.

A password-protected or locked device typically offers the examiner one initial collection method: physical. During a physical collection of the device, the examiner can hopefully obtain the passcode to access the device logically if required.

Note

Sometimes a device may be collected physically but the passcode cannot be obtained to unlock the device and conduct a logical collection. In these instances, a physical collection will suffice.

Device Damage

If a mobile device is damaged (such as by exposure to water, a broken USB connection, or a damaged electrical system or circuit board), the device should first be examined physically using non-invasive methods and then using invasive methods if needed. In some cases, the examiner has only one chance to obtain data from the mobile device, and the only connection that can be made requires using an invasive technique. Because a damaged mobile device is unlikely to be able to be examined both physically and logically, a physical examination may be the only option. However, every attempt should be made to collect the device logically after completing a physical collection.

Initial Documentation

The documentation stage, which should occur just prior to collection, enables the examiner to capture not only the device details but also all associated materials. Having photographic evidence of the device, UICC, battery, and media card not only enables the examiner to complete a detailed report, but it also helps when the device has to be reassembled!

Tip

Photographs and documentation of the initial collection should detail every angle so that there is no question as to which device a card or battery belongs with. In addition, when examining multiple pieces of electronic evidence, never pile the UICC cards together; this helps avoid the tedious task of identifying which UICC goes with which device later on.

Photo documentation should begin as soon as the evidence is received. The outside of each evidence container should be clearly visible in each photograph. In addition, the chain of custody form should be clearly visible, and if an evidence seal is used, it should also be photographed to show it has not been manipulated or tampered with. When the evidence is removed from the evidence container, it should be photographed immediately to include all pieces that are within the container; the associated case number should also be clearly visible, as shown in Figure 10-1. The same procedure should be completed with each evidence container prior to beginning the individual examination of the artifacts; the examiner should be sure to return the evidence to each container after completing the documentation and examination. This will ensure that evidence items from different containers do not commingle.

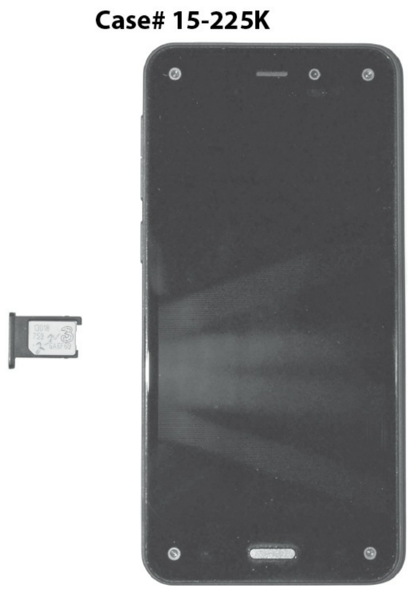


FIGURE 10-1 **A Kindle Fire mobile device shown alongside the UICC card, both a part of Case# 15-225, item K**

Each container may contain multiple artifacts, but a single artifact (such as a mobile device or media card) should be removed from one container at a time. Each item should be photo documented in great detail and examined individually prior to the extraction of data from the device. During the collection phase, only one piece of evidence should be out of the container at a time. This helps ensure that items that are not part of the current case or extraction are not introduced to the evidence for the current case; this can easily occur if multiple devices are lying around with loose UICC or memory cards.

The following sections offer descriptions and information regarding handling the various types of evidence encountered during a mobile device exam and the proper ways of documenting each artifact. Order of documentation depends upon the power status of the device, on or off, and typically defines the documentation phase. A diagram showing the process for documenting mobile device evidence is shown in Figure 10-2.

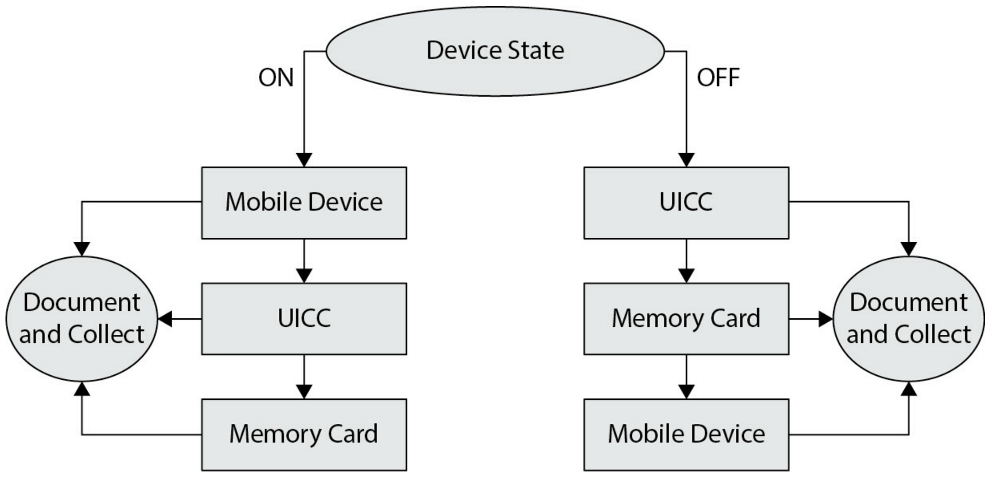


FIGURE 10-2 **Using the device state as a cue to the documentation phase can help the examiner develop a systematic approach while maintaining power to the device if needed.**

Device

If the device is powered on and isolated from the network, documentation should include a clear image depicting the device within the isolation environment; if the device is in airplane mode, the image can be taken outside of an isolation environment. Also, a photograph should document the current display of the device that clearly shows the device carrier, date, time, and desktop or main screen. Documenting the date and time on the device is extremely important during the examination phase of the extracted data. This information is often critical in an investigation; the owner of a collected mobile device can change its date and time in an effort to throw off the examiner, for example, if this detail is critical to text messages, e-mails, or device use. Without this information, the collected data from the mobile device can be extremely confusing.

Note

Documenting the initial screen was especially important with early-model Nokia mobile devices that lost the date and time and GMT (Greenwich Mean Time) offset when the battery was removed. Documenting the screen would enable the examiner to reset the date and time if needed to the proper time or use the offset information when converting dates that were extracted.

The back of the device and the USB connection should also be photographed to detail any imperfections, damage, or obstructions. This can help to dispel any allegations that the device was damaged during the examination process. Photographs should be taken of the device within its protective case, if applicable, and then outside of the case, including images of the front, back, and sides. It is important that the sides of a mobile device be photographed to show the location of a memory card slot. After a memory card slot is located, the protective flap should be carefully moved to photograph whether a memory card is inserted or not; the card should not be removed. With a device that is powered on, the collection phase of the examination would start at the completion of the device documentation.

Tip

Some devices, such as those in the Apple family, show clear model numbers on the exterior rear of the device. Be sure to document these numbers clearly.

If the device is powered off, the order of documentation is considerably different, with the exception of the initial photographs of the exterior of the device. The first few photographs should be of the front, back, sides, and USB connection with the case on (if applicable), and then with the case removed. These images will be used to identify the device and also document the current state of the device. If the device has a battery compartment, it should be opened and photographed. With the battery compartment exposed but the battery not removed, the examiner should locate and photograph the memory card and UICC slot if visible. Sometimes the memory card and UICC slot are under the battery, and the battery will have to be removed. Removing the battery should expose the device serial numbers, device information, memory card, and UICC if available (see Figure 10-3).

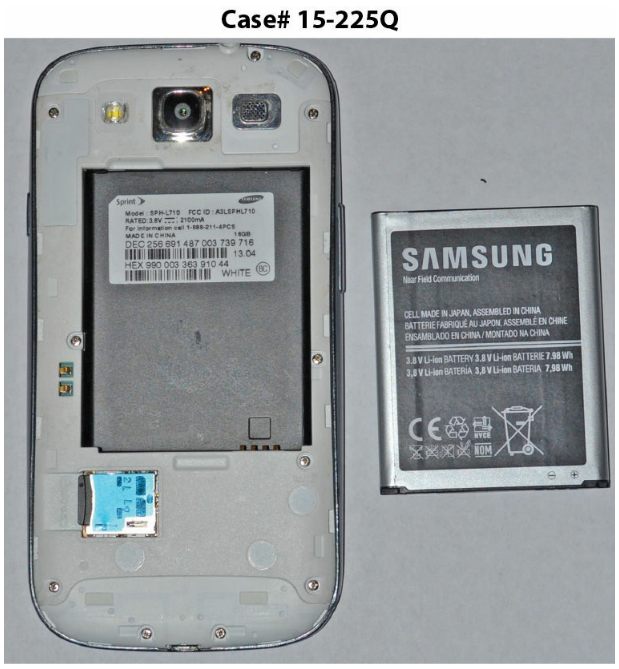


FIGURE 10-3 **The serial numbers of the device should be clearly visible, along with the memory card still in place.**

Tip

Some devices, such as those in the Apple family, do not allow the battery to be accessed and do not have memory cards.

Battery

If the device is powered on, the battery should not be photographed or accessed until after the device has been successfully collected logically. If the device battery is accessed while powered on, the device may lose power, and if security is enabled, the device may be locked. After a device that is powered on has been successfully processed, the examiner can logically access the battery compartment to complete the documentation, as described next when dealing with a device that is powered off.

If the device is powered off and the battery has been photographed in the device, the battery should be removed and photographed. The front, back, and sides should be photographed, and any distortions to the battery should be indicated. A failing battery should be documented, which is generally noticeable by a bulge in the center of the battery (Figure 10-4).

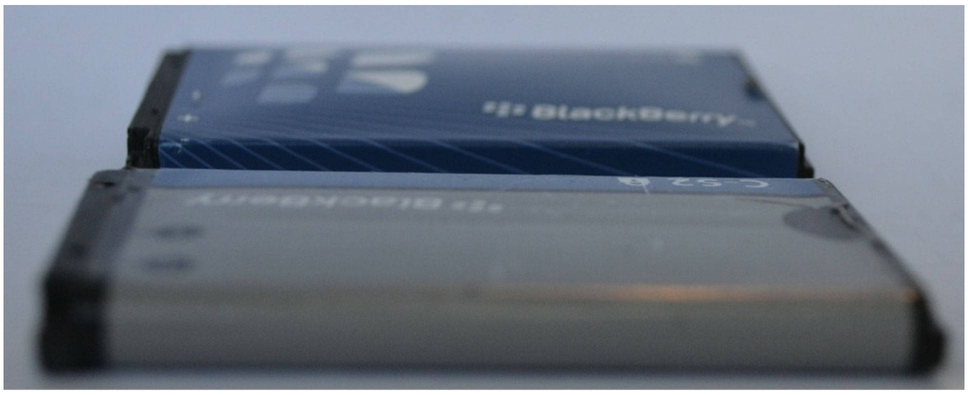


FIGURE 10-4 **A normal battery (front) and a bloated battery (back). The middle of the back battery is distended and could explode, causing damage to the evidence and possibly the examiner.**

Note

"Thermal runaway" is a phenomenon caused by the battery failing to dissipate heat faster than it is generated; the increased temperature of the battery distorts the plastic container, causing the battery to appear bloated. This was clearly identified as the cause of the recall of the Galaxy Note 7 phones in 2016.

Prior to starting an examination of the mobile device with a damaged battery, the examiner should insert a new battery into the device so that the damaged battery does not breach and subsequently damage the mobile device.

UICC

If the device is powered on, the UICC should not be removed until after a logical collection has been completed successfully. Access to most UICC cards is via the battery compartment, and removing the UICC card may allow the device to lose power. If a device loses power, and if security was enabled, the device may lock and a logical examination would not be possible. After a device has been processed logically, the examiner can follow the process to document the UICC card for a device that is powered off.

Note

Some devices can contain multiple UICC slots. If more than one UICC is located, all should be documented as indicated in this section.

If the device is powered off, the UICC may be in various locations, depending upon the device family. For iOS devices, the UICC slot is located on the top or side of the device. For most other devices, the UICC is in the battery compartment. The UICC location should be photographed in place prior to removal. Once removed, the UICC should be marked with the case number and evidence number if it has not been marked already. This marking should not obstruct the serial numbers on the exterior of the card. The UICC should also be photographed alongside the mobile device it was removed from; the UICC serial number, or the ICCID (Integrated Circuit Card Identifier), should be clear in the photograph. This documentation can help identify that the UICC is associated with the device if this detail is challenged later. Both sides of the UICC should be photographed along with the case number and evidence number. The UICC can then be processed with mobile forensic tools, and a forensic clone can be created, if needed.

Once processed, the UICC should be taped to the mobile device's exterior, not reinserted into the device, because this would enable the device to communicate with a cellular network if the device was somehow powered on when repackaged. If a UICC must be inserted into the device, use the created forensic UICC instead of the original card. By taping the original UICC to the device, the examiner can be sure that the card will not be misplaced during the examination of other evidence and that it is matched with the device to which it belongs. As outlined, mobile devices, wearables, and IoT devices may contain an internal UICC card, and currently the information must be obtained with common commands via the interface since the physical card cannot be removed.

Memory Card

If the device is powered on, the memory should not be removed until after a logical collection has occurred successfully. Most mobile forensic tools will allow for the collection of data from these memory cards during an extraction, so removing a card while the device is powered on could miss critical data.

Tip

Android devices containing external memory cards can corrupt the data on the memory card if it is removed while the device is powered on and not unmounted first. If the device is powered on, unmount the memory card prior to removal to avoid this issue.

Some memory cards are accessed via the battery compartment, and removing the memory card could cause the device to lose power and lock. After a device has been processed logically, document the memory card in the same way you would if the device were powered off.

If the device is powered off, the memory card location should be photographed prior to removing the card. The memory card slot is on the top or side of the mobile device, under the battery, or in the battery compartment. Once removed, the memory card should be marked with a case identifier and evidence number and photographed alongside the device. The media card can then be physically collected separate from the device and reinserted into the device when complete. If the device is going to be collected physically, the memory card can remain inserted; that way, if a logical examination is also performed, the memory card will be available.

Note

Remember that all photographs should include the case number and evidence number assigned to the article. Clearly photographing the progression of the collection will help for later documentation and also dispel any allegations of evidence tampering, destruction of evidence, or other issues.

JTAG, ISP, or Chip-Off

If a mobile device examination will be at the level of a JTAG (Joint Test Action Group), ISP, or chip-off examination, documenting the internal components is also important. For JTAG, ISP, and chip-off examinations, prior to disassembly, the examiner should follow the previously outlined steps to photograph the device and its components. Once the device is disassembled, the printed circuit board (PCB) should be photographed prior to making any connections. Once connections have been made to the JTAG test access ports (TAPs), a photograph should clearly document the connections to the ports. At the conclusion of the collection and removal of the connections, the printed circuit board should again be photographed and then again after the device is reassembled. For chip-off examinations, photographs of the memory chip should be taken prior to removal, after removal, after reballing (if applicable), and during a read of the memory chip.

Note

This amount of documentation may seem extreme, but showing the examination's progression and clearly showing that the chip in the device is the same chip that was collected can help immensely later if any challenges occur.

Mobile Device Isolation Methods

The techniques or methods used for isolating the device are often dictated by the device type. Whatever method or technique is employed; the isolation of the device must remain constant during the entire collection if the device is powered on. If the device can be collected while powered off, it is still a good idea to isolate the device if power is still available to it, because the device could be inadvertently powered on by the examiner or the software.

Several different isolation methods and techniques are covered in this section, from manually changing the device communication, to placing it in rooms devoid of radio waves. When selecting the type of isolation method, the examiner must remember that various factors can interfere with the technique's consistency and coverage. As outlined in Chapter 6, different frequency ranges are used by the various technologies built into a mobile device. The techniques used by the examiner should be tested on all frequencies encountered in the course of the examination.

If a device is not properly isolated from the network, data loss can result, because cellular, Wi-Fi, and Bluetooth networks offer conduits to the device's internal file system—a mobile device can pair with an active Bluetooth device in the examiner's office, or it can authenticate on an open Wi-Fi guest network in the area. Although the result may not be data loss, the integrity of the data will be changed with the addition of information unrelated to the case. If the device accesses a cellular network or the Internet, a signal submitted to the device can also remotely wipe all the information from the device, making any type of recovery next to impossible. By following the simple steps and techniques covered in this section, an examiner should not have to worry about these types of scenarios.

Chapter 6 covered the various communication methods that are used in today's mobile devices: cellular, Wi-Fi, Bluetooth, and near field communication (NFC) are used for both the reception and transmission of data. These services should all be inhibited during the collection process when possible using one or more techniques outlined.

Note

Some solutions, including Cellebrite UFED Touch 2, UFED 4PC, MSAB XRY, and Oxygen Forensic

Detective, among others, include a function for extracting data from devices using Bluetooth technology. If Bluetooth is inhibited, data transfer cannot occur from the device to the forensic solution. In these cases, the device Bluetooth must be enabled.

Methods, Appliances, and Techniques for Isolating a Device

There are various ways to isolate a mobile device from the network, and their differences are many. Isolation products range in cost from free to thousands of dollars. Some signal isolation products are portable; some are not. Some products require practice to use; others do not. Before deciding upon the best appliance solution, the examiner should vet it using real-world scenarios and testing, since many variables can affect even the most expensive appliances. Also, in some cases, an appliance is not needed and a method or technique can be used instead.

Before embarking on any isolation solution train, the examiner must test and retest the solution with live mobile devices where most of the mobile device collections are going to take place. If all device collections are conducted at a lab, testing is much easier. However, if the scene of mobile device collections is variable, the examiner must ensure that the solution is suited to isolating the strongest signal that may be available at the scene. What makes the selection of a signal isolation tool so difficult at times is that the installation of a new cellular tower by a local carrier can render a signal isolation device unable to protect the device from unwanted signals. The distance to the nearest tower can affect the usefulness of the solution, simply because the closer the device is to the tower, the stronger and more concentrated the signal will be. Also, if an isolation solution had been successfully tested with a third-generation (3G) device but the device to be examined is capable of using a fourth-generation (4G) signal—and soon fifth-generation (5G)—the solution might not be rated to block those signals, which ultimately can breach the supported barrier and reach the device. Many free apps for iOS (available from the iTunes Store) and Android devices (on Google Play) can read the radiofrequency (RF) signal strength in a particular area. Using these apps can help the examiner determine what course of action to take and whether a particular appliance will be effective.

Tip

An app should be thoroughly tested prior to using it in an actual forensic environment, where real evidence will be examined. Also, during testing, the app should be installed to a controlled device, not to the device that is to be examined.

Sometimes, the easiest solution when deciding on a proper appliance is not to use an appliance at all. Several chapters have mentioned placing a device into airplane mode, a technique that is the easiest to use and at times the most difficult to convince examiners to use. It is easy because all modern mobile devices have an airplane function, which inhibits all forms of communication, but it’s difficult for some examiners to embrace because the device must be manipulated—that is, buttons are pushed and screens are accessed. Some examiners believe doing this negatively impacts the device collection, but as mentioned in Chapter 6, placing a device into airplane mode is described in NIST and SANS documents as a viable technique to isolate a device.

If airplane mode is not available or the protocol requires that the examiner manipulate the device as little as possible, there could be another option. With GSM smart phones from Apple, Android, Windows Phone, and BlackBerry, the examiner can remove the UICC card and still process the mobile device. Without the UICC card, the device is effectively isolated from accessing the cellular network, but Bluetooth, NFC, and Wi-Fi can still be available. More information on processing mobile devices without UICC cards will be covered later in Chapter 11, but remember that the removal of a UICC card could mean the device must be powered off and any security feature that is active could be enabled. Because CDMA (Code Division Multiple Access) mobile devices operate primarily without a UICC, this method and approach will not work with those devices.

Tip

If security is enabled, airplane mode can still be accessed for the smart devices outlined in Chapter 5. Because security is already enabled, the device can simply be powered off, because turning off the device is not going to lock the device further. The examiner should then research the locked device to determine how to process it effectively.

If a device is powered on, the methods used to isolate it are similar no matter whether the device is GSM or CDMA, but they depend on security. If the mobile device does not have security enabled, both CDMA and GSM devices should be placed into a signal isolation appliance so that the examiner can visually see and manipulate the device.

Once inside the isolation appliance, the examiner can place the device into airplane mode and then remove the device from the appliance. Several available appliances will enable the examiner to see the device, push buttons, navigate capacitive screens, and even photograph it. Table 10-1 lists two appliances that have been used to process and interact with a mobile device in active investigations. Other isolation appliances that do not have the ability to connect to a computer are not listed in the table, but they are typically less expensive. At times, examiners have even used several wraps of aluminum foil or lined and unlined metal paint cans to isolate a mobile device during transport or while awaiting collection.

TABLE 10-1 **Appliances Used to Process and Interact with a Mobile Device in Active Investigations**

Name	Company	Description
STE3300F2	Ramsey Electronics (ramseytest.com)	Hardware case, large enclosure, power, USB 2.0 support, ability to interact with device from exterior
SFP1218	Select Fabricators (www.select-fabricators.com)	Soft portable enclosure, USB 2.0 support, viewable window, ability to interact with device from exterior

Note

A study, "A Field Test of Mobile Phone Shielding Devices," was conducted in 2010 at Purdue University that tested various RF-isolation appliances and mobile devices. The testing involved sending various messages and content to mobile devices while they were inside an appliance. No appliance tested passed at 100 percent signal blocking 100 percent of the time. Be aware of this when using any isolation appliance. Although the study is almost a decade old, it still holds valuable information on the lack of protection with some isolation techniques.

A signal isolation appliance enables the examiner to fix portable power, access external USB connections, conform to capacitive screens on today's mobile devices, and view the device through a transparent window. The appliance can be connected directly to a power source and a computer and easily monitored. This technique can be used when the examiner does not want to remove the device from the appliance or navigate the device to locate airplane mode. When the examiner can keep the device in the same isolation appliance, whether at the lab or on the scene, outside contamination by both radio signals and the various people who come into contact with the mobile device can be minimized.

Tip

Signal isolation bags are made of interwoven metal fibers that degrade over time, leaving gaps where signals can escape and enter. Because this degradation is not visible, it is important that you replace these bags frequently and handle them with care.

Some techniques and methods are specific to the type of mobile device (that is, CMDA or GSM). Because of these differences, each technology is discussed in a separate section.

CDMA Devices

CDMA devices can be isolated using appliance-based solutions (such as isolation bags and boxes) and airplane mode if they are powered on, but some CDMA devices can also be processed without powering them on, bypassing security. When connected with a USB cable that allows charging and data transfer, CDMA devices —feature phones, in particular—often have enough power to be recognized by the computer. Once the device is recognized, a driver is loaded, and the mobile device can be processed as if the device were powered on.

Tip

Carefully monitor the CDMA device when plugging the device into the computer, because some devices will start a boot-up process.

If the CDMA feature phone is powered off, removing the battery and then plugging the device into the computer with a charging cable seldom boots the device. However, when the battery is attached to the device and then plugged in, the phone booting sequence would begin. For smart devices of today, the device cannot receive enough power to allow the computer to recognize the device and load a driver, so this option is not available.

GSM Devices

For GSM devices, creating a forensic subscriber identity module (SIM) clone can effectively isolate a device from the cellular network. Because a feature/legacy CDMA device does not contain a card to access and authenticate on a cellular network, this process will not pertain to these devices, only to GSM. Why not just remove the UICC from the mobile device and in that way isolate the device? This would be the best course of action, and it has already been established that removing the UICC in a smart device such as an Android, Apple, BlackBerry, and Windows Phone still allows for the collection of the device's data. However, this is not always the case, especially with feature phones. By removing a UICC from a feature phone, forensic software is unable to communicate with the majority of handsets, and examiners are forced to take isolation of the mobile device seriously. Because the UICC must stay within the feature phone for a logical forensic examination to be successful, the device must be isolated correctly. This is why a forensic SIM clone is needed.

Note

The term "forensic SIM clone" was coined in a paper I wrote in 2008, titled, "SIMs and Salsa," which outlined the vulnerabilities of U.S. cell carriers and showed how a forensic SIM could be created without the original SIM card. Some of the content is used in this section to describe the process of creating a forensic SIM.

In essence, the examiner will create a SIM card that will satisfy the mobile device's need for specific files: the IMSI (International Mobile Subscriber Identity) and the ICCID. Here's an excerpt from my 2008 paper, "SIMs and Salsa":

> The two files we will focus on in this document are the ICCID … and the IMSI…. We will be focusing on these two important files because these files are needed to complete a Forensic SIM Clone™ and effectively isolate the device from the cellular network. To further elaborate, some handsets only need the IMSI, while some need only the ICCID and in some cases the device needs both the ICCID and IMSI.
>
> Mobile Forensics, Inc. coined the term Forensic SIM Clone because the completion of this method should not be considered in anyway a "clone" in the sense that the card is a duplicate of the original SIM. If that were the case then we have not effectively isolated the clone from the network because if it is a duplicate of the evidence all the network information still resides on the handset. If this network information still resides on the cloned card we know that the cellular network can add, delete or change data on the handset. Hence "Forensic" was added because all the network information is nonexistent.

The question asked by students and examiners alike is, "Why not just place a foreign SIM into the device so the device has a SIM and the forensic software can then process the device logically? Just place a T-Mobile SIM card into a T-Mobile phone, right?" Completing this type of process would result in dire consequences, however, because the mobile device would reject the SIM card and undoubtedly purge data, as the expected files (IMSI and/or ICCID) were not there. Most often, call logs would be deleted, but the deletion of images and videos was also observed on early iDEN (Integrated Digital Enhanced Network) model devices. To create a proper forensic SIM clone, the proper IMSI and ICCID must be obtained from the original SIM. If the examiner does not have the original SIM card or the original SIM card is locked, proper paperwork would have to be submitted to the carrier of record for the mobile device to obtain both the IMSI and ICCID. With both the ICCID and IMSI, a forensic SIM could be created manually onto a GSM test card and inserted into the mobile device. This, however, does not always work for the reasons outlined in another excerpt from "SIMs and Salsa":

> What if the original SIM card is PIN locked? SIM card is gone? Damaged? For starters, if the SIM is PIN locked you are limited by the inability of most logical software to process the device. Again, I say logical, we will discuss physical software shortly. We know if the SIM PIN is locked, the ICCID is still readable. This is due to the fact the ICCID does not need a security condition to be satisfied to be read. Reading the ICCID is easy, but how about the IMSI? Well, the opposite is true for the IMSI. The IMSI does have a security condition, the PIN. Unfortunately, if we do not have the PIN we cannot satisfy the condition to read the IMSI. Without the IMSI we cannot in confidence create a Forensic SIM Clone that the device will not think is foreign. So how do we obtain the IMSI?
>
> There are three methods that can help with this dilemma.
>
> - Send court order to cellular carrier
> - Obtain the IMSI from physical memory
> - Or create the IMSI yourself
>
> I am sure you are aware of the first solution. By sending a valid court order to the carrier they can supply the last IMSI utilized by the device and matches the ICCID you have supplied. The second way mentioned involves using specialized hardware/software to obtain RAW data from the device and then parse the data, recovering the last IMSI or last IMSIs of the SIM. The examiner can then manually type this information into the MFI Forensic SIM Cloner application and create a Forensic SIM Clone. I am sure you are wondering about the third situation, creating the IMSI yourself.

Here's an elaboration on the first condition outlined in the document in reference to obtaining a valid IMSI from the carrier: If a user contacts the carrier to obtain a new SIM card because the old card has been lost, the information from the old SIM will be lost and the information (IMSI) returned will be incorrect for the device. This would mean a failure in creating a forensic SIM clone, so obtaining information from a carrier does not always mean obtaining the correct information to compile a forensic SIM manually. Obtaining the IMSI from physical memory of a device was a better method, and once this was obtained, it could be used to create a forensic SIM card and obtain access to the device.

Note

To automate the process of both reading an original SIM and then writing to a test card, I created the MFI Forensic SIM Cloner (which is no longer available). All major commercial mobile forensic solutions now support the creation of forensic SIM cards when needed.

The creation of a forensic SIM card with today's investigations is generally not needed simply because the majority of devices that need to be processed and collected are smart devices. To reiterate, a smart device can be forensically collected without a UICC card, unlike legacy feature phones of the past. However, understanding how and why a forensic SIM should be created is critical, even if this technique is seldom required in today's examinations.

Mobile Device Processing Workflow

As you know, the examiner should follow a process for every mobile device examination. If proper isolation techniques have been followed, the recommended methods for processing a mobile device depend on whether the device is powered *off* or *on*. Several pieces of evidence can be collected from a mobile device, and each must be treated as a single piece of evidence during the processing phase. Figure 10-5 represents the suggested ordering of processing for various situations, and the following sections offer information on how to process the device in various situations.

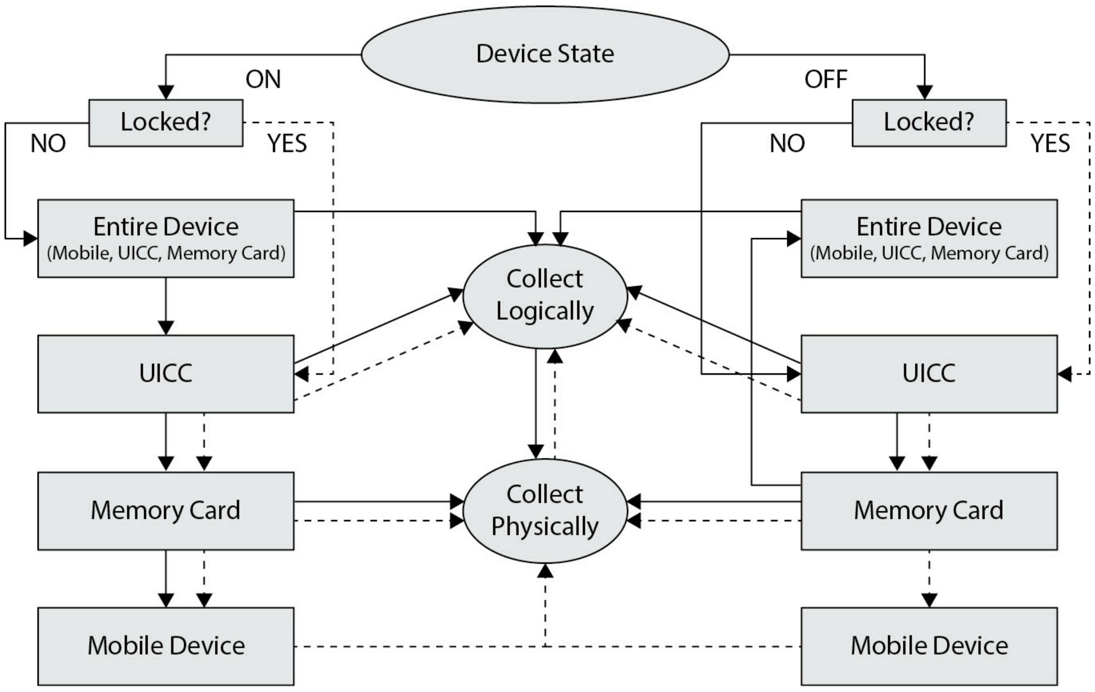


FIGURE 10-5 **Mobile device evidence processing workflow**

Device is powered on and unlocked Evidence processing should be conducted in this order:

1. **Mobile device** The mobile device should be processed logically, and when the examiner is satisfied with the results, it should be processed physically, if possible.
2. **UICC** The smart card should be processed, obtaining the complete file system.
3. **Memory card** The memory card should be processed outside of the body of the mobile device and a physical image created.
4. **Cloud extraction** Additional data is available that might not be part of the device collection, and every attempt should be made to collect the cloud services associated with the device and owner.

Device is powered on and locked Evidence processing should be conducted in this order:

1. **UICC** The smart card should be processed, obtaining the complete file system.
2. **Memory card** The memory card should be processed outside of the body of the mobile device and a

physical image created.

3. **Mobile device** The mobile device should be processed, physically if possible. If a physical image is obtained, obtaining security information for bypass from the physical image may be necessary to then conduct a logical examination.
4. **Cloud extraction** Additional data is available that might not be part of the device collection, and every attempt should be made to collect the cloud services associated with the device and owner.

Device is powered off Evidence processing should be conducted in this order:

1. **UICC** The smart card should be processed, obtaining the complete file system.
2. **Memory card** The memory card should be processed outside of the body of the mobile device and a physical image created.
3. **Mobile device** The mobile device should be powered on and, if not locked, the mobile device should be processed logically; when the examiner is satisfied with the results, it should be processed physically.
4. **Cloud extraction** Additional data is available that might not be part of the device collection, and every attempt should be made to collect the cloud services associated with the device and owner.

Device was powered off but is now powered on and locked If the examiner turns on a mobile device and it is locked, the UICC and memory card should have already been processed. The final processing would involve the mobile device and cloud services:

1. **Mobile device** The mobile device should be processed, physically if possible. If a physical image is obtained, obtaining security information for bypass from the physical image may be necessary to then conduct a logical examination.
2. **Cloud extraction** Additional data is available that might not be part of the device collection, and every attempt should be made to collect the cloud services associated with the device and owner.

Feature Phone Collections

Most collections of mobile devices today involve smart devices, but because feature phones are still available and in use, these devices are also considered here. In general, a *feature phone* is a mobile device that contains voice telephone calling, text messaging, and Internet features such as e-mail and web browsing, but it lacks the independent apps available for a smart device. A feature phone typically uses a built-in keypad or a number dialing pad, and the user must use a series of key presses to access alphabet characters. Feature phones do have built-in applications, but they lack the ability either to install or upgrade apps. These devices generally have a proprietary or embedded file system that does not allow outside developers to create applications, so they rely on built-in apps that ship with the device.

This does not mean that there is no available data to be collected from a feature phone—in fact, quite the contrary is true. As mentioned, these devices contain a file system, and within the file system are artifacts you could expect on a smart phone, with Internet histories, GPS coordinates, and multimedia messages. The difficulty is not in the collection, but in the decoding of the data. Because of the lack of documentation from manufacturers (for obvious reasons), feature phone data repositories can be difficult to decipher and decode.

Note

This book does not offer not an exhaustive list of feature phones, and some devices may support only the extraction of a small subset of the proposed list. The type of data that can be extracted varies, and the software that is used to complete the extraction should be consulted to see what data should be expected.

Connecting to a Feature Phone

The examiner should research to determine the best way to connect to a feature phone using available forensic software. Connection can be via USB, Bluetooth, or IrDA (Infrared Data Association). If the device has been isolated within an appliance, generally the only available connection method will be USB, so determining the software capabilities could impact the isolation mechanism, technique, or appliance. Once the connection method has been determined, the examiner should research whether or not the device must be placed into a certain mode to conduct a transfer to a mobile forensic solution. For example, to connect to a Nokia 5300 device and many other Nokia feature phones, the device must be placed into Nokia Mode by navigating to the appropriate menu on the device after a USB cable has been attached to the computer and to the phone. If the device is not in the correct mode, forensic software solutions will not be able to connect and collect the user data. The examiner should research data transfer methods and any specific settings by reviewing the model's user manual prior to beginning a collection. These manuals are easily located on the manufacturer's site, Phone Scoop (www.phonescoop.com), or GSMArena (www.GSMArena.com). Having a clear idea on how the connection will be made before beginning will save the examiner time when and if problems are encountered during the connection process.

Most connections to feature phones are made via a USB connector on the device. Feature phone USB connections have never been regulated, so many different types of connectors have been used, which can make connecting to the many devices difficult at times. All USB connections, however, communicate using specific pins on the USB head connected to the pins on the exposed USB connection on the phone. If any of the pins on either the cable or device are corroded, missing, or nonoperational, a connection cannot be made. The examiner should carefully inspect both the cable pin connections and the connections on the device to ensure a successful connection and data collection. If any debris is seen inside of the USB connection port, the examiner can use a small wire brush to clear it away. After a successful connection to the computer using a USB cable, the device is recognized and driver installation begins.

Note

If a cable with a serial-to-USB converter (such as Prolific or FTDI) is used, the examiner cannot be sure that a successful connection has been made to the device until a forensic software solution is used. Because the installed driver is for the cable and not the device, the examiner may be required to reexamine the device if the software cannot connect.

If a connection cannot be made and a driver does not load, the examiner can check the type of cable being used (see Figure 10-6). Some USB cables are for charging only and do not allow data transfer. Because specific pins are used to charge and others are used to transmit and receive data, a faulty cable may be a charging cable that does not have the Tx (Transmit) and Rx (Receive) pins. The examiner can exchange the cable with another cable that is known to be a data transfer cable.

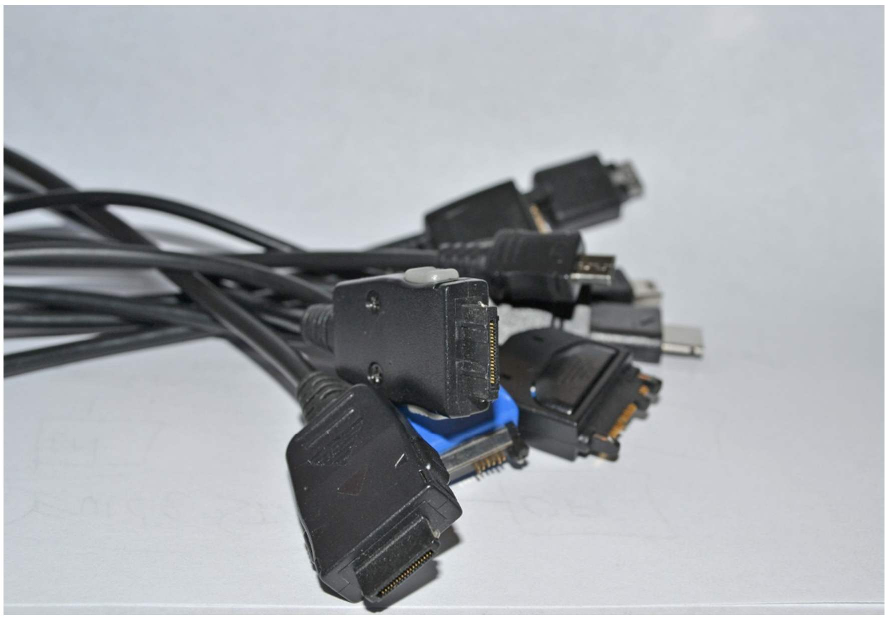


FIGURE 10-6 **The many different types of cables used for feature phones can make collecting data difficult at times.**

Device State Considerations

In order for a driver to install successfully and a solid connection to be made, the feature phone must be at least 50 percent charged or be connected using a charging cable. A charging cable has an extra pin that allows the device to receive power from the USB hub on the computer while connected. This is a good alternative if the device does not have a 50 percent charge and time is critical.

If a device does not have enough power when it is connected to the mobile forensic solution and computer, failure during processing may occur, such as the inability to recognize the device driver, a collection failure, or a successful collection but incomplete data. A computer communicates with the mobile device through the driver, and if the device does not have enough power to bridge the two devices, run system processes, and keep the phone functioning, this communication will not occur. When power is not sufficient or available, the driver either fails to load or ceases functioning at some point.

A failed collection is obvious to the examiner, because the software crashes or returns information identifying that connection cannot be made to the device, with a suggestion to reconnect and try again. Receiving this error during the collection can be frustrating, but it can be managed. Even if a device is not sufficiently charged, some software will return the collected data as though a successful collection had been achieved, but closer inspection will show that data is missing. This consequence can be devastating, because the examiner believes that no error has occurred and considers the collection complete. In fact, NIST, as part of its testing for mobile forensic software, disconnected a device that was being collected to determine whether this effect could occur to the software. The resulting physical disconnection of the device was similar to the behavior of a device that was 50 percent charged, and, as NIST noted, some software solutions will not indicate to the user a failure has occurred.

Consequences of Insufficient Device Power

During a class, while students were using Paraben Device Seizure, we witnessed a phenomenon that confirmed that battery condition and power really do matter for a successful collection. The mobile devices all contained the same information to allow instruction to flow efficiently for the class, with 25 text messages and 30 contacts. The feature phones used in class were sometimes not fully charged, and after a student reported that he acquired only 15 of the 25 text messages and 8 of the 30 contacts, I made a closer inspection of the device. All 25 text messages could be manually located on the device as well as the 30 contacts. The student reacquired the device, but this time, only 14 messages and 10 contacts were extracted. The student stated that the device battery light was flashing and needed to be charged. Over lunch, the device was charged, and upon returning, the student reacquired the device with Device Seizure and all data was successfully extracted. This was not a limitation of the software, but a limitation imposed by the battery power at the time of the extraction.

Collecting the Device Logically

If the feature phone is powered on and not locked, the examiner should first attempt a logical collection after the device has been isolated. The device can be connected to the computer using the prescribed and supported methods and placed into the proper mode if applicable. Next, the examiner can apply the solution of choice for a logical collection.

A *file system collection* should always be attempted first if such a collection is not a part of the standard capabilities supported by the software. Collecting a file system of a feature phone enables the examiner to demonstrate unequivocally in documentation and testimony where data came from during the logical collection of the phone. Too often, examiners simply allow the software to identify the data (such as contacts, call logs, and Short Messaging Service [SMS]) without completely comprehending where the actual file that contains this data "lives" within the device's embedded file system. When a file system is extracted from a feature phone, however, this is no longer a limitation, because the file is available for inspection, examination, hashing, and documentation.

If a file system option is not available or has already been completed, the examiner can obtain a *surface collection* of user data. This WYSIWYG extraction uses device protocols to query the device files, parse the extracted information, and present the data to the examiner in the solution interface. This type of logical collection can be compared to navigating the mobile device interface to locate the information and then using the software to extract that same information into the forensic solution. Typical feature phone surface collections include contacts, call logs, SMS, Multimedia Messaging Service (MMS), calendar, notes, and media.

Having this information immediately available can help steer any additional follow-up and also help to identify areas of interest to the examiner during a file system examination and after the device is collected physically. The biggest benefit to conducting a logical collection as a first step with a feature phone is ease of collection and support. Most feature phones that are not disabled by the carrier can be collected logically by most mobile forensic solutions with a USB cable, and most surface data can be parsed. However, obtaining the file system of a feature phone along with the surface data is recommended.

Associated Evidence (UICC and Memory Cards)

Feature phones also can contain a UICC and memory cards. As mentioned, each item should be treated as a separate piece of evidence. Following the recommended guidelines on processing, the examiner should examine the UICC and memory card first if the device is powered off and last if the device is powered on. Collecting each card individually outside the body of the mobile device is important, because it will assist the examiner in collecting much more information that can be used later in the investigation if needed.

UICC With feature phones, the UICC can contain much more evidence than would be found on a smart device's UICC and should not be ignored. Because of the limitations on the internal storage of a feature phone, some valuable information is stored on the UICC, including text messages and contacts. This storage setting can be controlled within the device by the phone's user.

Most mobile forensic tools will allow some information to be read from the UICC from within the mobile device during a standard logical collection. Unfortunately, this information is very limited. The examiner should always collect the UICC outside of the body of the device, to enable the UICC and its complete file system to be collected and later examined. When conducting a collection of a UICC, the examiner should make sure that the forensic tool allows for a full file system collection of the card, not just the surface data. With the file system, the examiner can verify and validate the data that has been parsed and displayed and locate additional information, as outlined in Chapter 11.

Memory Card Generally, a memory card within a feature phone contains far less data than a memory card in a smart device, but some newer Nokia feature phone devices contain a 32GB card. Typically, the smaller size limitation has to do with the phone's internal capacities to recognize larger memory cards. During a logical examination, the memory card can be read by most forensic software while it is within the phone, so the card should not be removed until this is complete. Once the memory card is removed, the examiner should also physically read the card using a forensic tool that will allow for the creation of a forensic image of a mass storage device. AccessData FTK Imager is a free alternative for creating a forensic image, along with paid commercial tools from Guidance Software (EnCase), Oxygen (Oxygen Forensic Detective), Cellebrite (Physical Analyzer), and several others.

Tip

When creating an image of a memory card using one of the mentioned tools, the examiner should be sure to have a write-blocking mechanism or appliance in place. Doing so will disable the abilities of the software to write to the evidence.

The creation of a physical image will allow for a deeper analysis of the memory card, which will include unallocated space and file slack, which generally contains deleted data. This type of data is not recoverable when the memory card is within the device and standard logical collection has taken place.

Collecting the Device Physically

If the feature phone has been powered off and is locked or disabled, or after completing a logical extraction of the mobile device, the examiner should do a physical collection of the device. A physical collection of a feature phone is critical and advisable in all situations if the device is supported for a simple reason: deleted data is almost never available in a feature phone's file system with a logical collection. If deleted data is required for a case, a physical collection of a feature phone is necessary.

Cellebrite supports the most feature phones for both GSM and CDMA physically, with MSAB coming in second because of its limited CDMA support. Unlike a smart phone, a feature phone's physical image includes only a small part of the device's internal memory. With a feature phone's memory measured in megabytes, this does not yield a lot of additional data, but, when possible, a physical image should be obtained.

When collecting a feature phone physically with a mobile forensic tool, the examiner should remove the UICC and the memory card from the device prior to starting the acquisition. Methods for obtaining a physical collection of feature phones were derived from the methods and code used by service tools, as discussed in Chapter 8. Overheating has been known to occur during acquisition, which could damage the UICC and memory card if they are inserted in the device at the time of the physical collection. Also, the UICC and memory card are not needed, nor are they a part of the area the tool will read to obtain the internal memory area.

If using other means, such as a flasher box, JTAG, or chip-off, the examiner should follow the procedures described in Chapter 8. Prior to the examiner beginning the physical acquisition, it is critical that a logical collection occur.

After a collection is completed, the examiner can analyze the information obtained in the physical collection. This is a labor-intensive process, because a lot of forensic tools currently do not support the decoding of the feature phone's file system. Tools such as MSAB XRY and Cellebrite Physical Analyzer do allow for the decoding of many feature phone file systems. These file systems can then be compared with the logical extraction during the critical data analysis phase. File system and data analysis for feature phones are covered in Chapter 12.

Archiving the Data

During each phase of the feature phone collection, all data collected from the logical collection, to the UICC and memory card, to the physical acquisition of each image, along with all associated data, should be securely stored on an evidence drive prior to the examiner beginning the analysis phase. This storage location should use a unique name that pertains to the case or investigation. All subsequent analysis will be on a copy of the data that was collected during the primary acquisition. By doing this, the examiner is never working on the original data from any phase. By archiving data, the examiner can be assured that if the software or computer malfunctions, the original evidence will still be available if a new examination of the data must occur.

BlackBerry Collections

BlackBerry devices were once prevalent in many examinations, but today, in my experience, they account for less than 1 percent of investigations. BlackBerry devices were one of the most easily collected devices by forensic software, but now they are one of the most difficult. Early BlackBerry devices' content, backup, and storage methods were documented and available to developers all the way up to version 7. Because of this level of documentation, software vendors and open source engineers built tools to mimic BlackBerry desktop software to create a local backup of the device and then parse the resultant files. This method drastically changed with the release of BlackBerry 10, an operating system built from scratch. With this new file system, along with the methods a version 10 device uses to create a backup, forensic tools have a difficult time producing usable evidence. Today's BlackBerry collections remain tedious, but since devices are using Android, the analysis and decoding of the data have become more reasonable.

> **Note**
>
> With the current market share of BlackBerry reported at the end of 2016 as 0.0 percent by Gartner, commercial tool manufacturers have stopped development cycles on these devices. Forensic examiners around the globe encounter no more than five devices a year per current surveys and polls conducted during conventions and speaking spots of active examiners. However, information on all devices is important for a well-rounded examiner.

Connecting to a BlackBerry Device

As with all mobile forensic solutions, the connection to a BlackBerry device requires a USB cable—either a mini USB or micro USB connection. Like any other mobile device, the BlackBerry must be isolated using an appropriate isolation method or appliance.

> **Note**
>
> BlackBerry was the first reported smart device that could be remotely wiped using settings within the phone. This was first reported on the forums at crackberry.com in 2008, when BlackBerry had not even added the feature to its user guide. However, this feature was used by a criminal to wipe his BlackBerry while it was being transported to be examined, and the story is often used in forensic training classes.

If a BlackBerry is powered on and locked, the examiner must know the password to access the device. Also, if the BlackBerry has encryption turned on, a secondary password might be requested, which could be different from the first. If either password is not known, the examiner will not be able to connect to the device using a USB cable; however, connection to any other associated artifact (such as the UICC and memory card) can still be attempted.

Device State Considerations

If possible, the examiner needs to know the BlackBerry access passwords, whether the device is controlled by a BES (BlackBerry Enterprise Server) or the device is unlocked. By having both passwords (handset and encryption), the examiner will be able to conduct a logical or physical collection. If the device is set up with a BES, the administrator of the BES can disable or enable the password and encryption for the device, if needed.

For BlackBerry 7 and older devices, the examiner will recognize that a device is locked by the padlock icon that is displayed in the lower-right corner of the screen. If the device is also encrypted, another padlock will be visible at the upper left. The device will have to be unlocked using the previous suggestions or by enabling the info.mkf file on the memory card. This file is available only on the memory card at /BlackBerry/system/ if the device is locked and the card has been encrypted using either the BlackBerry Security Password or Device Password mode. If this is the case, the examiner can use the Elcomsoft Phone Breaker tool to obtain the password. This method will not uncover all passwords, however, especially if they are complex, unless a dictionary has been compiled. The password, if obtained, can then be used to create a backup of the device.

On BlackBerry 10 devices, security is similar to that of prior versions with device passwords, device encryption, and media card security and encryption. One difference, however, is that locked version 10 devices cannot be accessed using the info.mkf memory card file. If a device is locked, the only way to bypass security and obtain a forensic image is by obtaining the password, or the BES 10 admin can reset the password if possible.

Collecting the Device Logically

If the device is powered on and the examiner has access to the device with disabled security, he or she can perform a logical extraction using most mobile forensic software. This involves attaching a USB cable, plugging the cable into the computer, and selecting the device to be collected. The software will perform a backup of the device, similar to the backup functions in the BlackBerry Desktop software, and then parse out the database files for supported user data files. The BlackBerry collection does not contain a typical file system as you would expect, but a listing of all of the database files from the device. In a logical collection, it is important that the software enables the examiner to access each of the database files, regardless of the software's ability to parse out the user information. In other words, the examiner should be able to examine the complete set of files collected from the BlackBerry device to verify and validate the solution's parsing ability, as well as uncover additional data the solution might not be currently able to extract.

Note

Current mobile forensic tools do not support the collection of a BlackBerry 10 device directly, but some can produce a backup and then conduct an analysis. This method will be explained later in the section "BlackBerry Collections."

Much like a feature phone, if the BlackBerry also contains a UICC and memory card, the artifacts should be removed and processed. However, because of the limitations also imposed on a physical collection by a locked device, a physical collection should be attempted before an acquisition of the UICC and memory card if the device is unlocked. If the device is locked or powered off and known to be locked, the UICC and memory card can be processed first as described in the preceding section, making sure a physical collection of the memory card takes place.

Collecting the Device Physically

The physical collection of a BlackBerry device using a USB connection is limited to a small number of devices and even a smaller set of forensic tools. Cellebrite UFED Touch currently supports physical collection of 50 various CDMA and GSM devices, as indicated in its current supported devices list (www.cellebrite.com/mobile-forensics/support/ufed-supported-devices), but the device password must also be known. Currently, BlackBerry 10 devices are not supported for a physical collection by any tools via a USB connection.

JTAG can be used on locked devices if the password is not known, but this technique is relegated to older devices because BlackBerry has made the TAPs extremely difficult, if not impossible, to use. The only alternative method would be an invasive physical examination. The chip-off method can be used on devices up to but not including BlackBerry 10, and the created binary file can be examined and decoded using Cellebrite Physical Analyzer or Oxygen Forensic Analyst. However, even when the chip-off method is used, examining BlackBerry 10 devices is impossible because the data is encrypted at the chip level. So even with a successful binary dump, the data could be unreadable and unusable.

Additional Collection Methods

Both free and commercial mobile forensic solutions use the same protocols used by the manufacturer's software, and at times using the software developed by the manufacturer can be beneficial. However, the BlackBerry software up to version 7.x, BlackBerry Desktop Manager, also allows data to be restored and synced to a connected device, so the examiner must be careful when using a software tool that can synchronize data to a device being examined. BlackBerry 10 devices use BlackBerry Link software, which also allows the device to be backed up and restored. Currently available tools cannot complete a backup of a BlackBerry 10 device independent of using the BlackBerry Link software. Furthermore, the output file that is created by the software is encrypted and unable to be parsed by most currently available software, with two exceptions: Oxygen Forensic Detective and Elcomsoft both support the ability to open BlackBerry backups from BlackBerry 10 devices.

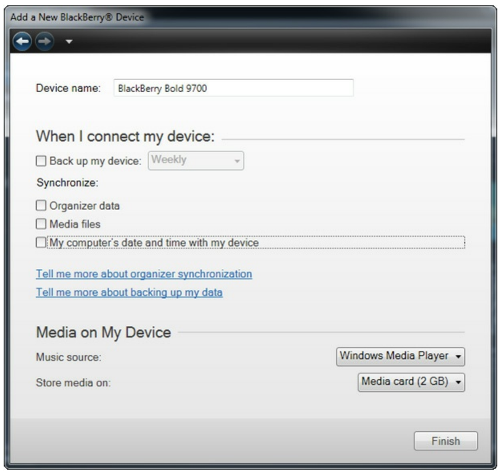
With new BlackBerry devices and the underlying Android operating system, the examiner can conduct a backup of the data using the USB debugging bridge (also known as ADB) as well as the associated Google account. These devices have a function that allows for the backup of the data for later restore if the device is lost, stolen, or damaged (https://help.blackberry.com). The information that is backed up to the Google account includes the following:

- Google Calendar settings
- Wi-Fi networks and passwords
- Home screen wallpapers
- Gmail settings
- Apps installed on Google Play
- Display settings
- Language and input settings
- Date and time settings
- Third-party app settings and data (varies by app)

To back up the information manually on a BlackBerry, tap the cog icon and then tap the Backup & Reset button. A Google account must be configured.

BlackBerry Desktop Backup Software To create a backup of a BlackBerry device successfully using

BlackBerry Desktop Manager software, the examiner must first install and properly configure the software, which can be downloaded from http://us.blackberry.com/software/desktop.html. Both Mac and Windows are supported. The examiner should make sure to disable automatic syncing upon connection and ensure that automount is disabled on the forensic computer so the memory card is not altered by the operating system. When the examiner sets up the device, My Computer's Date And Time With My Device should be unchecked, as shown in Figure 10-7. The BlackBerry can then be backed up.


Add a New BlackBerry® Device

Device name: BlackBerry Bold 9700

When I connect my device:

☐ Back up my device: Weekly

Synchronize:

☐ Organizer data

☐ Media files

☐ My computer's date and time with my device

Tell me more about organizer synchronization

Tell me more about backing up my data

Media on My Device

Music source: Windows Media Player

Store media on: Media card (2 GB)

Finish

FIGURE 10-7 **Disable the synchronization feature before connecting the device.**

BlackBerry IPD and BBB Backup Files An IPD (Inter@ctive Pager database) backup file is produced with the BlackBerry Desktop software running on Windows machines up to version 7, and a BlackBerry Backup (BBB) file is produced on the Mac versions of BlackBerry Desktop up to version 7 of the BlackBerry software. The IPD is a single file that contains multiple other database files that make up the content on the BlackBerry device. The Mac BBB file (up to version 7) is a compressed IPD file. At version 7.1, BlackBerry transitioned to creating BBB files on both Mac and Windows machines using BlackBerry Desktop, and the BBB is still a compressed file, but it now comprises .dat files. Each .dat file contains various data and data types. BlackBerry Desktop is no longer updated by BlackBerry because the company is transitioning users to the BlackBerry Link solution.

BlackBerry Link also creates a BBB file that differs from the Desktop BBB file. The difference is encryption: The newer BBB file is fully encrypted independent of user interaction on the device or within the Link software. This means that if a backup is going to be read within a forensic software solution, the examiner must have a key piece of information: the BlackBerry Link password associated with the BlackBerry user account. Technically, the examiner will need both the user account associated with the device and the password, but current software, such as Oxygen Forensic Detective, which supports the analysis of BlackBerry 10 devices, reads the user ID from the manifest and requests only the associated password. What can cause difficulty is the fact that the Link password is completely independent of the device password, so even if the device password is known for the BlackBerry 10 device and it is unlocked, the collected backup might still not be able to be examined if the Link password is unknown. The BlackBerry Link password must be used to decrypt the files to enable the software solution to parse the available user data. Furthermore, the verification of the BlackBerry ID by the BlackBerry server is used for BlackBerry Link and is necessary for the decryption of the backup-and-restore process. The software solution must be online for this process to be successful.

The BBB file that is produced is similar to the original BBB compressed file, but new BBB files contain a manifest file and informational file that are not encrypted and can offer information about the device. This file is needed by the current software solutions mentioned in previous sections along with the BlackBerry Link password. Mobile forensic software from Oxygen Forensics allows the new BBB files to be decrypted and analyzed with the correct BlackBerry account password for both live and backup collections.

With the introduction of the Android runtime to the BlackBerry device, the device is much more accessible to recover without the use of the BlackBerry backup software.

Archiving the Data

Much like a feature phone collection, the BlackBerry data and UICC and memory card information should be securely stored on an evidence drive prior to beginning the analysis phase. This storage location should have a unique name that pertains to the case or investigation. All subsequent analysis will occur on a copy of the data that was collected during the primary acquisition.

Note

Even if the BlackBerry 10 device is currently encrypted, the examiner should create a backup if possible. This will enable the examiner to perform an analysis at a later date if a password is eventually discovered or additional techniques are introduced. Software solutions are always advancing with more options available. Having the file at the onset, even if not currently usable, would enable it to be parsed with new technology.

Windows Mobile and Windows Phone Examinations

Although the Windows Mobile operating system is no longer on the market and Microsoft has transitioned to Windows Phone, Windows Mobile systems are still being used. Windows mobile devices, both Windows Mobile and Windows Phone, differ regarding collection, the ability to access user data, and data layout. Today, the Windows Phone holds a small portion of the smart device market share, but it continues to grow. Undoubtedly, an examiner will be requested to complete an examination of a Windows Mobile or, more likely, a Windows Phone at some point.

Windows smart phone devices were released in 2000 as a mobile solution based on Windows CE (Compact Embedded) and saw exposure as Pocket PCs and the Windows smart phone based on CE 3.0. Windows Mobile 2003 was running 3.0, and in 2004, Windows Mobile 5 was released and 6 followed, all progressing on the CE architecture. The last available Windows Mobile version, 6.5, was released in 2009, and Windows Phone 7 became available in 2010, which was not just a complete rebranding, but something completely new.

At the release of the Windows Phone 7, users and examiners alike saw a complete change in the collection and recovery of a Windows mobile device. Windows Phone 7 was not based on previous Windows Mobile functions and accessibility, and as such it created problems for digital forensic examinations. In 2011, Windows Phone 7.5 was released, and version 7.8 added some Windows Phone 8 capability, since 7.8 devices could not be upgraded to Windows Phone 8 because of hardware limitations. In 2012, Windows Phone 8 was released and in 2014 upgraded to version 8.1. The device was again rebranded as Windows 10 Mobile in 2015 as a way to unify with the PC operating system. What has been of benefit to mobile forensics examiners is simply the market share of 0.01 percent by Windows devices, which means Windows devices aren't involved in most mobile forensic examinations. However, with the massive move to cloud services by most mobile device apps and devices themselves, the data can often be recovered as long as the data was backed up to the service.

Remember that a Windows device can also contain a UICC and memory card, so the correct procedure in processing these pieces of evidence should be followed, depending upon the state of the mobile device.

Connecting to the Device

Communication with a Windows Mobile device and a Windows Phone both generally require a USB cable and computer, but the way and conduit in which the communication occurs differ between the two devices. Communication to the Windows Mobile device occurs using Active Sync or Windows Mobile Device Center (WMDC) protocols. Both operate using a hardware abstraction layer, much like how iTunes operates with an iOS device. Active Sync and WMDC are not capable of the same communication layer with a Windows Phone device, so Microsoft Zune was released to accommodate that. However, the access to a Windows Mobile device using Active Sync and WMDC is considerably more verbose than a connection to a Windows Phone using Zune. In 2015 Microsoft released the Windows Phone Companion to act as a conduit to the flagship Windows 10 operating system that supported Windows 10 Mobile devices, Android, and even iOS. It has been rumored the software has been discontinued and a form of the Companion will be incorporated into the Windows 10 OS.

When connecting either a Windows Mobile or Windows Phone device, the software must have preinstalled drivers, or the examiner must install drivers for the device. Most forensic software that supports Windows Mobile devices will either have Active Sync or WMDC installed as part of the driver package. Connecting and then collecting with a Windows Mobile device, as mentioned, is similar to doing so on an iOS device, in theory. The examiner will use the data syncing function built into the Windows Mobile device architecture and operationalized via Active Sync or WMDC, install a file to the device or device SD card, and initiate the card on the device. Connection and logical collection can occur only using an intermediary file, also referred to as an *agent*. The agent, using the Active Sync or WMDC connection, transfers data to the forensic software. The examiner must ensure that the loaded file does not overwrite inactive data.

Windows Phone logical collections using commercial and open source tools involve using the Zune software, ActiveSync, WMDC, Windows Phone Companion, or a derivative.

Note

Some commercial tools have harvested some of the functionality from the Zune software to bundle into their driver packages, but these tools will not allow Zune to be installed along with the forensic software. If a Windows Phone cannot be recognized, make sure the forensic software is not conflicting with Zune, if it's installed.

Collecting the Device Logically

If the device is powered on and unlocked, and after connection has been made to the Windows Mobile device using Active Sync or WMDC, the agent recovery process will begin. Typical data recovered using mobile forensic tools includes contacts, call logs, SMS, media, and other critical user information. On the other hand, a Windows Phone will yield very little user type information with a logical collection. The information that is collected in a logical extraction of a Windows Phone is limited to media and documents. Really, anything that is accessible using a Windows Explorer instance will be collected in a logical collection using modern forensic solutions.

If the device is locked with a password, the only option will be to conduct a physical collection of the Windows Mobile or Windows Phone device.

With today's Windows Phone 10 devices, a considerable amount of data can be collected from the user's cloud account to include SMS, app data, phone settings, and media (images/videos). To set a device to back up to the account, the proper method, as outlined by Microsoft, is to tap the Windows key | Update & Security | Backup and then select both settings and app data. Furthermore, if a manual backup is needed in the case of a forensic examination, tap the Windows key | Update & Security | Backup | More Options | Backup Up Now. To back up the messages on the device, tap Settings | System | Messages | Turn On Sync Messages Between Devices. If media items (images/videos) are to be backed up, this will occur on One Drive, which can be extracted via the cloud backup. To set this option on the device, tap OneDrive | Menu | Settings | Camera Upload. Using this method will enable the collection of much more information than can be obtained using the standard USB connection.

Collecting the Device Physically

Most Windows Mobile 8 device models can be collected physically using non-invasive methods. Cellebrite, MSAB, and Paraben software include methods to recover a file system from a Windows Mobile device. When the examiner is conducting a physical collection of a Windows Mobile device, Active Sync or WMDC will be used to place an agent onto the device. The agent is then initiated and the recovered data is sent to the forensic software.

Most of the forensic tools are based upon the work by Willem Jan Hengeveld and the open source Remote API (RAPI) tools (www.xs4all.nl/~itsme/projects/xda/tools.html). RAPI is defined by the Microsoft Developers Network as follows:

> The Remote API (RAPI) library enables applications that run on a desktop to perform actions on a remote Windows Mobile device. The functionality that RAPI provides includes the ability to manipulate the file system on the remote device, including the creation and deletion of files and directories. RAPI functions can be used to create and modify databases, either in the device's object store or in mounted database volumes. RAPI applications can also query and modify registry keys as well as launch applications and invoke methods on the remote device. Although most RAPI functions are duplicates of functions in the Windows Embedded CE API, a few new functions extend the API. Use these functions to initialize the RAPI subsystem and enhance performance of the communication link by compressing iterative operations into one RAPI call.

Caution

Using RAPI tools and other advanced tools and command utilities without properly understanding and testing them is not advised.

The RAPI tools shown next enable the examiner not only to extract the database and system files but also dump RAM and entire partitions from Windows Mobile devices with more than 30 different functions. Only two of these functions are discussed in this section:

- **itsutil.dll** This file is copied to the connected device using Active Sync and is used as a helper library for access to the Windows Mobile device. This file can be copied to various locations on the mobile device and is set by using a Registry key on the examination computer.

  ```
  HKEY_CURRENT_USER\software\itsutilsdevicedllpath [ "\Storage Card\itsutils.dll"
  ```

  This places the file onto the storage card of the connected mobile device. A log file is also created if set to true, and its location can also be set in the Registry of the connected computer. When any item is executed using the itsutil.dll, the examiner will be prompted to accept this function on the mobile device interface.
- **`pdocread`** This command can be used to make a copy of the partitions on the mobile device. Using this command with a `-l` flag will list the known devices and the active handles on the device. This information will be used by the examiner to identify the correct user partition to be selected and then extracted. Using a `-w` flag when reading the selected partition will specify use of the standard disk API to perform the read. The format and parameters of the command should look like this:

  ```
  pdocread -[flag(s)] -[partition] [starting byte] [ending byte] <path and output
  file name>
  ```

  After obtaining a list of the partitions using `pdocread -l`, the examiner will see partitions identified as disks with partitions, but more importantly for collection, with string handles. Using the partition name (in hexadecimal) or handle number and total size of the partition (in hexadecimal), the examiner can then extract the partition from the device. If the output produced by `pdocread` lists the partition of interest as `handle#3 73efe04a 60.98M (0x3cfc000)`, the command to create a binary image of this partition looks like this:

  ```
  pdocread -w -h0x73efe04a 0 0x3cfc000 D:WindowsMobile.bin
  ```

  A binary file will then be output to the D: drive and can be analyzed by EnCase, FTK, Oxygen Forensic Analyst and Detective, Physical Analyzer, or XACT.

Note

On devices containing larger partitions, the standard disk output controlled by Windows can fail to extract the complete disk size specified in the command.

Windows Mobile applications cannot run on a Windows Phone, so a different approach to obtaining a physical image of the device is needed. Several methods for obtaining partition-level information from Windows Phone 7 have been circulated, which are much like the earlier versions of rooting techniques for

Android. Still termed "rooting," the level of sophistication and technical work involved with both Windows Phone 7 and 8 devices to obtain root-level access is daunting for most examiners. For version 7 and 8 devices, the examiner must have a device with an unlocked bootloader, and if the device is not unlocked, he or she must perform necessary modifications to the device to unlock the bootloader. Once unlocked, another application will be run to obtain permission and root the device. ChevonWP7 was frequently used with earlier Windows Phone 7 devices but has since ceased to be developed or used because of the difficulty involved in running it successfully.

With Windows Phone 7 and 8 devices, the most successful methods currently used to obtain a physical-level collection involve invasive techniques. Both version 7 and 8 devices are supported by JTAG techniques and chip-off. The collected image can then be examined in EnCase, FTK Imager, FTK, Oxygen Forensic Detective, Physical Analyzer, or XACT.

Alternative Collection Methods

Windows Mobile devices are supported pretty extensively by all commercial tools. With coverage for both logical- and physical-level collection, a commercial mobile forensic tool will often be the most reliable. However, alternative applications are available for Windows Mobile devices.

PIM Backup (www.dotfred.net/) has been used in MFI mobile forensic training courses and offers a free alternative to the logical recovery of valuable data from a Windows Mobile device. PIM Backup functions much like the commercial forensic tools and uses Active Sync or WMDC to install an application to the device over a connected USB cable. After the application is installed to the mobile device, it runs and the selected data is exported into individual files (such as CSV, XML, or iCAL files).

Windows Phone 7 and 8 do allow for the creation of a backup using Zune, but the backups are encrypted, and currently no tool can decrypt the backups. Some third-party tools also allow backups, but they also use the Windows Phone backup service that creates the encrypted backup. Windows Phone 7 and 8 devices enable the examiner to access files and folders that are stored on the internal memory and also the external card. Both versions use external and internal storage areas much like the standard Windows operating system and use these areas to store data that is directly related to the device operating system. If a memory card is removed, a Windows Phone could become unstable, and sometimes the device will have to be reset. This creates a great opportunity for the examiner to recover valuable information simply by moving the data from the memory card into an evidence location. This data could include media types, documents, and other files.

Windows 10 Mobile devices are also supported by the majority of mainstream tools, but the type of data that can be extracted is typically limited to the data that can be observed and extracted when the device is plugged in and mounted as a portable device in the collection operating system (for example, documents, media, or files on media card). However, with many tools now supporting the extraction of data from cloud services, critical data such as SMS messages, application data, and settings are now accessible.

Archiving the Data

All data, including physical images, exported files, backups, and logical images, along with physical images of memory cards and UICC information, should be copied to an evidence drive before the analysis phase. The information needed for the analysis phase should then be copied to a temporary location. The storage location should include a unique folder specific to the case and an internal folder that is specific to the device. Any additional evidence recovered as part of the analysis should also be stored permanently at this location once the case is completed.

Note

As with BlackBerry devices, the current market share of Windows Mobile devices was reported in the first quarter of 2017 as 0.15 percent by Gartner. Commercial tool manufacturers have stopped development cycles on these devices, and forensic examiners will see even fewer Windows Phone devices than BlackBerry devices. However, knowledge of all devices is important for a well-rounded examiner.

Apple iOS Connections and Collections

A collection of an Apple device by a forensics examiner can be an almost daily occurrence, because Apple mobile devices are some of the most widely used devices globally. Connections to and collections of these devices are rather straightforward processes. Apple allows the device to communicate with a computer using its proprietary protocols via iTunes. iTunes enables communication to occur with the attached device, and a user can update applications, media, device firmware, and more. Forensic software vendors use the same type of communication methods used by Apple with iTunes. Forensic software vendors, third-party free utilities, and open source tools use methods and services that have been exposed in Apple's API, enabling the software to simulate an iTunes communication session, but these tools also use methods that are not used in the iTunes application.

Communication with the Apple device is generally straightforward for each forensic solution. What will be the most confusing part for an examiner is understanding the logical versus physical support of an Apple device.

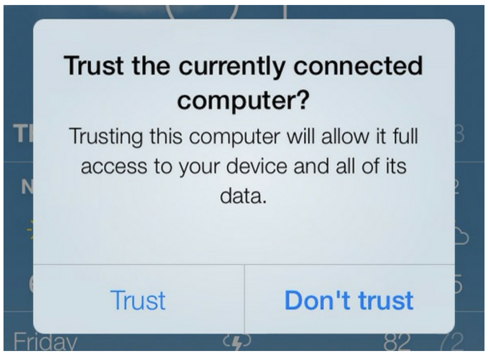
Connecting to the Device

Connecting to an Apple device with most traditional mobile forensic solutions requires a USB cable. Apple mobile devices up to and including the iPhone 4S and iPad 1 will use a standard iPhone 30-pin cable. When the iPad 2 was released, Apple changed the USB cable to an 8-pin Lightning Cable, and now all Apple devices use this cable for charging and syncing.

Note

Many manufacturers have released less expensive cable products that violated Apple's patent and produced a non-OEM (non-Lightning Cable) cable. Most of these cables are recognized by the Apple device when plugged in, but will not allow data transfer, or will cause failures during data transfer. For these reasons, using the Apple Lightning Cable for connection to an Apple device is recommended. More and more Lighting Cables are becoming available on the market that do not fail, however, and allow for a quality extraction—at a lower price.

The cable connection is located at the bottom of the device. After the cable is plugged into the computer, with iOS 7 and later devices, the examiner will see a prompt asking whether the computer should be trusted (see Figure 10-8). This security feature enables the device to create a pairing record with the attached computer if a pairing record has not already been established. (If the device has previously been connected to the computer and is unlocked, this prompt will not be shown. The pairing record that is created upon acceptance is discussed a bit later in this section.) Once the device is attached to the computer and the drivers have been successfully installed, the device can be collected using a software solution of choice as long as the iOS device and its OS version are supported by the solution.


Trust the currently connected computer?

Trusting this computer will allow it full access to your device and all of its data.

FIGURE 10-8 **The Trust Computer prompt must be accepted for successful data transfer.**

If at any time the examiner chooses not to trust the computer, the device will have to be unplugged from the computer and then reattached for the trust prompt to be displayed again. If the examiner does not indicate that the computer can be trusted, the forensic software will be unable to collect the iOS device. If the prompt to trust the computer does not display, either the device is running an iOS version earlier than 7 or the device has already been deemed trustworthy. This is often the most encountered failure in the connection phase with an iOS device.

Device State Considerations

If the device is unlocked and powered on or off, collection will not be a problem, no matter what the operating system version. If the device is locked by a password and the password is known, the examiner should attach the device to the computer that will complete the collection, unlock the device, and then accept to trust the computer. Unlocking the device will create a trusted pairing relationship that will be used later during the collection if the device locks. If at any time the device has to be reexamined, a password will not be needed to complete a collection.

If the device is locked and running any version of iOS, the examiner can use the *escrow keybag*, also known as the pairing record. The escrow keybag is created and used for user experience during an iOS device backup-and-restore process. Where the escrow keybag for the iOS device is located depends on the type of computer operating system being used for the examination:

- **Windows** %ProgramData%\Apple\Lockdown\
- **macOS X** /private/var/db/lockdown/

Within the lockdown folder is a property list (plist) that is identified using the device UDID (Unique Device Identifier). Using these property lists will be covered in the discussion regarding locked devices in the following section.

When a user plugs in an iOS device with a set passcode, he or she is asked to enter the passcode. The device then creates an escrow keybag (pairing record) that contains the same keys that are used on the device, along with a new generated key. The escrow key and the new key are split between the device and the computer to which it is connected. With any reconnects to that same computer, a password does not have to be entered into the device for processing. By using the escrow keybag, the software can bypass the lock and process the mobile device. With iOS 7 and later versions, there is a caveat, however: if the device is rebooted and then reconnected to the computer that contains the device escrow keybag, the device could request the passcode to be re-entered, creating a new trusted relationship. Obtaining the lockdown folder from the computer with which the iOS device was last synced can help to process a mobile device that is locked. Using this folder and included property list files, the examiner can use mobile forensic software to use the file to simulate a pairing relationship with the iOS device and conduct a logical collection. Without this pairing record for today's iOS devices, the examiner will be unable to connect and ultimately process any mobile device data.

Collecting the Device Logically

All mobile forensic tools use Apple File Conduit (AFC) and forms of AFC along with Apple Services to conduct a logical extraction of an iOS device, just like Apple's own iTunes. iTunes must be installed on the forensic computer for a logical collection to occur for most software solutions; however, some software solutions that collect iOS devices do not require that iTunes be installed. All iOS logical collections are not created equal, and it is imperative that the examiner understand and research the data that can be extracted with the software solution that will be used. A software solution for an iOS device should be able to collect more data than would be available in a standard iTunes collection.

Note

Consulting the forensic software solution's documentation will assist in determining whether iTunes must be installed on the forensic computer prior to conducting a collection.

The connection and logical collection of an iOS device is the same for all devices, with some special considerations. Typically, the examiner can connect the USB cable, connect to the computer or hardware device with the collection software, and extract the data. The special considerations come in the form of an iTunes password and encrypted backups. If the device to be examined has the backup encryption set, the backup that is to be collected will not be readable. During the collection phase, the examiner must supply the password for the iTunes backup in order to decrypt the iOS data after it is extracted from the device. Most forensic software solutions offer the ability to decrypt the device backup from an iOS device. If the iTunes password is not known, some data will still be available for collection if the device is not locked, such as media and some application data.

Note

A logical collection of an iOS device can include a significant amount of valuable data. Far too many examiners believe that only a small amount of data is available in a logical iOS collection and neglect to perform a collection. In Chapter 13, you'll see why this is not the case.

An iOS device does not have an external media card, but it can contain a UICC, which should be removed prior to the collection. Personal experience has shown little valuable user information on a UICC card from an iOS device, but network information can be observed. However, the UICC should be collected and included in the overall case file as evidence.

Collecting the Device Physically

Probably the most frequently asked question of software vendors, support staff, and training staff has to do with the non-invasive physical collection of an iOS device. The question is always centered around the collection of an iOS device version later than an iPhone 4S from those versed in the limitation, but from new examiners it is more along the lines of questioning why a piece of software that specifically states it supports the physical collection of iOS devices fails on anything above an iPhone 4S or iPad 1. This is not a limitation of the software or the technology, but a limitation imposed by Apple in an effort to patch vulnerabilities. Devices with an A5 or newer processor (as of today, an A12) will not support the vulnerabilities exposed by today's forensic software. The iPad 2 was the first device to use the A5, with the iPhone 4S soon following and now the iPhone XS sporting the A12. The A4 and earlier processors can be exploited using the same method by all forensic software vendors that currently support a physical collection of an iOS device.

An iOS device can run in normal mode, DFU (Device Firmware Update), or Recovery mode. The non-invasive physical methods used by forensics tools use DFU or Recovery mode to obtain access to the device via USB. If the device is not running in DFU or Recovery mode, the device typically boots up in normal mode starting with the read-only bootROM as the first stage when powering on. This startup procedure in normal mode is called the "chain of trust." In the chain of trust, the device boots and walks through a series of security checks using signatures for each level. Each level will then check the other level (for example, the LLB [Lower Level Bootloader] checks the iBoot, iBoot checks the kernel), and if at any time a signature does not match, the iOS device will stop the boot process.

When an examiner places a device into DFU or Recovery mode, the iOS device boot procedure changes to involve second-level bootloaders, iBSS and iBEC. These are stripped-down versions of iBoot that allow for the preparation of the device Restore Ramdisk, but in the case of a forensic collection, the iBSS bootstraps the iBEC to deliver a custom RAM disk into the volatile memory of the iOS device. The custom RAM disk allows access to the iOS device and the partitions (OS and UserData) that otherwise could not be accessed when the iOS device is in normal mode. The forensic software will take advantage of the newly granted access to the device's file system and a non-invasive physical image of the device can be collected. Once the device is rebooted after the collection, the custom RAM disk is removed, and the only trace that something has occurred to the device would be the indication that it had been rebooted.

The HFS+ (Hierarchical File System Plus) file system contained within the physical image of the iOS device can then be examined within the forensic software. As outlined in Table 10-2, the custom RAM disk method is possibly with only a certain set of iOS devices. With the supported devices, Apple added a layer of data encryption for unallocated areas of the user partition. With this layer of encryption, the data can be collected, but the keybag used for the decryption of that area is unavailable. This occurred in device versions later than and including iOS 4.

TABLE 10-2 **Devices Capable of a Physical Collection Using the Customized RAM Disk Option**

iOS Device	Versions Supported
iPod Touch	Generations 1–4
iPhone	iPhone 3G, 3GS, 4
iPad	Generation 1

All other iOS devices not supported using the bootROM or iBoot exploit and RAM disk function (such as iPhone 6+, 6, 5S, 7, 8, X, and newer versions of the X) can also have their internal file systems collected with exposure to protected files not available in a logical file system (HFS+ and iOS 10.3 APFS) collection. In this case, the device must be jailbroken using a userland exploit, which is completely software based, and access is available only to the user area without access to the boot process. This means that the entire partition, as in a bootROM or iBoot exploit, is not available for full extraction; only the internal file system can be extracted. When an iOS device is jailbroken, most forensic software will obtain the new jailbroken verbose file system using the standard logical collection methods, along with advanced protocols added in the jailbreak. This collection, because of the now-jailbroken device, will allow access to files and folders otherwise not available in a standard logical collection. These folders and files include Apple Mail, Safari, applications, the protected data store, the cache, and many other files not accessible by other means.

The most common solutions used for iOS devices used to be Pangu (http://en.pangu.io/) and TaiG (www.taig.com/en/). TaiG had offered an untethered solution for up to and including iOS 8.2, while Pangu supports up to and including iOS 9.3. However, as the file system versions advanced, more and more jailbreaks became available, with published exploits Meridian, Electra, and LiberiOS shown to be the most utilized and popular. If using a tethered jailbreak, the examiner must plug the iOS device into a computer every time it is booted so the iOS device can boot with the help of the jailbreak application. Most jailbreaking tools use a tethered solution, but in today's jailbreaking world, semi-tethered and semi-untethered jailbreaks are available. The semi-tethered jailbreak operation is much like the tethered version described previously, but more stable, and it enables the device to again be jailbroken upon reboot with the jailbreaking tool. With the semi-untethered jailbreak, the device is not in the jailbroken stage upon reboot but can be re-jailbroken by the user of a side-loaded app or even a web site.

Note

Jailbreaking a mobile device was ruled *not illegal* by the Library of Congress in 2012 (www.copyright.gov/fedreg/2012/2012-26308_PI.pdf) and extended to tablets (such as an iPad) a few months later. However, jailbreaking can void the warranty by Apple, so using third-party methods to jailbreak a device should be used only in situations where warranted.

An examiner who understands the methods used to obtain physical access to an iOS device, either using bootROM exploits, iBoot exploits, or userland, can positively impact later testimony and documentation, especially if the method is challenged.

Unlocking an iOS device and obtaining a raw file system using a tool such as GrayKey from Grayshift can produce the same type of file system as a jailbroken phone, but can also acquire a snapshot of the device memory as well. This information and artifacts are discussed in Chapter 13.

iTunes Collection

If performed correctly, an iTunes collection will yield much of the same information you would expect from a collection with forensic software. However, an iTunes collection backup will contain many files that are represented by a hash of the path of the actual file on the device. Although the collection process and the connection of the device to the computer are no different from what has already been discussed with other operating systems, the collected data's format is much different from the others. The examination and analysis of this data are covered in Chapter 13.

To create a backup successfully using iTunes, the examiner should first download and install the most current version from Apple (www.apple.com/itunes/download/). After installing and launching the software, and after iTunes is running, the examiner should navigate to Edit | Preferences and click the Devices icon on the tool ribbon. On the Devices screen, the examiner should select the checkbox for Prevent iPods, iPhones, And iPads From Syncing Automatically. The iOS device can then be connected.

iTunes displays the connected device. On the device information screen, the examiner must clear the checkbox for Encrypt Backup. If this box is selected, the device has been set to encrypt the backup, and the examiner, when unchecking the box, will have to enter the password for the iTunes backup that the device owner used. If the password is not known, a backup can still be created, but the data that is produced will be encrypted. An examiner should still create a backup, even if the backup password is not known, in case the password is later available. Or the examiner could use software such as Elcomsoft Phone Password Breaker (EPPB) or Oxygen Forensic Detective to retrieve the password. While still connected, the examiner should right-click the connected device and select Backup. A backup will be created in the predefined default location depending on the operating system:

- **Mac** ~/Library/Application Support/MobileSync/Backup/
- **Windows Vista, Windows 7, Windows 8, and Windows 10** \Users\(username)\AppData\Roaming\Apple Computer\MobileSync\Backup\

Note

The backup password and the PIN/Password for entry to the phone are two separate security features. This means they can be, and usually are, completely different passwords.

Note that many users are using the iCloud backup function for iOS devices versus the standard cable connection to a PC or Mac using iTunes. However, the data backed up using a cable and iTunes contains additional data not available in an iCloud backup. This information was covered in Chapter 4 and more information will be available in Chapter 13.

First iOS Examination

In 2007, when no forensic tools were making collections of iOS devices, I was asked to complete a forensic examination of a new type of phone—the iPhone. After doing some research, I learned that the device created a backup using software called iTunes. After I installed iTunes and connected the device, it immediately started to sync the information from the device into iTunes (and I learned a valuable lesson: I must disable this option first!). When the collection was completed, I researched the backup function and initiated a backup of the device onto the computer. When the backup was completed, I was amazed at the amount of information that I could find within the backup file using AccessData FTK software. I uncovered numerous text messages, photographs, and call logs that were needed for the case. As many more iPhones came into the lab and there still was no solution out there for forensic examiners, I wrote a white paper covering the way I completed my first iPhone collection. This document was later used by both open source developers and commercial developers to create a solution to harness the way iTunes completed a backup and then parse the user data. The initial document I published in 2007, "iPhone Data Extraction," is still available on the Internet at https://upslide.tips/iphone-data-extraction-by-lee-reiber-this-document-will-give.html.

Archiving the Data

All data, including physical images, logical file systems, backups, pairing records, and UICC information, should be copied to an evidence drive before the analysis phase begins. The appropriate information should then be copied to a temporary location for the analysis phase of the investigation. The storage location should be a unique folder specific to the case and an internal folder specific to the device. Any additional evidence recovered as part of the analysis should also be stored permanently in this location after the case is completed.

Android OS Connections and Collections

With more than 18,000 different types of Android devices available to users around the world, it is no wonder that connecting and collecting from these devices can be difficult at times. An Android logical collection is much different from an iOS logical collection because the internal logical file system of an Android is not available by default. Settings on the device must be enabled for proper communication, and these can vary depending upon the device. Not only does it matter what type of Android device is being examined, but the version of the operating system will also determine whether a connection and collection can occur. Other considerations for Android collections include whether the device is powered on or off, locked or unlocked, encrypted or not, and also whether or not security is enabled. A locked device with security enabled can prevent the examiner from turning on settings needed to communicate with an Android device. All of these facts must be considered before the examiner even connects the device to be collected.

The Android device can contain a UICC along with memory cards, and the processing order described in the "UICC" and "Memory Card" sections earlier in the chapter should be followed. These sections cover the connection of the mobile device.

An Android device should be collected with the memory card still inserted in the device to collect and extract the media and file data that are located on the removable media. If the memory card is removed, actual files that are on the external media will not be mapped to data that is contained on the mobile device's internal database.

Connecting to the Device

The Android Debug Bridge (ADB) is used to communicate with an Android device. The ADB.exe command line tool is part of the Android Standard Development Kit and is used for testing and development of Android apps and applications. This tool offers a computer a way to communicate with a connected Android device. It comprises the client, a server, and a daemon. The *client* is the action or job initiated by the ADB command. The *server* is the management portion that is running on the computer that manages the communication between the client and the ADB daemon that is running on the Android device. The ADB *daemon* is a background process that is running on the device conducting the command and control as given by the client. ADB must be used for all communication to and from an Android device, whether it is a logical or particular non-invasive physical collection. The Android device that is to be collected must have ADB enabled, either manually by the examiner or programmatically by the software. Once ADB is enabled, the proper ADB driver must also be installed and functioning on the examination computer for proper communication.

Tip

The only driver that would need to be installed successfully to collect an Android device is the ADB driver; all others are not significant.

Whether the examiner will need to set the Android manually into ADB will depend upon the operating system. Outlined in Table 10-3 are the various ways of setting an Android device into ADB, depending upon the operating system version. All ADB menus are located in the Developer options menu.

TABLE 10-3 **Ways to Locate the ADB Functions Within Android Devices (http://developer.android.com/tools/help/adb.html)**

Android OS Version	To Place a Device in ADB...
2.x and 3.x	Choose Settings, Applications, Development, and then USB Debugging
4.x	Choose Settings, Developer Options, and then USB Debugging
4.2.2 and later	Choose Settings, About Device, and then tap Build Number seven times Choose Settings, Developer Options, and then USB Debugging

Note

Up until Android 4.2, Developer Options was visible in the Settings menu. Android 4.2 and later versions have hidden developer options, and to see the menu, the examiner must go to About Phone in Settings and tap Build Number seven times. Developer Options will then be visible in the Settings menu.

In addition, when conducting forensic collections, you need to set the device to allow unknown sources in the Security settings on the Settings menu. Allowing unknown sources will enable the forensic software to "sideload" a special package file, called an Android application package (APK), onto the device. *Sideloading* an app means installing APK files that are not directly from the Google Play store. If the device will not allow unknown sources, the forensic software will not be able to install the forensic package to collect logical data.

Another setting—more of a security feature—is the acceptance of an RSA key when the device is connected to a computer (for devices running Android 4.2.2 and later). This must be accepted on the device, not the computer to which the device is connected. This RSA key acceptance is similar to selecting to trust the computer with an iOS device. By accepting the RSA key on the device, the examiner is stating that the connected device is safe to connect with the computer.

Once all settings have been configured, the device can be connected to the mobile forensic software and the collection can begin. Typically, the display and subsequent manual acceptance of RSA certificate will vary depending on the software that is used. For example, with Oxygen Forensic Detective, the RSA certificate screen will show upon communication with the device, after connection via USB cable.

Device State Considerations

If an Android device is unlocked and powered on, unlocked and powered off, or the security is enabled and the password is known, the connection and collection will be typical and straightforward. There are instances, however, when an unlocked Android device cannot be recognized by the software or even the forensic computer or hardware. In such instances, it is important that the examiner determine whether the device's USB port is intended for charging only or whether data can actually be transferred to and from the device via the port.

Here's an easy way to determine this: After you plug the device into a computer, if a device can transfer data, the device will be shown as a portable device within the computer's device tree, and a menu will be displayed on the mobile device screen with at least an option to turn on USB mode for the device. USB mode will allow the device to move files on and off the media portion. If the device is capable of doing this, it can be collected using ADB.

Tip

Android devices are predominantly collected using a USB cable connected to the computer running the forensic solution. In some situations, however, the use of alternative connections may be warranted.

Locked and Hidden USB Connections

Many "burner" Android phones will not show an ADB driver as being installed in the computer's Device Manager, even if ADB has been enabled in the Developer Options menu. Here's a perfect case in point: While assisting an investigator with the examination of an LG VS908 Android device, I found that the phone was not showing up to allow collection. The device was shown as a portable device and all the folders and files could be browsed in Windows Explorer, but ADB was not available in the Device Manager even when it was enabled in the Settings menu.

I had the investigator unplug and then plug in the device again from the computer several times to check for a bad connection. Still, ADB could not be installed, but a menu was enabled every time the device was plugged in. The menu presented three options: Charging Only, Media Sync, and Internet. Knowing that the Media Sync option would allow access only to the media card, I had the investigator select Internet. A submenu became available with Modem and Ethernet as choices. These options are the tethering options for the device, and if either is selected, ADB will be available in the Device Manager. Once he selected Modem, the investigator was able to process the device normally with his forensic software.

The settings to inhibit data transfer are very common with inexpensive Android devices. I have found submenu items on most of these devices that, when activated, will activate ADB that had been previously been turned on in the Developer Options menu.

If the device is locked by a password and the password is not known, the device must be physically collected, unlocked using software such as Cellebrite or Oxygen Forensic Detective, or JTAG/ISP/chip-off used to access the data. As mentioned earlier, in order for data transfer to occur using USB, ADB must be enabled. Android devices by default do not have ADB enabled, but an examiner should always plug the device into the forensic solution, even if it's locked, to see if ADB has been turned on. Users who frequently sideload applications or hack Android devices will have ADB enabled, and when it is available, the device will still be accessible if locked. However, if the device is locked and ADB is not enabled, logical access will not be available and the examiner will be limited to non-invasive physical or invasive physical collections.

Collecting the Device Logically

Once ADB is enabled, a logical collection of an Android device can occur. Forensic software vendors all use a customized APK file to collect this valuable user data. The examiner needs to understand firmly the method and installation of a program onto an Android device to collect data. Because the forensic software is writing to the evidence and altering the internal file system, a challenge could be raised that the evidence is no longer valid. This, of course, is not true, but without an understanding of the forensic tool and how the process works, an examiner may have a difficult time dealing with such a challenge.

An APK is a compressed file containing uncompiled code. This package contains files and folders that, when compiled on the Android device upon installation, create a port for which the forensic software can communicate and collect the user data from the mobile device. Each software vendor's APK is different: some extract user packages and Wi-Fi settings, while others do not; some remain installed on the device after running, while others do not. What is important, other than what type of data will be extracted, is how the APK is delivered, where it is delivered, and what happens to it once it has been installed and run. Typically, the APK is pushed onto the Android device and placed into the system/tmp folder using ADB. The APK is then installed and compiled onto the device and an instance is initiated, also via ADB.

Note

Most forensic software solutions instances are not visible on the device during the collection, with the exception of Oxygen Forensic Detective and Analyst, Compelson MOBILedit, and Cellebrite. These programs display a splash screen on the device while the app is running, collecting data.

While running, the app will be sending data to the forensic software based upon queries to the supported databases. This is an important concept for the examiner to understand, because a logical extraction of an Android device will not include the actual database files where the queries were directed. In simpler terms, if the SMS messages are collected from an Android device, the mmssms.db file is not extracted during a logical collection—only the data from the SQLite database table is extracted. Because the Android protects the data area of the file system, access to these database files is not available using SQL commands and is available only if the FLAG to include in backup is not FALSE for an ADB backup, or with a physical collection and elevated privileges, commonly referred to as *root privileges*. If database files, media files, or other files are extracted in a logical collection agent extraction, they will be extracted from the internal and/or external storage area using the Media Transfer Protocol (MTP).

Once the collection has completed, the forensic software will issue another command to the installed application to uninstall the software. The software is then removed from the target device and the system/tmp file will be cleaned and purged of any leftover data. If the examiner is unsure whether the forensic software removed the installed APK, he or she can navigate to the applications area on the Android device. If an icon for the forensic software APK is located, the app was not uninstalled and should be uninstalled manually.

Collecting the Device Physically

Collecting an Android device using non-invasive techniques requires *root access* to the device. Typically, if an Android device is supported in a forensic software solution for physical collection using a USB connection, two methods can be used, either *automated rooting* or *custom ROM* installation.

What Is Root Access?

On an Android device, there are levels of user access, much like on a Windows system with users and administrators. Android, based on Linux, has many user IDs that are granted access to perform various tasks on the device, such as run processes and control functions and calculations, but each type of user is restricted to performing particular privileges only. All Linux systems also have a superuser with a user ID of 0, called the root ID. This superuser can perform any task on the device, even if the file, folder, or setting has been marked as read-only. The superuser can change the access to read and write, access the area of the system that has been marked inaccessible, and more. This is called having “root access.” With root access to an Android device, the examiner can access any part of the device without limitation. No matter what method is used by a forensic software vendor, if access to the device is via a USB and a physical image is obtained, that software solution “rooted” the mobile device.

Note

Different root levels are available on a device. A *shell* root is the most desirable when you’re conducting mobile forensics because this can obtain root access to the device. Upon reboot, after the extraction is complete, the device will be back in the unrooted state. With a *permanent* root, on the other hand, a reboot will not remove the root access. Some third-party rooting tools can also "unroot" a rooted tool, and if available this should be attempted after the collection has completed. Leaving evidence in a rooted state could introduce malware or other issues that would not be possible with a secured device.

Automated rooting procedures require an unlocked phone with ADB available and enabled; to perform the rooting process manually, ADB also must be enabled and available. Forensic tools such as Oxygen Forensic Detective, UFED Touch 2/4PC, Magnet AXIOM, and XRY have built-in functions that will attempt to root the device when conducting a physical collection. If a tool is unable to root a device and no other methods for physical access are available, however, some third-party applications can be used: dr.fone from Wondershare supports devices from 2x to 5.1; KingoRoot for Android supports devices up to 4.2.2 and includes unroot capabilities; and SRSRoot supports Android versions up to 4.2 and has an unroot option, while One Click Root supports devices up to 6x for root as well as unroot. Other available tools also allow for the rooting of devices up to 7x, but they come with caveats of unlocking the bootloader (OEM unlock), which voids the warranty of the device.

If the device is locked, supported for physical collection, and ADB is disabled, the forensic software will connect to the device typically by using fastboot, recovery, emergency download (EDL) mode, or download mode. These modes are used to install customized ROM that enables ADB and grants access to the device with root access. Once the collection is completed, the original ROM is returned.

Caution

It is extremely important that the specific ROM matches that of the evidence to be collected when using a forensic solution with this type of physical method. Because of slight differences in the ROM for any Android device, using another device's ROM can cause disastrous results. This is primarily why this method fits in the invasive physical collection method category.

JTAG/ISP can also be used and has become extremely popular for Android collections. There are many different boxes and cables used in the mobile forensic community, but as with most everything else, no single mobile forensics solution supports every mobile device encountered. JTAG/ISP software typically displays the TAPs for the model to be collected in an image within the application. Following the image, the connection to the JTAG hardware can be made either by soldering to the correct port and then matching to the referenced port on the JTAG hardware, or connection can be made with other solutions such as Molex or jigs. When the device is connected and the JTAG software has been configured, a binary image is created and can then be imported into mobile forensic solutions such as Cellebrite Physical Analyzer, Oxygen Forensic Detective, and MSAB XACT.

Alternative Collection Methods

As with other device types, Android mobile devices can be collected using other, more raw, collection methods. If a physical collection is not available, these alternative methods can be used to gather information from an Android device over and above what a standard logical collection can gather, including file system and app data. These methods do not require having device root.

ADB Backup This function became available in Android 4.0 and enables a backup of individual apps or the entire app catalog along with the application data, cache, and internal storage. This function can be initiated using ADB.exe, available in the Android SDK in the /platform-tools/ folder. To use ADB Backup, the device must be unlocked and USB debugging must be enabled. Once an app or apps are downloaded using ADB Backup, the created file will be compressed and encrypted. To uncompress and decrypt the created file, the examiner runs the file through Android Backup Extractor (http://sourceforge.net/projects/adbextractor/), which will create a tarball file that can be examined using mobile forensic software capable of importing and viewing these types of files. Use of ADB Backup involves using a command line instruction and device interaction once ADB Backup is initiated on the mobile device.

Note

ADB Backup does not allow for the backup of Digital Rights Management (DRM) protected applications.

The parameters and flags needed to create an ADB Backup, along with a breakdown of each parameter, are shown next:

```
adb backup [-f <file>] [-apk|-noapk] [-shared|-noshared] [-all] [-system|-nosystem]
[<packages…>]
```

- **`[-f <file>]`** Specifies the location where the backup will be stored.
- **`[-apk|-noapk]`** Specifies whether or not the app APK will be backed up. Backing up the APK could allow for later scanning for malicious software. The `-noapk` flag is set by default.
- **`[-shared|-noshared]`** Specifies whether or not shared memory areas on the internal and external storage media will be backed up. The `-noshared` flag is set by default.
- **`[-all]`** Specifies all applications on the device. When conducting a forensic backup of the device, use the `-all` flag.
- **`[-system|-nosystem]`** Specifies whether or not ADB Backup includes system applications in the backup. The `-nosystem` flag is set by default.
- **`[<packages…>]`** Specifies the app that will be backed up. If this is set, only the specified app will be included (that is, com.android.provider.telephony).

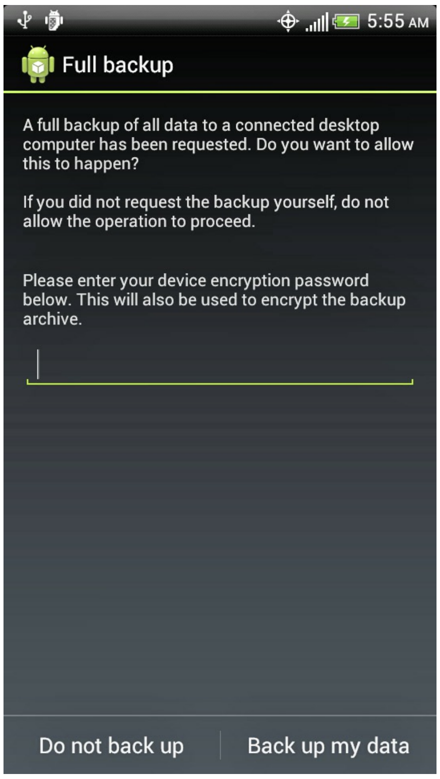
Here are the command parameters and flags for ADB Backup to collect all applications and internal/external media to a file called *backup* without system files at C:\Evidence:

```
adb backup -f C:\Evidence\backup.ab -all -nosystem -shared
```

If the examiner wants to extract all apps and memory contents of internal and external areas and their APKs to scan for malicious applications, the command would look similar to this:

```
adb backup -f C:\Evidence\backup.ab -apk -all -system -shared
```

When the command is complete, the examiner will be prompted with the screen displayed in Figure 10-9. Do not enter a password to encrypt the backup, and tap Back Up My Data. The duration of the process will depend upon the amount of data that is being backed up. Once the process completes, the examiner will be notified on the Android screen. The backup can then be extracted using the Android Backup Extractor and examined in the software solution of choice.



A full backup of all data to a connected desktop computer has been requested. Do you want to allow this to happen?

If you did not request the backup yourself, do not allow the operation to proceed.

Please enter your device encryption password below. This will also be used to encrypt the backup archive.

FIGURE 10-9 **Screen activated on a connected Android device when running the ADB backup command**

Helium Helium, formally known as Carbon, is an app-based solution that uses the front end of ADB Backup as well. To use Helium on a non-rooted device, install the application on the target device and on the desktop. An examiner can create a backup of select applications or individual applications to the PC; by default, the backup is saved to the SD card, but it can be added to Dropbox, Google Drive, and so on. As with ADB, some applications are not available for backup. What is quite interesting with this solution developed by Koushik "Koush" Dutta is the fact that the user simply connects the device and then untethers the device after connection; all extraction occurs over Wi-Fi (https://www.clockworkmod.com/).

Windows Explorer

Android devices are recognized as portable devices by both Mac and Windows systems. If a forensic solution is unable to extract the data from the internal or external memory areas, the examiner can copy the information from the mobile device to the location where the evidence is to be saved. Because dates will probably be altered using this method, it should be used only if there are no other alternatives.

Archiving the Data

All data, including physical images, logical file systems, backups, and UICC information, should be copied to an evidence drive before the analysis phase. The information that is needed for analysis should be copied to a temporary location. The storage location should have a unique folder specific to the case and an internal folder specific to the device. Any additional evidence recovered as part of the analysis should also be stored permanently in this location once the case is completed.

Chapter Summary

The collection of mobile device data is often the easiest part of mobile forensics, but only if device state, drivers, and workflow are considered appropriately. Thoroughly documenting the device, UICC, and memory cards is the first step, but the examiner must also determine what evidence should be processed first, depending on whether the device is powered on or off.

Logical examinations are just as important as physical collections, and both types of collections should always be attempted. Obtaining a logical collection first can usually help the examiner obtain valuable data in case a physical collection is not supported, does not successfully complete, or disables the mobile device. The device type (whether a feature phone, an Android, a BlackBerry, an iOS, or a Windows Phone) can also determine the order of processing and workflow.

Often, mobile device forensic software will use functions and technology to conduct logical collections that have been formulated from the device manufacturer's own backup tools. Because of this, an examiner can use the manufacturer's software to create a backup as an alternative. It is important to understand, however, that some settings might need to be adjusted to maintain a valid backup. Collection methods may not work or may be unsuccessful, but the initial preparation of the device prior to collection of the digital data should be the same, no matter what device is to be examined and what software is used.

By understanding the way forensic software communicates with a mobile device to obtain data, the examiner can be prepared to face many challenges. The expert examiner should know how to determine that an APK file is installed, that the Android device data is placed into a temporary area and later removed, that a custom RAM disk is created in the iOS volatile memory and later removed upon restart, and that obtaining root to an Android device changes security privileges and not the underlying user data.

11 Analyzing SIM Cards

Previous chapters covered procedures, seizure, and collection of physical and digital evidence on a mobile device. The next several chapters discuss the analysis of the data within the device files and folders. In this chapter, you'll learn about the subscriber identity module (SIM) card, once used only in GSM devices but now found in CDMA devices, and the evidentiary data that may reside within the file system.

SIM cards contain information that can be used by an examiner to locate a device, identify contacts, and see a user's communications. The examiner will need to concentrate on the card's file system—not an easy task, especially because SIM cards often store data in a different format than mobile device file systems and in various locations with different security levels. Nevertheless, the SIM card is well documented, which ensures that the manufacturers and cellular communication companies have a clear set of instructions regarding how to communicate within the GSM/UMTS network. The pertinent standards and documents are discussed throughout this chapter and can be referenced often if the examiner's case involves the analysis of a SIM card.

This chapter covers SIM and UICC (Universal Integrated Circuit Card) smart cards, the SIM file system, and the various artifacts that may be used in any investigation. This chapter does not cover all of the files and folders available on a SIM, because many of these are used strictly for network communication—hundreds of files are contained on a single SIM, and covering them all within a single chapter is not possible. You can find more information by consulting the materials mentioned throughout the chapter.

Smart Card Overview: SIM and UICC

The European Telecommunications Standards Institute (ETSI) defines *smart cards* as "micro-processor equipped tokens, able to store and process a diverse range of data and applications" (www.etsi.org/technologies-clusters/technologies/smart-cards). Many examiners use the terms "UICC" and "SIM" interchangeably, though, strictly speaking, UICC (Universal Integrated Circuit Card) is the latest generation of the SIM card. The UICC is the hardware portion of the smart card, while the SIM, USIM, and others are software applications included on the card. Initially, a SIM was used only for GSM networks as the key to the mobile device operating on a cellular network along with both the hardware (mobile device) and software component.

UICCs now are used in most smart devices, including CDMA devices. UICC covers the gamut of subscriber cards to include the removable user identity module (R-UIM) and CDMA2000 subscriber identity module (CSIM) cards, also part of the CDMA devices that can be used globally.

All the applications on the UICC are defined by ETSI and adopted by the Third Generation Partnership Project (3GPP). Applications (such as USIM, SIM, ISIM, and CSIM) contain information and help with flow of information to the mobile device depending upon the network used. If HSPA (High-Speed Packet Access), HSPA+, and LTE (Long-Term Evolution) networks are used in conjunction with the USIM, the ISIM handles the high-speed packet access (HSPA). If a UMTS network is being used, the USIM application will maintain control of communication and includes the data needed to help the device operate on that network; if a 2G or Enhanced Data rates for Global Evolution (EDGE) connection is needed on a

GSM network, the SIM application would be used. A SIM card containing only a SIM application cannot operate on a UMTS-only network, but a UICC with both the SIM and USIM application can operate on either a GSM or UMTS system.

The physical size of a UICC is described in ETSI TS 102 221; version 14.2.0 references both ISO/IEC 7816-1 and ISO/IEC 7816-2 for acceptance. The most significant change to the UICC has been the size of the smart card; Figure 11-1 shows the various size progressions of the card. The sizes are defined by ETSI as 1FF ID-1 UICC, 2FF Plug-in UICC, 3FF Mini-UICC, 4FF Nano-UICC, and M2M Form Factors (MFF1/MFF2) (embedded UICC). The ID-1 UICC was never used in a mobile device because of its large size (that of a credit card), but the Plug-in UICC is the most commonly referenced and was used in feature phones and early smart devices until 2004. The Mini-UICC was then released and made its way into most smart devices until the Nano-UICC was released in 2012 and adopted in newer iOS, Android, BlackBerry, and Windows Phones. The MFF1/MFF2, or embedded UICC, is a SIM typically soldered directly to the device's PCB and used for machine-to-machine (M2M) applications. The benefits are many and include longer life and ease of moving from one carrier to another because of over-the-air provisioning—not a change of UICC. MFF2 are utilized in many automobile infotainment systems, point of service (POS) systems, tablets, wearables, and IoT devices. ETSI has since released TS 103 383 outlining the eUICC (e-SIM) concept and its current use in today's businesses and devices. The eUICC should pose a unique problem with traditional mobile forensic examinations. Subsequently, Apple has expressed the most interest in using eUICCs within all of its products, from iPhones to iPads.

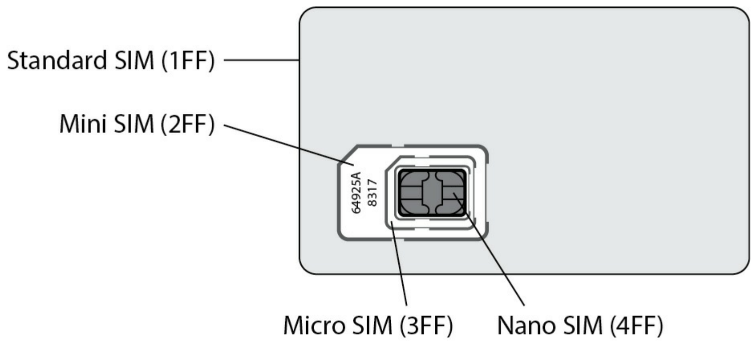


FIGURE 11-1 **The evolution of SIM/UICC size**

The actual microprocessor did not change between the SIM and Nano-UICC, but the shell that contains the microprocessor has changed. There was a debate between ETSI and Apple on Apple's ability to change the size of the microprocessor, and Apple finally won. Most UICC cards that are used in today's devices are still the original size for feature phones, but the majority of smart devices use the Nano-UICC.

SIM Card Analysis

Mobile forensic software replicates commands used by the mobile device to communicate with the applications on the UICC. The software can collect various data files using application protocol data unit (APDU) commands. These commands communicate with the mobile device and the UICC to obtain and store data by writing information to the UICC. A UICC is passive and does not initiate contact with the mobile device to begin a transaction; instead, it stands by to listen for the APDU commands sent from the mobile device. A mobile device, or the operating firmware on the mobile device, can request the device serial number, the last SMS message, and even the last known location. By replicating APDU commands when using mobile forensic software, the examiner can also request this information to be used for the investigation.

Before we move into the communication methods from the software solution to the UICC, you should understand the physical characteristics of a UICC. The UICC comprises a microprocessor (CPU), RAM, ROM for firmware, and Electrically Erasable Programmable ROM (EEPROM) for nonvolatile storage. The UICC accepts communication via the I/O contact point (Figure 11-2) from the mobile device, but to operate correctly, the UICC needs all points in contact with the mobile device terminal pins. The UICC surface is contacted by the six or eight points along reciprocal contact points on the mobile device—most UICCs in today's devices have six or eight pins. Reading, querying, and writing to the UICC occurs at these contact points (6 pin: VCC, GND, I/O, VPP, RST, and CLK / 8 pin: VCC, GND, I/O, VPP, RST, CLK, SIM_PRESENCE, and GND).

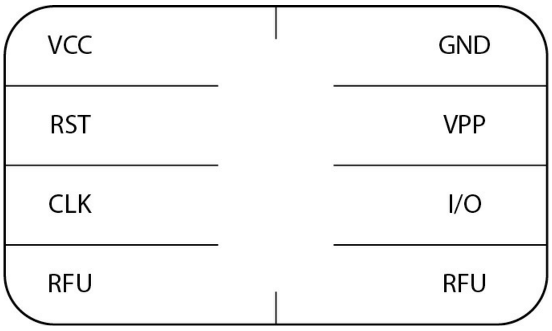


FIGURE 11-2 **UICC smart card contact points: communication to the chip occurs via the I/O contact, which is the pathway to the UICC CPU.**

As mentioned, the UICC is a passive part of the mobile device and must be contacted with APDU commands for action to occur. An APDU command comprises two components: a command and a response, as shown in Figure 11-3. The command is always initiated outside of the UICC (from forensic software or mobile device firmware), and a response is always returned, even if a command is incorrect. The UICC's response can indicate that a command was successful, unsuccessful, successful with security problems that must be addressed, or successful with a return of data.

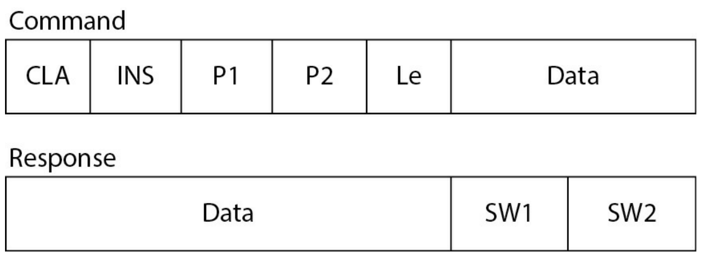
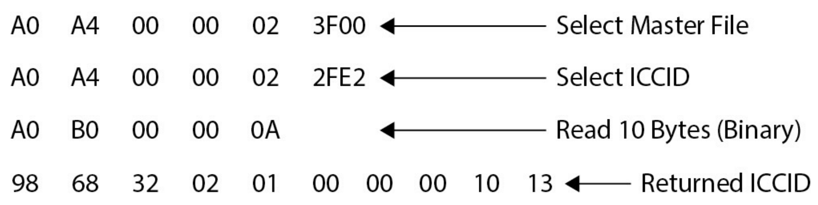


FIGURE 11-3 **APDU command and response structures**

An APDU command first needs to navigate the SIM card file system using INS (instruction) commands on the way to the appropriate file ID. During the navigation, if any security or permissions need to be satisfied, they, too, will have to be entered as APDU commands. Security conditions for SIM cards are referred to as ADM (Administrator) or CHV (Card Holder Verification). Once at the file ID, the APDU command must send an instruction (INS) that will tell the SIM what is to occur at the file ID. The instruction codes that can be used in an APDU are outlined in ETSI TS 102 221 (www.etsi.org/deliver/etsi_ts/102200_102299/102221/08.02.00_60/ts_102221v080200p.pdf). A file ID must then be "selected" to include additional instructions. To "select" a file ID, the APDU command must include A4 as the INS code. The ID for the file to be acted upon, if using the select INS, would then fill the Data portion of the command structure shown in Figure 11-3. Whatever the length of the data portion in bytes, this number will be added to the Le block of the command. A file ID is made up of 2 bytes, and its specifications are outlined in ETSI TS 131 102 and 3GPP TS 31.102.

The commands that would be issued to a UICC to retrieve the integrated circuit card identifier (ICCID) using a command APDU are shown in Figure 11-4. The first commands navigate to the ICCID via the file system, identifying the file and then sending the instruction code to read the number of bytes located within the record. The return command contains the ICCID indicating success. The ICCID is in a "reverse nibble" format (each byte is flipped to create the actual value—for example, 21 80 48 would be 12 08 84). Response codes are also listed within ETSI TS 102 221. This is the same process forensic software will use to read the UICC ICCID. The reading of various portions of the UICC is covered later in the chapter in the section "User Data Locations."



```
A0   A4   00   00   02   3F00 <----------- Select Master File
A0   A4   00   00   02   2FE2 <----------- Select ICCID
A0   B0   00   00   0A        <----------- Read 10 Bytes (Binary)
98   68   32   02   01   00   00   00   10   13 <---- Returned ICCID
```

FIGURE 11-4 **APDU commands to navigate and read the ICCIC**

File System UICC Structure

Obtaining the file system of all pieces of evidence is the goal of an examination, and it's the same with UICCs. Most information on the UICC is used for network communication and authentication, but the card can also contain valuable user information. Most mobile forensic solutions can obtain a UICC file system, with a few known exceptions—MOBILedit Forensic and Susteen Secure View obtain user data directly from the UICC and import the information into the user interface. Some solutions even query the UICC through the handset rather than via a secondary SIM card reader. This section will not cover these tools, but it will include the artifacts that can be found within the file system and their formats.

Note

The examiner will benefit from working through the UICC file system manually in the same way a forensic solution would work automatically. Understanding the method in which an automated tool obtains the data from these smart cards is an important feather in the mobile forensic examiner's cap, and it's a necessity for an examiner who wants to be an expert in mobile forensics.

Understandably, today's mobile devices contain little user information, simply because the data transmitted and received with a smart phone far exceeds the limited storage capacity of the UICC. However, mobile devices, including smart devices, can still store user data to the UICC that would otherwise be missed if the smart card were not examined. It is critical that the examiner obtain a full file system collection from the SIM/UICC, and I always recommended that the UICC card be collected and examined outside the body of the device, no matter the device, for reasons that include but are not limited to network location, networks accessed, subscriber information, SMS, contacts, and calls, to name a few. Some "fileIDs" that are discussed later are not extracted because of their protection status. Each of these records can also contain information that an automated tool either cannot parse or does not parse correctly, so manual inspection is sometimes necessary.

The UICC file system is made up of several levels that contain four types of files: a Master File (MF) and Dedicated Files (DF), which include Application Dedicated Files (ADF), and Elementary Files (EF). There is only one Master File on the UICC, and it is analogous to the root folder of a computer file system. The file ID of the MF is 3F00. There can be several DFs on the UICC (for example, GSM, DCS1800, TELECOM, USIM, PHONEBOOK), and these are analogous to a directory on a computer and files that are identified and described by ETSI TS 151 011 and 3GPP 51.011. The partitions on a SIM have always been described in forensic circles as SIM, USIM, CSIM, and ISIM (LTE, HSDPA+), but in reality they are applications that are used depending upon the type of network that is accessible.

Note

SIM partitions are actually applications, and within the applications are various files that are referenced by what is commonly referred to as "file IDs."

If a CDMA system is available, the CSIM partition/application is used, but if a UMTS system is available and preferred, the USIM partition/application is used. The UICC file system can be confusing, with several layers within each application and folder system, some containing duplicated information. This information is duplicated in the file system view but is actually written only once to the smart card, as is clearly observable when using the UICC file ID system. The file ID system simply references the needed record within the application. Because the UICC storage space is finite, a reference to this information must be made, and then, within the application, a pointer to the file ID is made. As mentioned, file IDs for most UICC data are defined and outlined within ETSI TS 131 102 and 3GPP TS 31.102. This does not mean that a file can be on the UICC only if the file ID is defined and listed by ETSI, however. In fact, a file can be created on a UICC that is specific to a particular carrier and used only by the carrier within its supported mobile devices. This is another reason the examiner should make sure to do a file system collection of all UICCs encountered.

Tip

If a UICC contains multiple partitions/applications, it is important to examine each referenced area, because a USIM application can contain additional phonebook entries over and above those in the SIM application.

Network Information Data Locations

Many files reside on the UICC card, most of which are network related. Generally, most files within the file system can be examined manually if needed. They contain information used by the mobile device to communicate on the network in a format that makes sense only to the network of the carrier system engineer. However, the examiner should be aware of several of these network-related files on the UICC when conducting a mobile device examination because they often contain data that could be used in a mobile device investigation.

Some files, such as the ciphering key (Ki), although important and necessary for the authentication process and contained on all SIM cards, are unavailable to the forensic examiner using any forensic software. Although some files are not available for examination, they should still be recognized and their existence documented.

ICCID

The ICCID is analogous to the serial number of the UICC; it represents the unique number assigned to that single UICC—that is, no other smart card on the cellular network uses the same unique number. Typically, this number is located on the exterior of the smart card, but not always. The number can be a maximum of 20 digits. The ICCID must always be available to the mobile device without limitation, primarily because this number will be used for the authentication process. This means that even if a UICC is locked with a PIN, the ICCID can be ascertained, and the examiner can then use the ICCID to obtain the PIN unblocking key (PUK) to change the PIN and access the UICC contents by sending the carrier of record the appropriate court order.

Tip

The "authentication process" is involved in accepting the subscriber onto a cellular network so that calls, data, and other transactions can occur. Without a successful authentication process, the mobile device will be unable to use the cellular network. Emergency service calls are an exception, however.

Table 11-1 shows valuable information such as the size, number of bytes, file ID, and additional information about the ICCID and its contents. Figure 11-5 shows the actual representation of the ICCID on the exterior of the UICC, and Figure 11-6 shows the output of the ICCID using mobile forensic software. The ICCID is stored in a reverse nibble format within the record found directly under the Master Files in EF_ICCID. So when the examiner is decoding the ICCID, each byte must be reversed to interpret the ICCID digits properly.

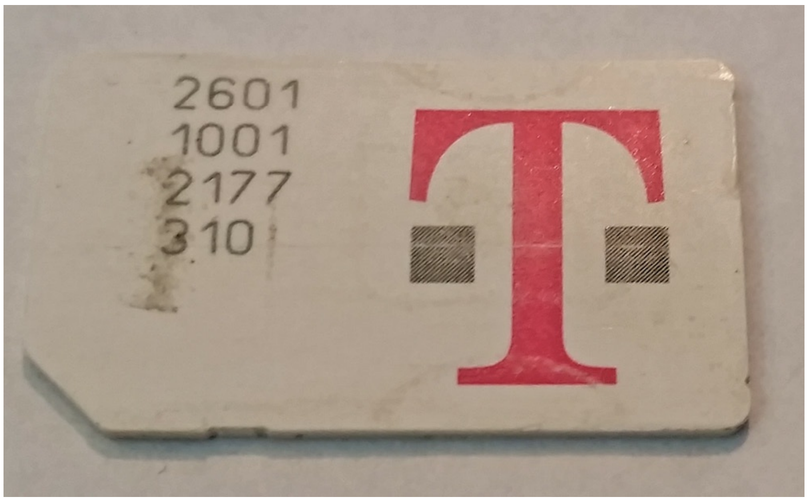


FIGURE 11-5 **UICC exterior with a visible ICCID**

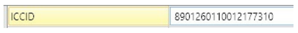

ICCID 89012601100012177310

FIGURE 11-6 **Decoded ICCID**

TABLE 11-1 **TS 102 221 Information on the ICCID**

File ID	Length (Bytes)	Bytes	Need
2FE2	10	1–10	Mandatory

The ICCID is made up of 10 bytes that comprise several values. The first two digits are the system code, a constant value for mobile devices, 89, which represents ISO 7812, "Telecommunications Administers and Private Operating Agencies." The next two or three digits represent the country code for the UICC. For the United States, the country code is 01. The next two or three digits represent the Issuer Identifier Number, which for a mobile UICC is analogous to the first two digits on a credit card that identify the credit card issuer (such as Visa or MasterCard), but for the UICC it would be T-Mobile, AT&T, Vodaphone, and so on.

Tip

To find other country codes, refer to https://en.wikipedia.org/wiki/Mobile_country_code or consult international numbering plans at www.numberingplans.com.

The first three data groups cannot exceed seven digits. The remaining digits represent the UICC number and are made up of the year and month of manufacturing, configuration specs, and the unique UICC number. The final digit is referred to as the *checksum* digit.

Consider, for example, the ICCID number 89310170105113168601:

89310170105113168601	System code
893**10**170105113168601	Country code
89310**170**105113168601	Issuer identifier number
8931017010**511316860**1	UICC number
8931017010511316860**1**	Checksum

IMSI

The International Mobile Subscriber Identity (IMSI) is the unique number that identifies the subscriber on the cellular network. This number is used to find a subscriber across the network and deliver calls, data, and other transactional information. Often, examiners think the phone number of the device is important to a cellular network. But it is the IMSI, for both GSM and CDMA networks, that is needed for contact to be made. This is an important file and should always be reported during the investigation. In fact, this record can be used to subpoena the records of the subscriber.

The IMSI is a protected file, and as such, if the UICC is locked with a PIN it will be inaccessible. The IMSI, under GSM/UMTS standards, will be 9 bytes, but the subscriber number can be a maximum of 15 digits. This is an interesting concept when the examiner recognizes that the total number of digits equals 18 when looking at the raw hexadecimal values under EF_IMSI in the UICC file system. The values are also stored in reverse nibble, with the first byte as the length, which will always equal x08. The preceding bytes compose the IMSI in reverse nibble, and in order to decode the digits, you must first reverse them; the first digit, which will always be 9, should be dropped because it is not part of the IMSI value but is a placeholder (that looks surprising when added to the length byte 89, indicating telecommunication like the ICCID). Table 11-2 shows information on the IMSI and its contents.

TABLE 11-2 **TS 151 011 Information on the IMSI**

File ID	Length (Bytes)	Bytes	Need
6F07	9	1 = Length 2–9 = IMSI	Mandatory

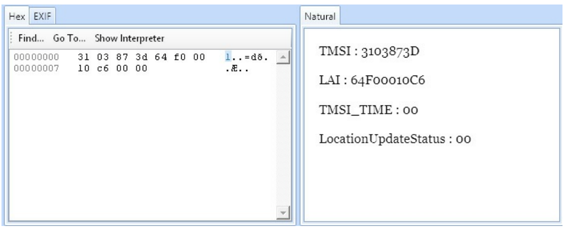
Much like the ICCID, the digits within the IMSI represent data to the examiner. The first 3 bytes are the mobile country code (MCC), the next two digits are the mobile network code (MNC), and the remaining digits are the subscriber identification number. The following shows the extracted IMSI using mobile forensic software and the breakdown of the IMSI number. Consider, for example, the IMSI number 310260123456789:

310260123456789	Mobile country code
310**260**123456789	Mobile network code
310260**12345678**9	Subscriber identification number

LOCI

The location information (LOCI) is an elementary file under the DF_GSM (DF structure for GSM) that identifies the geographic area *where the device was last successfully powered off.* The last portion of the sentence is critical: The LOCI writes the last tower location to which the device was registered when a successful power-off occurred so that the device can access the network quickly upon powering on. If the device's battery is removed, if it is not powered off correctly, or if the device is broken prior to powering off, this file may not be available or may be incorrect. If, in fact, the LOCI file contains data, the examiner can use this key to identify the geographic location by contacting the carrier of record with the key information. The LOCI comprises the TMSI (temporary mobile subscriber identity), LAI (location area information), TMSI time, and the location update status.

The TMSI, as indicated by its name, is a temporary random ID that is assigned via the VLR (visitor locator register) to the subscriber; the actual IMSI is not sent out via the handset because of possible capture and identification of the mobile subscriber, and the temporary IMSI changes when the device moves to a different VLR. The LAI comprises the MCC (mobile country code), the MNC (mobile network code), and the LAC (location area code). By consulting the web site www.mcc-mnc.com, the examiner can at least identify the country and carrier to contact to obtain possible subscriber information. Table 11-3 shows the identifiable information for the LOCI, and Figure 11-7 shows an example from an actual GSM device.



```
00000000   31 03 87 3d 64 f0 00    1..=dð.
00000007   10 c6 00 00             .Æ..
```

TMSI : 3103873D

LAI : 64F00010C6

TMSI_TIME : 00

LocationUpdateStatus : 00

FIGURE 11-7 **LOCI and interpreted information from a mobile forensic solution**

TABLE 11-3 **TS 151 011 Information on the LOCI**

File ID	Length (Bytes)	Bytes	Need
6F7E	11	1–4 = TIMSI 5–9 = LAI 10 = RFU 11 = Location update status	Mandatory

FPLMN

Forbidden Public Land Mobile Network (FPLMN) is an elementary file under DF_GSM that identifies networks with which the device carrier does not have a valid agreement to access. In other words, a PLMN (both MCC and MNC) will be written to the FPLMN if the network rejects the location update, because PLMN is not allowed. This elementary file once had a limit of four records, but with the new UICC specification, it can hold up to *n* records, depending upon the card manufacturer and carrier specifications. When a record is added to the FPLMN elementary file, that record is placed after the last record; if there are no additional slots, the first record is removed and the new record is added to the last available slot.

In terms of value for the examiner, the FPLMN identifies specific country codes along with the carrier of record, which could also identify a geographic region in which the mobile device was being used or attempted to access. If a user, for example, had stated that he was not in a certain country and admitted to having control of the mobile device, but access was attempted to a forbidden network in the country the user denied being in, an argument could be made that the user was indeed within the country. This information could be based upon the evidence located within the FPLMN elementary file. Table 11-4 shows the technical information regarding the FPLMN format.

TABLE 11-4 **TS 151 011 Information on the FPLMN**

File ID	Length (Bytes)	Bytes	Need
6F7B	12	1–3 = PLMN 1 4–6 = PLMN 2 7–9 = PLMN 3 10–12 = PLMN 4	Mandatory

User Data Locations

When examining the information from a mobile device UICC, forensic applications primarily focus upon personal data that could be stored on the SIM. With the storage capacity of today's mobile devices, the storage of personal data to the UICC is not the norm. Mobile device settings default to storage to the handset, and a user would have to change the storage location settings manually for the UICC to be used. Because the capacity of a SIM is finite, as set by standards, along with the mass amount of personal data that is transmitted and received by a mobile device, the storage of data that can be used for investigations is limited. Previously, the portability of a UICC could enable users to take their SIM with them, complete with their contacts and SMS messages, place the SIM into another GSM device, and be able to be up and running with their personal information, subscriber information, and network details. With the movement to world phones that cover the necessary cellular bands globally, there is no need to have multiple devices to cover a particular cellular band using the singular UICC. Because the need for portability is no longer a necessity, carriers and phone manufacturers alike no longer store data to the UICC by default. This does not dismiss the importance of collecting the UICC content, however.

Mobile forensic tools concentrate on several areas to recover this personal data within the UICC, as listed in the following section. The examiner should understand the location, format, and structure of these areas to help validate the recovery of the automated mobile forensic software solution. Again, understanding how an automated tool identifies, converts, and displays mobile device data will satisfy another step in your becoming a mobile forensic expert.

SMS

The Short Message Service (SMS) is another elementary file located on the UICC. It contains several records that define many aspects of the message and service. Either 3GPP TS 23.040 or ETSI 123 040 for UMTS defines these records as messages that either originate from the mobile equipment (ME) or are received from the subscriber's network. The record length cannot be more than 176 bytes, with the first byte being a status of the record and the remaining bytes making up the architecture of the message. Table 11-5 shows the file information obtained from ETSI 123 040 with specific information on the format for EF_SMS files.

TABLE 11-5 TS 151 011 Information on the SMS

File ID	Length (Bytes)	Bytes	Need
6F3C	176	1 = Status information 2–176 = Remainder	Optional

The status byte of the record, shown in Table 11-6, will tell the examiner several interesting facts. If the status byte indicates “Unused” and content is still contained within bytes 2–176, the examiner should assume, with great certainty, that the message has been deleted. The ME simply changes the status of the record and does not remove the content. Since the status shows “Unused,” the record slot is available and thus can be overwritten with a new message, but until that time the record is available to be recovered by the examiner. The status byte can assist in an investigation if the SMS was alleged to have been sent, but did not have service, or if it had been submitted but was rejected by the network and was never delivered. Using the status byte, an examiner can make this determination.

TABLE 11-6 Message Status Byte from ETSI TS 131 102 V9.5.0 (2011-01)

Binary Value	Status	Hexadecimal Value
00000000	Unused	x00
00000001	Mobile terminated, read	x01
00000011	Mobile terminated, unread	x03
00000101	Mobile originated, sent to network	x05
00000111	Mobile originated, message to be sent (unsent)	x07
00001101	Status report requested but not (yet) received	x0D
00010101	Status report requested, received, but not stored in EF-SMSR	x15
00011101	Status report requested, received, and stored in EF-SMSR	x1D

Note

It is important that the examiner understand that the recovery of “slack space,” or partial SMS messages, from a SIM card is not possible because of the way a record is written to the card. When an SMS message is submitted to be 281written to a SIM card and an open “unused” slot is identified, the 176-byte record is then written to the SIM. It does not matter if the message occupied only 56 bytes; all 176 bytes will be written to the record, overwriting any record that had previously existed. When is SIM card is written to, it is all or nothing.

Bytes 2–176 make up the content with a set length for the actual message content depending upon how the data is formatted; this is often referred to as the Transport Protocol Data Unit (TPDU). The TPDU contains a variety of data, depending upon whether the message was sent (SMS-SUBMIT) or received (SMS-DELIVER). For an SMS-SUBMIT message, Table 11-7 shows the various types of data stored within the TPDU. For an SMS-DELIVER message, the content differs slightly, as shown in Table 11-8. Each type of content is covered in the following tables with specific comments and explanations for better understanding.

TABLE 11-7 -SUBMIT Structure from TS 100 901

Type	Description	Need
TP-MTI	TP-Message-Type-Indicator	Mandatory
TP-RD	TP-Reject-Duplicated	Mandatory
TP-VPF	TP-Validity-Period-Format	Mandatory
TP-RP	TP-Reply-Path	Mandatory
TP-UDHI	TP-User-Date-Header-Indicator	Optional
TP-SRR	TP-Status-Report-Request	Optional
TP-MR	TP-Message-Reference	Mandatory
TP-DA	TP-Destination-Address	Mandatory
TP-PID	TP-Protocol-Identifier	Mandatory
TP-DCS	TP-Data-Coding-Scheme	Mandatory
TP-VP	TP-Validity-Period	Optional
TP-UDL	TP-User-Data-Length	Mandatory
TP-UD	TP-User-Data	Optional

TABLE 11-8 SMS-DELIVER Structure from TS 100 901

Type	Description	Need
TP-MTI	TP-Message-Type-Indicator	Mandatory
TP-MMS	TP-More-Messages-to-Send	Mandatory
TP-RP	TP-Reply-Path	Mandatory
TP-UDHI	TP-User-Date-Header-Indicator	Optional
TP-SRI	TP-Status-Report-Indication	Optional
TP-OA	TP-Originating-Address	Mandatory
TP-PID	TP-Protocol-Identifier	Mandatory
TP-DCS	TP-Data-Coding-Scheme	Mandatory
TP-SCTS	TP-Service-Center-Time-Stamp	Mandatory
TP-UDL	TP-User-Data-Length	Mandatory
TP-UD	TP-User-Data	Optional

The breakdown, description, and decoding of the TPDU are covered in the following list:

- **Length of SMSC (Short Message Service Center) information** This is the number of octets (8 bits or 1 byte) that will be used to store the type of number and also the number of the service center. Typically, SMS-SUBMIT messages have the value 00, which indicates the use of the service center number internal to the mobile device. Not all handsets have this value, and if it is missing, it means the SMSC will be obtained from the handset along with the TON/NPI and service center number.
- **Type of number and numbering plan indicator (TON/NPI)** This single octet indicates the type of number that the preceding telephone number will represent. This byte is representative of the binary number created, with the first bit always 1, and is combined with the TON, which is 3 bits, and the NPI, which is 4 bits. Table 11-9 represents the various combinations and values for TON and NPI values. Once the correct hexadecimal number is located, the number will be converted to a binary number that will be used to decipher the type of number and number plan used in the message. For example, a common value is x91, which, when converted to binary, is 1 001 0001 and will indicate that a plus sign (+) is attached to the number in front of the country code. Using Table 11-7, the examiner would be able to confirm that this number is an international number conforming to the ISDN/telephone numbering plan by dropping the first MSB (always 1), decoding the following 3 bits for the TON and the remaining 4 bits for the NPI.

TABLE 11-9 **Binary Representation of TON/NPI Key of SMS Messages**

TON Binary	Interpreted	NPI Binary	Interpreted
000	Unknown	0000	Unknown
001	International number	0001	ISDN/Telephone Numbering Plan
010	National number	0011	Data Numbering Plan
011	Network-specific number	0100	Telex Number Plan
100	Subscriber number	0101	Service Center Specific
101	Alphanumeric (7 bit)	0110	Service Center Specific
110	Abbreviated number	1000	National Numbering Plan
111	Reserved	1001	Private Numbering Plan

- **Service center number** This value is the representative number of the service center that was used to route the SMS message. This value per ETSI TS 123 040 V12.2.0 (2014-10) is stored in semi-octets in binary-coded decimal (BCD) format, as defined in GSM 44.008 and again in ETSI TS 124 008 V11.8.0 (2013-10). This format is also reverse nibble. Often, the numbering does not complete an octet, so an F is added to complete the octet.
- **First octet of Short Message Transfer Protocol (SM-TP)** This single byte indicates the type of message from the six defined types in ETSI GSM 03.40. This hexadecimal byte should be converted to binary, and the two least significant bits are used to determine the type of the SMS message. Table 11-10 lists the various types of messages that are identified by the two least significant bits (1, 0) in the first octet of the SM-TP message. These bits are referred to as the TP-MTI (message type indicator).

TABLE 11-10 **Binary Representation of SM-TP Byte of SMS Messages, which Indicates the Message Protocol Used**

TP-MTI	Direction	Message Type
0 0	MS → SC	SMS-DELIVER-REPORT
0 0	SC → MS	SMS-DELIVER
0 1	MS → SC	SMS-SUBMIT
0 1	SC → MS	SMS-SUBMIT-REPORT
1 0	MS → SC	SMS-COMMAND
1 0	SC → MS	SMS-STATUS-REPORT

- **Address length** A single octet represents the length of the actual sender number. This byte will need to be converted to decimal to obtain the number of nibbles that are represented by the telephone number. Unlike the SMSC length, this value will not include the following byte that indicates the TON/NPI.

- **Type of Number and Numbering Plan Indicator (TON/NPI)** Table 11-9 can again be used to determine the number plan type and type of number of the sender.
- **Sender number** The sender number is again in semi-octets or reverse nibble (BCD), as was the case with the service center number.
- **TP-Protocol-Identifier (TP-PID)** This octet will identify the protocol that has been used for the transmission of the message. For standard ME-to-SC communication, this value will most likely be 00 as defined by ETSI TS 123 040 V12.2.0 (2014-10).
- **TP-Data-Coding-Scheme (TP-DCS)** This octet represents the coding that has been used to encode the message. This value will assist the ME in decoding the format once received. As with all other values, this octet, when converted to binary data, can be interpreted to determine the value. In Table 11-11, values are given as represented in 3GPP 3G TS 23.038 V2.0.0 (1999-06). Most often, the value for the TP-DCS will be 00 to indicate that the default 7-bit data code scheme will be used. However, in countries such as China, Korea, Japan, and others that use characters outside the ASCII range, this value will be different, because UCS2 will most likely be used. An example would be x04 in the TP-DCS section of the SIM record. The x04 in binary would be 01 00, which indicates 8-bit data and class 0 message.

TABLE 11-11 **Binary Representation of TP-DCS Byte of SMS Messages, which Indicates the Data Coding Protocol Used**

Bits 3 and 2	Translated	Bits 1 and 0	Translated
00	Default alphabet	00	CLASS 0
01	8-bit data	01	ME specific
10	UCS2	10	SIM-specific message
11	Reserved	11	TE specific

- **TP-Service-Center-Time-Stamp (TP-SCTS)** This value is represented by semi-octets and reverse nibble (BCD) in the ordering of Year, Month, Day, Hour, Minute, Second, and Timezone. The Timezone is the number of quarter hours from the local time to GMT time, and the most significant bit of the first octet indicates whether the number is positive or negative GMT. The Timezone is significant because if the ME has knowledge of the local time zone, the ME can display the received time in the local format. The time zone and time are local to the sending entity, which is important for the examiner to understand, especially when time and date are critical to the case.
- **TP-User-Data-Length (TP-UDL)** This integer value, represented in hexadecimal, is the length of data that is contained within the message. This value is also determined by the TP-DCS, or data format, discussed earlier. If the TP-DCS is the default, 7-bit is the length represented by septets (2 bytes), and 8-bit and UCS2 are represented by octets (1 byte). After converting this number into a decimal value, the examiner can identify the length of the message data immediately following as displayed in octets.

  The message maximum length set by standards is 140 bytes. If the message is formatted using the 7-bit

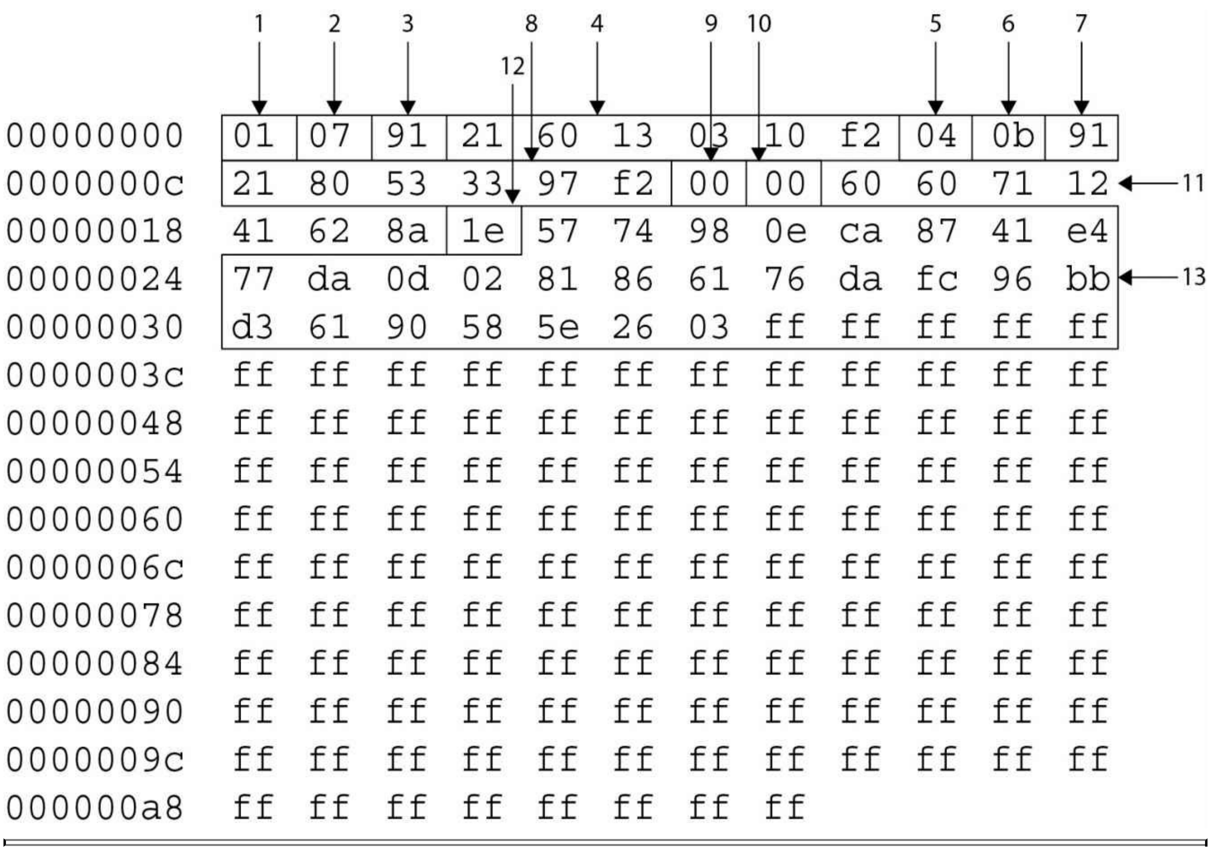
GSM alphabet, the records shall not exceed 160 characters. However, if the message is formatted using 8-bit, the record content shall not exceed 140 characters. To make matters more confusing, when you're examining SMS output and UCS2 coding (Unicode) is used in the message content, the message length cannot exceed 70 characters using the 16-bit UCS2 alphabet format. All formats are documented in 3GPP TS 23.038. The various coding formats for the message allow for the transmission and reception of messages in multiple languages and also messages with more than the default 140 characters. The 7-bit or the GSM alphabet as defined by 3GPP TS 23.038 is the default format and is mandatory for network providers, but countries that use languages not supported by the extended ASCII table of the GSM alphabet (China, Korea, Japan) use the UCS2 (16-bit) format. The ME will always default to the 7-bit format, but as soon as a character is entered that is not part of the 7-bit GSM alphabet, the entire message is re-encoded into UCS2.

- **TP-User-Data (TP-UD)** The user data portion of the SMS contains the message in 7-bit, 8-bit, or UCS2 format as specified in the TP-DCS. This data is represented in forensic tools as hexadecimal values.
- **TP-Message-Reference (TP-MR)** The single octet is included in sent messages that indicate the integer value of a message reference. This value will typically be x00 but can be a value from 0 to 255.

Both SMS-SUBMIT and SMS-DELIVER use a combination of the items outlined in this section and others outlined in ETSI TS 123 040. Figure 11-8 shows an actual SMS-DELIVER message, and Table 11-12 describes the values and decodes the data in the figure. These illustrations can assist the examiner in locating key artifacts within the SMS message that are typically observed in a UICC file system extraction.

TABLE 11-12 **Corresponding Values with Figure 11-8 with Decoded Data for Values Listed from an Actual Message**

Index Number	Value Description	Decoded Data
1	Status byte	01 = Mobile terminated, read
2	Length of SMSC information	7 bytes
3	Type of SMSC	91 = International format
4	SMSC number	12063130012
5	First octet of SMS-DELIVER message	04 = SMS-DELIVER
6	Length of sender address	0b = 11
7	Type of address of sender number	91 = International format
8	Sender number	12083533792
9	Protocol identifier	00
10	Data encoding scheme	00 = Default
11	Time stamp	06/17/06 21:14:26
12	Length of user data	1e = 30
13	User data	What ya doin California bred



```
00000000  01 07 91 21 60 13 03 10 f2 04 0b 91
0000000c  21 80 53 33 97 f2 00 00 60 60 71 12
00000018  41 62 8a 1e 57 74 98 0e ca 87 41 e4
00000024  77 da 0d 02 81 86 61 76 da fc 96 bb
00000030  d3 61 90 58 5e 26 03 ff ff ff ff ff
0000003c  ff ff ff ff ff ff ff ff ff ff ff ff
00000048  ff ff ff ff ff ff ff ff ff ff ff ff
00000054  ff ff ff ff ff ff ff ff ff ff ff ff
00000060  ff ff ff ff ff ff ff ff ff ff ff ff
0000006c  ff ff ff ff ff ff ff ff ff ff ff ff
00000078  ff ff ff ff ff ff ff ff ff ff ff ff
00000084  ff ff ff ff ff ff ff ff ff ff ff ff
00000090  ff ff ff ff ff ff ff ff ff ff ff ff
0000009c  ff ff ff ff ff ff ff ff ff ff ff ff
000000a8  ff ff ff ff ff ff ff ff
```

FIGURE 11-8 Actual hexadecimal message from an SMS message

Notice that sometimes characters are within BCD values and must be interpreted as a 0, but in Table 11-13, the extended BCD characters are identified. These can be seen within HEX values throughout SIM records.

TABLE 11-13 **Extended BCD Values**

BCD	Character/Meaning
0	0
9	9
A	*
B	#
D	"Wild" value; this will cause the MMI to prompt the user for a single digit (see GSM 02.07 [3])

Contacts

Contacts on a UICC are referred to as Abbreviated Dialing Numbers (ADN). This elementary file is located on a SIM application under DF_Telecom and can also be under DF_Phonebook on a USIM application. Before the UMTS and USIM applications, the SIM contained only a single phonebook located under DF_Telecom, where it still is today on the UICC. When the USIM application was added to the UICC, two phonebooks were possible. For an examiner, this often led to problems when using a software solution that was unable to obtain both the USIM and SIM applications' stored data. This was typically nontransparent to the examiner and often went unnoticed. Today, mobile forensic solutions support both application phonebooks.

Often these phonebooks coexist and contain duplicate records, but it is entirely possible that the records under DF_Phonebook on the USIM are unique and accessible only by reading the DF_Phonebook on the USIM application. The phonebook located on the USIM application is under the DF_Phonebook, which is under DF_Telecom. This is referred to as the global phonebook. Multiple phonebooks are available under the DF_Telecom/DF_Phonebook and can be application-specific. This means that a single phonebook under DF_Phonebook can be used by a single application on the mobile device up to 250 record entries. UICCs with the DF_Phonebook can hold thousands of contact names and numbers, and this dedicated file should be investigated. The contacts located under the USIM and SIM application under DF_Telecom/EF_ADN hold a maximum of 250 records and are duplicated between applications (see Figure 11-9). Contacts can be located in several places within a mobile device's UICC, but the information that can be stored is much different from what is observed within the contacts storage location of a smart phone (such as photos, addresses, and usernames).

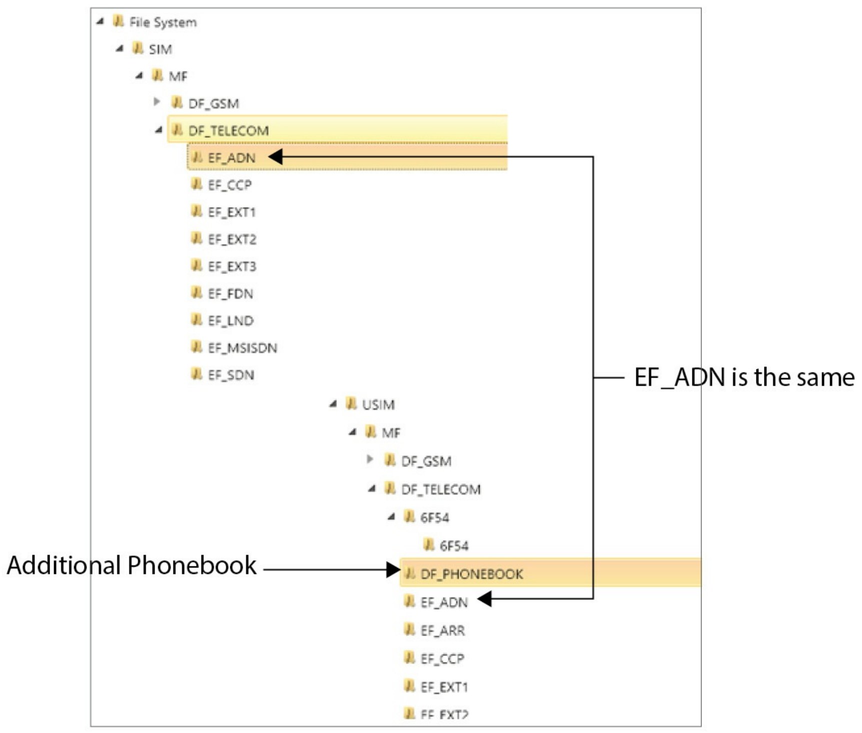


FIGURE 11-9 **Both the USIM and SIM applications can hold ADN records; one EF_ADN will be duplicated between the USIM and SIM.**

Like other data on the UICC, the data within the ADN record is coded in a semi-octet (BCD) format with the possibility of an alpha identifier, which is a name associated with the listed phone number. When an alpha identifier is used, the format, as specified by standards in 3GPP TS 23.038, will be 7-bit GSM alphabet, left justified, and all unused bytes will use FF or UCS2 format. The alpha identifier can be 0 to 242 bytes in length, and the rest of the record, to include the ADN length, TON/NPI, ADN, configuration record, and extension record, must be 14 bytes.

Much like the SMS record, the ADN will be coded in BCD format, preceded by the length of the ADN and the TON/NPI. The length is much like the SMS embedded address length discussed previously. Using Table 11-9, the Type of Number and Numbering Plan Indicator (TON/NPI) can be determined by the examiner before decoding the actual ADN that immediately follows. The ADN, as previously indicated, is coded in semi-octets or BCD format just like the service center and sender number in the SMS records. The balance of the ADN, if larger than 20 characters, will be written to an extension file under EF_EXT1 and is indicated in the last byte of the ADN record. The preceding byte, called the configuration record or capability record, can indicate whether additional configuration or capabilities are needed for the call and points to a record in the EF_CCP1 file. The complete breakdown of an actual record in an EF_ADN record is shown in Table 11-14.

TABLE 11-14 **Abbreviated Dialing Number (EF_ADN) Are Contacts in Both the USIM and SIM Applications**

File ID	Length (Bytes)	Bytes	Need
4F3A	$n + 14$	1 to n = Alpha identifier n +1 = Length of BCD number n + 2 = TON/NPI n + 3 to n + 12 = Dialing number n + 13 = Capability/configuration n + 14 = Extension1 record identifier	Optional

It is extremely important that the examiner understand the layout and limitations of storage within an ADN record, because overflow of data could reside within another file that the forensic software quite possibly will miss. It is also important to examine the UICC to locate data not recoverable or missed by forensic software. For example, if a record from the EF_ADN is deleted, because EF_ADN does not employ a status byte, the entire record is overwritten with FF when the record is deleted from the SIM. A deleted contact is not recoverable. However, records in the DF_Phonebook do not have to be contiguous, and noncontiguous records could indicate that the record was deleted before another contact could be added to the phonebook. This could be valuable information for the investigation.

Fixed Dialing Numbers

A UICC contains a phonebook that can be set to allow only numbers within the EF_FDN to be dialed from the mobile device. This phonebook, once activated, is protected by Card Holder Verification 2 (CHV 2), often called PIN 2, which must be entered to access and edit information. By the user disabling the EF_IMSI and EF_LOCI functions of the device for any other number, the mobile device cannot call any numbers other than what is included in the fixed dialing list. Of course, emergency numbers can operate without a SIM inserted into a mobile device, so this does not affect emergency services when dialing out. The EF_FDN is located under DF_Telecom in both the SIM and USIM applications.

Companies, and often parents, can restrict a mobile device to allow only specific call-out numbers but still receive incoming calls. At times, this is also used to store a phonebook that is protected by a layer of security. This enables a user to conceal a set of names and numbers and, in essence, creates a secret phonebook. As an investigator, being aware of the location of the EF_FDN is important, because no mobile forensic solutions currently check the fixed dialing numbers elementary file.

The layout of the EF_FDN is exactly the same as that of the EF_ADN with one exception: if the record exceeds the size allotted for the record, the overflow will be stored in EF_EXT2 instead of EF_EXT1 as with EF_ADN records.

Call Logs

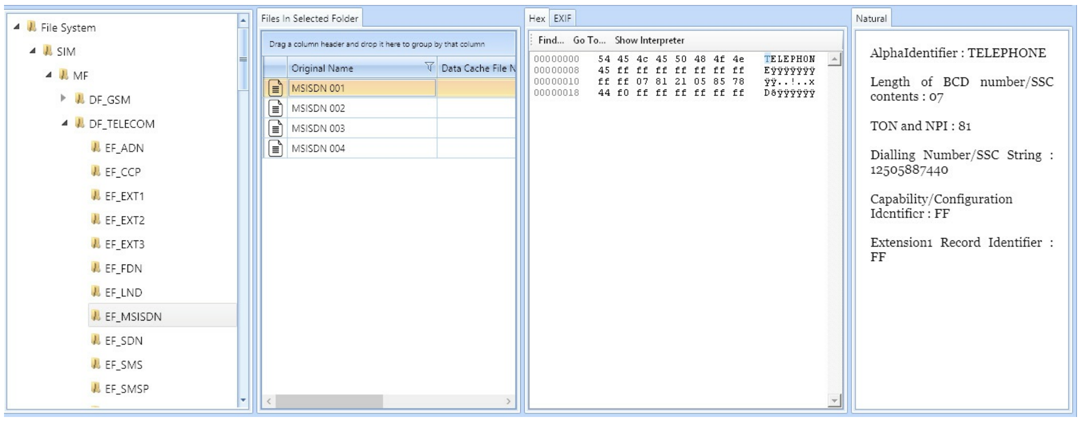
A UICC stores only the last numbers dialed (LND) and does not store incoming calls to the ME's SIM memory. Incoming calls would be stored on the device itself. The elementary file LND is located under DF_Telecom in both the SIM and USIM application. The LND record is similar to the EF_ADN in the storage capacity and data layout. The LND record can store an alpha identifier (name), a byte to identify the length of the phone number, a byte for TON/NPI, the actual dialing number in BCD format, a configuration/capability byte, and finally the extension byte.

The EF_LND has limited records, depending on the carrier, but typically no more than ten records can be stored to the UICC. When a new call is made and all records are occupied, the first record is removed and all records then shift up, with the new record taking the last position. Most of today's mobile devices do not store the call history directly to the mobile device, but it is important for an examiner to understand the format, layout, and location of this valuable data should it be populated.

Dialing Number

The dialing number for the mobile device is not necessarily important to the mobile carrier; the carrier relies on the EF_IMSI to identify the mobile user within the network, so a dialing number does not necessarily need to be stored on a UICC. If during an examination the EF_MSISDN is not located on the UICC, this does not indicate that the device was not used or is not in service. As described, the ability for a mobile device to use the cellular network is not dependent on the MSISDN, and only a valid EF_IMSI is needed. However, the EF_MSISDN is located under the DF_Telecom for both SIM and USIM applications on a UICC. The EF_MSISDN can include several records within the file, depending upon the carrier. Multiple EF_MSISDN records enable the user of the UICC to have a phone number for a business, personal use, fax, or other, with only a single EF_IMSI. This enables the device to be associated to multiple dialing numbers depending on which number was called.

The format structure for the EF_MSISDN and records is the same as that of the EF_ADN and should be used if the records are to be decoded manually. Figure 11-10 shows the EF_MSISDN records and first record of an actual UICC.



Original Name	Data Cache File N
MSISDN 001	
MSISDN 002	
MSISDN 003	
MSISDN 004	

```
00000000   54 45 4c 45 50 48 4f 4e   TELEPHON
00000008   45 ff ff ff ff ff ff ff   Eÿÿÿÿÿÿÿ
00000010   ff ff 07 81 21 05 85 78   ÿÿ..!..x
00000018   44 f0 ff ff ff ff ff ff   Dðÿÿÿÿÿÿ
```

AlphaIdentifier : TELEPHONE

Length of BCD number/SSC contents : 07

TON and NPI : 81

Dialling Number/SSC String : 12505887440

Capability/Configuration Idcntificr : FF

Extension1 Record Identifier : FF

FIGURE 11-10 **The MSISDN within the file system, its data, and decoded data**

The UICC contains a large number of files that can be examined and verified (see Figure 11-11). It is just a matter of how important the UICC file system might be to the examination.

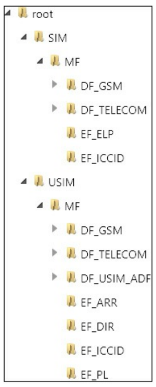


FIGURE 11-11 **File system view of a UICC card and the many folders available to the examiner**

Chapter Summary

UICC cards in today's devices might not contain a massive amount of user-related data, such as SMS, contacts, or last numbers dialed. However, the UICC is always needed by a GSM/UMTS when operating on a cellular network.

Even though a UICC is unable to hold data such as videos or other large files, it can hold hundreds of files within the card's file system and applications. Because the UICC must always be inserted into the device while operating on the cellular network, some type of evidence can always be collected—it might be information on last location, networks used, or the vital IMSI needed to crack a case. The UICC should never be neglected in any investigation.

Tip

Remember that CDMA devices also must always use a UICC when operating on the GMS/UMTS band.

The UICC data storage methods and structure are clearly documented by standards. This type of documentation can be used to help the examiner understand how to interpret the information contained on the UICC if the forensic software is unable to recover the data or the type of data is not supported.

This chapter introduced the examiner to the decoding of SMS, ADN, LND, IMSI, and ICCID, among other key artifacts. This information will be critical when examining other file system data contained within feature phones and today's smart devices. A lot of the storage formats, such as 7-bit and reverse nibble, are used throughout the industry, and understanding them will aid the examiner in other decoding exercises within the book and also during actual mobile device investigations.

12 Analyzing Feature Phone, BlackBerry, and Windows Phone Data

In the previous chapter, you were exposed to the advanced analysis of UICC data contained within the file system, which is a small part of the bigger evidence picture. Collecting mobile devices ranging from the feature phones to smart devices involves a connection using supported methods (USB, Bluetooth, JTAG, ISP, chip-off), an understanding of supported capabilities of the forensic solution, and a procedure for handling the collected digital evidence. What happens next with the data? Proper analysis of the mobile forensic evidence can uncover information that often makes an ordinary case extraordinary.

Note

Analyzing iOS and Android devices are covered in Chapters 13 and 15.

Feature phones are not as prevalent as they used to be, but they occasionally show up in forensic examinations. Typically branded "dumb phones," an inverse to a smart phone, the feature phone can contain information over and above what is collected—or not collected—by a mobile forensic solution. Smart devices, including BlackBerry, Windows Mobile, and Windows Phone devices, can hold the same type of data and files that are stored on a personal computer, but the examination of these devices by most forensic solutions typically captures only a small amount of information. In fact, *most mobile forensic solutions currently used in analysis collect and report only 10 percent of the data.*

The traditionally collected mobile device information is referred to in this book as "tip of the iceberg data." This is generally not the fault of the forensic solution, but the examiner may be to blame because he or she either does not understand that additional data can be accessed or does not have the time to look for it. Either way, an enormous amount of data is missed. With such a wealth of information available within both the logical file system and physical partitions, an examiner must be prepared to get his or her hands dirty.

This chapter is not for those looking for the "Easy" button, but for those who are looking for that needle in the haystack or the smoking gun. This chapter will lead the examiner into the file system of the mobile device, into the areas where the "tip of the iceberg data" has been collected, and beyond. If you're averse to getting your hands dirty with a little hexadecimal digging, this chapter, and mobile device forensics, might not be for you.

Avoiding Tool Hashing Inconsistencies

One big reason for analyzing a device's file system is to confirm or deny a mobile forensic solution's reporting of a hash value. Recall that a hash value is generated by running a mathematical algorithm generally upon a binary file. This hash value can be used as the digital fingerprint for the file. If anything changes within the file, the entire hash value will change, indicating a possible problem with an evidentiary file. Hashes in the mobile forensic solution world can be confusing, however, especially when the same data is collected with two separate software solutions and returns two different hash values. If the information and data are the same, shouldn't the hash values be the same no matter what software is used? In fact, the reality is quite the contrary.

When a mobile forensic solution extracts data from a mobile device, the data is parsed from the file and displayed within the user interface. The examiner can run through the various user data types (such as SMS, contacts, call logs, and MMS), previewing the data. The examiner then creates a report based upon the selected user data. Generally, it is within this report that a hash value is assigned for each selected user data type and prominently displayed alongside the collected data. If an examiner uses a different piece of mobile forensic software to collect the same device, select the same user data types, and create a report, there is an almost 100 percent certainty that if the software supports a hash value function, the hash from the user data type will not match the hash returned by the previously used software. This is not because of corruption of the data by the software solution, but it is determined by what the forensic software is actually hashing and subsequently displaying in the report.

Mobile forensic solutions display a hash within the report dialog after hashing data that was extracted from the actual file; this means that the solution is not extracting data from the original file. In other words, if software A collects the contacts from a mobile device, the contacts are placed into a temporary (or permanent) file and hashed. When software B is used, it also extracts the contacts, writes them to a temporary or permanent file, and then hashes that file. Both software A and B extract the data from the contacts file, transfer the data to their respective files intermediately, and then hash that file. Even if the same software solution conducts the collection of the same mobile device twice, back to back, the hash value of the user data types (SMS, MMS, contacts, call logs, and so on) will be different! This is the result of the software creating a file on the analysis machine, filling it with the selected user data, and hashing that file. If only 1 bit is different in the file, which could result from simple file type metadata (such as date, time, file type), the entire hash will be different.

Note

Understandably, the overall image hash can change upon each collection simply because of system file changes on the device, but the hash of the user type files should never change.

The probability that the hash value will be the same for extracted user data from two extractions is extremely low, and this is why the hash that is created by one software solution should never be used to identify the data fingerprint unless the hash is also the file to which the data was parsed. If an examiner understands that the hash value produced in the report is the hash of the created file (such as a .csv, Excel, or text file) by the mobile forensic software and that any change to the file prior to presentation would change the overall value and could indicate tampering, then the hash has evidentiary value. However, if the hash is used to compare the same data and produced a hash from another software solution, or it is used to testify to the fact that the produced hash is of the actual collected file and not simply the extracted file data, problems in presentation and in court could be possible. This, of course, does not pertain to the hash value as reported by the software if the actual physical file has been hashed—only when the hash of the data collected by the software is used.

Hash inconsistencies of reported data in mobile forensic examinations are prevalent with multi-tool collections, and, as such, the examiner should understand that advanced analysis techniques are necessary when multiple tools are used. To circumvent any challenges, the examiner should understand what the hash represents: the file or the extracted data. With this information, the examiner can easily communicate about the method and data the software is representing with the digital fingerprint.

Multiple Tools Often Equal Multiple Hashes

Here's an example of hash inconsistency with a collection conducted on a Motorola V3 device using both Secure View and Cellebrite UFED Classic. This feature phone collection is a simple logical collection, where user data such as SMS, contacts, images, calendar, and call logs are selected for extraction and a report is immediately generated for all the collected data. Both solutions produce HTML reports and display a hash of the captured information above each data type. The hash values for each data type from both software solutions are then compared along with the content. This information is outlined in Table 12-1.

TABLE 12-1 **List of Collected Data and Hashes from Two Mobile Forensic Solutions**

User Data Type	Secure View No. Extracted	Secure View Hash Value	Cellebrite No. Extracted	Cellebrite Hash Value
SMS	11	9f9812c30f011ebb9e506a6b009423d6	Failed	
Contacts	15	7b926b47937770e12610ccb99a7212	15	905151d8dbe2cced352a2d0857c76404
Calendar	4	6779b3ce262c6ccbf8b299379c93dea3	No Support	
Call Logs	16	63987ecdc7bae137fecf6770acb59277	16	C38d019a78108f638c6182b0c41a5d98

Clearly, the number of each extracted data type matches between solutions, and when the content is examined, all content is identical. What is evident, however, is that the hash values shown in the table are not the same. The different hashes, collected with the same data, drive home the point that a solution's representation of the data by a hash can lead to problems when multiple tools are used on a single device. Understanding why this occurs, as outlined in this section, will assist the examiner in explaining this discrepancy. Individual files, such as images and videos, however, are correctly identified by hashes between both solutions because both solutions hashed the actual file, not the temporary report file into which it was embedded.

Iceberg Theory

The "iceberg theory" was actually coined by Ernest Hemingway to describe the "bones" of a great novel. The great writer would give enough information (minimalistic writing style) in the written text to paint a picture that would be filled with the thoughts, and beliefs of the reader, to create a more personal experience. This would mean the reader would "fill in the detail" and become more involved in the novel. This is the format that is used in many motion pictures that are adapted from books to not only adapt the movie to a long book, but often backfire. How often have you pictured a character from a book, only to be disappointed by what they look like in a movie?

Enter the iceberg theory of writing; the analogy to mobile forensics is not as theoretical.

An iceberg floats on water because the density of the ice (around 0.92 g/ml) is greater than that of the sea water (1.03 g/ml). Ice is about 90 percent as dense as water, so only 10 percent of the ice is actually above the water's surface. This means that an iceberg's total mass is 90 percent below the surface of the water! What does an iceberg have to do with mobile device forensics? If we compare the total volume of data on a mobile device versus what a mobile forensic tool actually parses and displays to an examiner, the displayed data is what I call the "tip of the iceberg data."

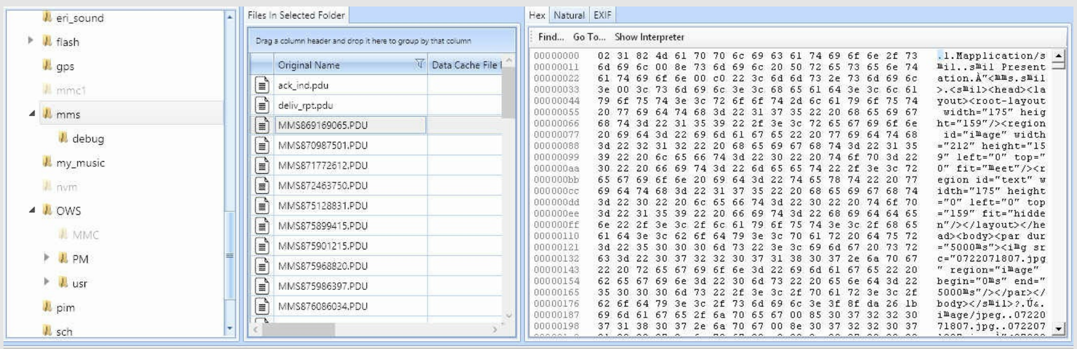
Mobile forensic solutions connect to a mobile device, enable the user to select the items to extract, and then pull the data from the appropriate files. Once complete, the solution will display the collected information and the examiner can then create a report. The information collected is displayed to the examiner in a parsed format (in a report user interface), but it is actually less than 10 percent of the collected data that the examiner would get if the mobile solution were capable of a file system or physical collection. This fact alone leads credence to the "push-button forensic" aura that mobile forensics produces in digital forensic circles and propagates the banter that mobile forensics should instead be called "mobile extraction."

The iceberg theory, as it relates to a mobile forensic examination, is demonstrated in the statistics gathered from an LG VX8300 feature phone and described in the sidebar, "Mobile Device–berg."

Mobile Device–berg

In 2006, the LG VX8300 flip phone was the standard for Verizon cell phones in the United States. It has since been discontinued, but the statistics produced during the testing of the iceberg theory can be seen across all feature phones. First, the LG VX8300 had only 28MB of internal storage, which seems unbelievable when you consider that today's devices can have more than 512GB onboard! This value represented the storage space for both the firmware of the device and any user data—which considering today's device numbers for internal storage is hard to fathom.

The collection of the LG VX8300 produced extracted contacts, call history, SMS, and media—all of which were displayed in the user interface; these capabilities for extraction are generically supported across mobile forensic solutions. The total size of this data was just over 11MB, with most of the space occupied by 61 images and two videos taken with the cell phone. The file system containing the files and folders had a total size of 20MB. The images, videos, and other collected user data occupied roughly 55 percent of the file system and were immediately viewable to the examiner. This was the "tip of the iceberg data," the WYSIWYG data commonly regarded as the only data available on the device—so the mobile forensic software says. However, a closer look at the device file system revealed that there were 366 files within the file system, and 65 of those files were the files that produced the WYSIWYG data; what is important to realize is that another 301 files within the file system could potentially hold data that would be valuable to the investigation. This is the *WYDSIWYN* (what you don't see is what you need) data and the part of the examination that is often neglected. Out of the 366 files within the file system, only 18 percent were interrogated and examined—quite lazy for an automated process. Within this LG VX8300's file system was the other 82 percent—a mixture of system data and user data. The user data in the file system could be critical, as could the actual files the user data was pulled from during the automated process. Consider, for example, the additional 112 image files that were carved from the file system and unaccounted for in the "tip of the iceberg data" the automated tool collected and displayed within the media view. Also, within the file system were the MMS files that were not automatically parsed out by the mobile forensic solution. See Figure 12-1.



Original Name	Data Cache File
ack_ind.pdu	
deliv_rpt.pdu	
MMS869169065.PDU	
MMS870987501.PDU	
MMS871772612.PDU	
MMS872463750.PDU	
MMS875128831.PDU	
MMS875899415.PDU	
MMS875901215.PDU	
MMS875968820.PDU	
MMS875986397.PDU	
MMS876086034.PDU	

```
00000000   02 31 82 4d 61 70 70 6c 69 63 61 74 69 6f 6e 2f 73   .1.Mapplication/s
00000011   6d 69 6c 00 8e 73 6d 69 6c 20 50 72 65 73 65 6e 74   mil..smil Present
00000022   61 74 69 6f 6e 00 c0 22 3c 6d 6d 73 2e 73 6d 69 6c   ation.À"<mms.smil
00000033   3e 00 3c 73 6d 69 6c 3e 3c 68 65 61 64 3e 3c 6c 61   >.<smil><head><la
00000044   79 6f 75 74 3e 3c 72 6f 6f 74 2d 6c 61 79 6f 75 74   yout><root-layout
00000055   20 77 69 64 74 68 3d 22 31 37 35 22 20 68 65 69 67    width="175" heig
00000066   68 74 3d 22 31 35 39 22 2f 3e 3c 72 65 67 69 6f 6e   ht="159"/><region
00000077   20 69 64 3d 22 69 6d 61 67 65 22 20 77 69 64 74 68    id="image" width
00000088   3d 22 32 31 32 22 20 68 65 69 67 68 74 3d 22 31 35   ="212" height="15
00000099   39 22 20 6c 65 66 74 3d 22 30 22 20 74 6f 70 3d 22   9" left="0" top="
000000aa   30 22 20 66 69 74 3d 22 6d 65 65 74 22 2f 3e 3c 72   0" fit="meet"/><r
000000bb   65 67 69 6f 6e 20 69 64 3d 22 74 65 78 74 22 20 77   egion id="text" w
000000cc   69 64 74 68 3d 22 31 37 35 22 20 68 65 69 67 68 74   idth="175" height
000000dd   3d 22 30 22 20 6c 65 66 74 3d 22 30 22 20 74 6f 70   ="0" left="0" top
000000ee   3d 22 31 35 39 22 20 66 69 74 3d 22 68 69 64 64 65   ="159" fit="hidde
000000ff   6e 22 2f 3e 3c 2f 6c 61 79 6f 75 74 3e 3c 2f 68 65   n"/></layout></he
00000110   61 64 3e 3c 62 6f 64 79 3e 3c 70 61 72 20 64 75 72   ad><body><par dur
00000121   3d 22 35 30 30 30 6d 73 22 3e 3c 69 6d 67 20 73 72   ="5000ms"><img sr
00000132   63 3d 22 30 37 32 32 30 37 31 38 30 37 2e 6a 70 67   c="0722071807.jpg
00000143   22 20 72 65 67 69 6f 6e 3d 22 69 6d 61 67 65 22 20   " region="image" 
00000154   62 65 67 69 6e 3d 22 30 6d 73 22 20 65 6e 64 3d 22   begin="0ms" end="
00000165   35 30 30 30 6d 73 22 2f 3e 3c 2f 70 61 72 3e 3c 2f   5000ms"/></par></
00000176   62 6f 64 79 3e 3c 2f 73 6d 69 6c 3e 3f 8f da 26 1b   body></smil>?.Ú&.
00000187   69 6d 61 67 65 2f 6a 70 65 67 00 85 30 37 32 32 30   image/jpeg..07220
00000198   37 31 38 30 37 2e 6a 70 67 00 8e 30 37 32 32 30 37   71807.jpg..072207
```

FIGURE 12-1 **MMS folder containing some of the additional files unaccounted for in a "tip of the iceberg data" examination**

Note

WYSIWYG (*what you see is what you get*) is a term coined back in the 1970s, when onscreen text in a word processing program (Bravo) looked like what would be output to print. *WYDSIWYN* (*what you don't see is what you need*) is a term I used in a talk on mobile forensics in early 2013.

Each examination, whether it is on a feature phone or a smart device, should be performed by the examiner in a way that extends outside of the automated exam in an attempt to uncover some of the valuable information contained below the surface. This is critical to becoming an expert in mobile forensics—living in the 90 percent below the surface of the mobile device–berg.

Feature Phones

As mentioned, feature phones, or legacy phones, are common even in today's smart device world. Feature phones come with stock apps installed with the firmware of the device, while smart devices can be updated to allow the addition of apps from various online repositories without changing the underlying firmware of the devices. This does not change the fact that a wealth of information can be found within a feature phone file system. A feature phone contains an operating system and a file system, where user data is stored and is retrievable if the examiner takes the time to investigate.

Culling through the file system of a feature phone can be tedious work, because it often represents "uncharted territory." In fact, sorting through the file system of the feature phone is like searching through thousands of papers for the one sheet possibly containing the username and password. Because little documentation exists regarding the format of data and file system structures and because of a lack of conformity, a feature phone is often more difficult to examine than a smart device. As a result, some mobile forensic examiners may rely on data that an automated tool will extract and present, neglecting the valuable information available within the file system—but this is where the most important data is nested.

Feature phones will continue to be available to users around the world who value price over functionality, and some simply believe they offer more security. Because of this, examiners should be prepared to examine and analyze feature phones at a level that exceeds the data output by a forensic solution.

Feature Phone "Tip of the Iceberg Data"

User data collected automatically by a forensic tool is often the only information an examiner will present in a case; the examiner relies on only this information for the investigation or examination. *User data*, in the context of this discussion, is data that can be placed into the PIM (personal information manager) category, such as contacts, call logs, SMS, MMS, calendar, and notes. PIM information does not encompass the amount of data that could possibly be stored on a feature phone, but typically it is the only data collected by a mobile forensic solution during many "routine" examinations. These data types are often separated into views for easy navigation and reporting. The examiner should always consult the forensic tool documentation to determine what data can be extracted and parsed.

When a mobile forensic solution presents data, it has already been pulled, parsed, and decoded from a file or files contained within the device's file system. Because the mobile forensic solution then creates a hash of the parsed data, not of the actual file, a thorough examiner should locate the actual file from which the data had been pulled within the device file system and manually examine and hash this data. This hash can then be used to verify a mobile forensic tool's interaction with the file to confirm or deny any allegation that the tool changed the file during extraction and analysis. If the forensic tool does not hash the file in the file system, the examiner can use free tools such as MD5summer (www.md5summer.org) or Bullzip MD5 Calculator (www.bullzip.com/products/md5/info.php) to accomplish the hashing. The hash should then be documented in the examiner's report.

In the following sections, different types of formats for date and time, along with data layouts, will be discussed. Using this information, the examiner can decode files that contain the user data parsed with automated tools to help verify and validate what the automated tool is providing.

Note

Not all file systems are accessible on feature phones. These methods apply only to devices from which a file system can be extracted.

Parsing a Feature Phone File System

One of the most important features of any mobile forensic solution is the ability to acquire a feature phone's file system. Even a simple CDMA LG device with only 28MB of storage space can provide a wealth of information should the examiner look into the file system outside of what an automated tool can pull.

Unfortunately, one of the most difficult and time-consuming jobs for an examiner is looking through each and every file and folder for information that might pertain to the case or investigation. There are many different embedded file systems (EFS) formats, and a lot of them are proprietary and specific to manufacturer and carrier. However, CDMA feature phones often contain a similar directory structure because the internal components conform to the Binary Runtime Environment for Wireless (BREW) mobile platform (MP), which is the most widely used application platform in CDMA devices. The BREW MP EFS is documented on the BREW web site (https://developer.brewmp.com/resources/tech-guides/file-system-technology-guide/high-level-architecture/brew-mp-file-system) and outlines the format for the major structures within the EFS. Because Qualcomm chips can be used in both CDMA and GSM devices, understanding the major structures of the BREW MP and what data can be located in these areas can be important in identifying where to look for valuable data.

By comparing the LG file system in Figure 12-2 and the UTStarcom file system in Figure 12-3, you can see that both conform to the BREW MP, and important folders can be readily observed. Accessing the CDMA feature phone EFS is much easier than accessing a GSM device EFS, because BREW is an open source format developed by Qualcomm. Because of the open API, developers often can obtain file system–level access using the available documentation. GSM feature phones, however, are not as open.

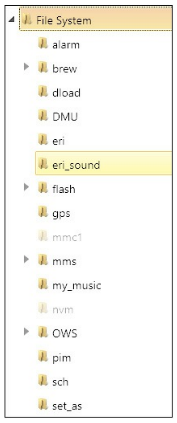


FIGURE 12-2 **The LG VX8300 file system contains many files and folders.**

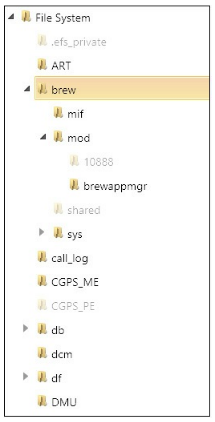
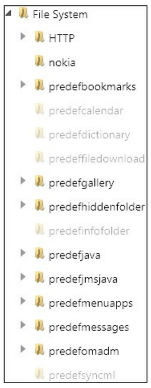


FIGURE 12-3 **The UTStarcom feature phone file system also contains many files and folders.**

Access to the GSM feature phone EFS is often accomplished only with non-invasive physical access; however, some GSM devices from Nokia allow for file system dumping via common protocols. Figure 12-4 shows a Nokia GSM file system that was collected from the device and was not a decoded physical binary file. If the GSM feature phone's file system cannot be accessed without a non-invasive physical or an invasive collection, the device should be "dumped" by mobile forensic tools (such as Cellebrite Touch 2 or MSAB

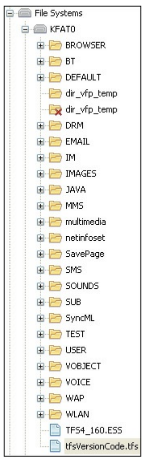
XRY). Once collected, the binary file can then be decoded. The decoding process rebuilds the file system that is specific to the mobile device "family," the NOR/NAND chip, and underlying EFS, as shown in Figure 12-5, with a Cellebrite decoding of a Samsung non-invasive physical collection.



```
File System
  HTTP
  nokia
  predefbookmarks
  predefcalendar
  predefdictionary
  predeffiledownload
  predefgallery
  predefhiddenfolder
  predefinfofolder
  predefjava
  predefjmsjava
  predefmenuapps
  predefmessages
  predefomadm
  predefsyncml
```

FIGURE 12-4 **Nokia GSM feature phone file system**

EFS could be versions of TFS (Transactional File System) for Samsung, Paragon2000 (P2K) and Paragon2005 (P2K05) for Motorola, Qualcomm for LG, or Nokia's proprietary format. (Throughout this book, all of these will be referred to as "EFS" because it is not the intention of this book to rebuild the file system; we're simply navigating and decoding various artifacts.) In the file system example shown in Figure 12-5, the EFS clearly shows an MMS folder with available data, but this was not automatically parsed by any tool. The examiner would now be able to parse manually and locate valuable information once thought nonexistent because it was not parsed by the mobile forensic tool.



```
File Systems
  KFAT0
    BROWSER
    BT
    DEFAULT
    dir_vfp_temp
    dir_vfp_temp
    DRM
    EMAIL
    IM
    IMAGES
    JAVA
    MMS
    multimedia
    netinfoset
    SavePage
    SMS
    SOUNDS
    SUB
    SyncML
    TEST
    USER
    VOBJECT
    VOICE
    WAP
    WLAN
    TFS4_160.ESS
    tfsVersionCode.tfs
```

FIGURE 12-5 **File system from a binary file obtained with a non-invasive EFS extraction of a Samsung feature phone**

Because of the many different formats of data and file systems, it is difficult for any mobile forensic solution to uncover all data important to an investigation. That is where the investigation into the device file system, especially with the feature phone, is most beneficial. The key is to obtain the EFS, if possible, from any feature phone that is to be examined.

Common Layout

Although no specific type of file system is always used within a feature phone, after the data is decoded and a file system is visible within the forensic solution, the examiner can find data in some common locations. Looking into these areas can be critical to obtaining additional information that possibly was not parsed automatically by the tool—or it can help to verify that the tool correctly parsed the file. Table 12-2 lists various folders and files that store information about the user or device that could assist in investigations. Many additional files can also be investigated, and those listed are files I've used in various investigations. Some of the files will be covered in greater detail in following sections.

TABLE 12-2 **Various Locations of Critical Data in the EFS of Feature Phones**

Type of Device	Manufacturer	Path	File	Artifact
GSM	Samsung	/Browser/History	.dat files	Browser data (NFPKDDAT)
GSM	Samsung	/Browser/Cache	.dcf files	Web cache files
GSM	Samsung	/Email	Draft/Inbox/Sent	Browser data (NFPKDDAT)
GSM	Samsung	/MMS	SMIL files	MMS messages
GSM/CDMA	Nokia	/predefbookmarks/bookmarks	.bak files	User bookmarks
GSM/CDMA	Nokia	/predefgallery/predefphotos	.jpg files	Phone camera storage
GSM/CDMA	Nokia	/predefs40browser	LastURL.bmk	Last visited URL
GSM/CDMA	Nokia	/predefmessages	SMS and MMS	SMS/MMS messages
CDMA	AudioVox	/camera	CameraInfo.dat	Picture names if changed by user
CDMA	AudioVox	/camera	nextphoto.int	Integer of next photo; good to show if one was deleted
CDMA	AudioVox	/db	default_snd.txt	User auto messages
CDMA	AudioVox	/memo	memo_000.pad	User memos
CDMA	AudioVox	/MMS	Various	MMS messages

CDMA	All CDMA devices	/nvm	$SYS.ESN	Electronic serial number
CDMA	AudioVox	/nvm/ch	call_history	Call history file
CDMA	All CDMA devices	/nvm/nvm	nvm_0000	Device serial number, carrier name
CDMA	All CDMA devices	/nvm/nvm	nvm_0002, nvm_minlock, nvm_0000, nvm_002	Name, device lock, subsidy lock
CDMA	AudioVox	/nvm/phbk	phonebook	Device phonebook
CDMA	AudioVox, Samsung, others	/nvm/sms	SMS	SMS messages
CDMA	AudioVox	/nvm/smssaved	SMS	Saved SMS messages
CDMA	AudioVox	/VoiceDB/Memos	.qcp files	Voice memo files
CDMA	UTStarcom	/db/pb_db	pb_list_table	Copy of contacts
CDMA	UTStarcom	/db/sms_db	sms_inbox_table	Copy of SMS even if SMS deleted
CDMA	LG	/brew/16452/mp	.jpg files	Camera location
CDMA	LG, others	/brew/16452/mf	.3g2 files	Camera location
CDMA	LG, others	/brew/16452/ms	.qcp, .mid files	User recordings
CDMA	LG, Motorola, others	/brew/mod/4972	Various files db_X	X = Instant Messenger (AOL, MSN, Yahoo!) contacts
CDMA	LG, Motorola, others	/brew/mod/4972	Various files db_X_msg	X = Instant Messenger; stores messages for IM account
CDMA	LG	/pim	Various files	Call history, phonebook
CDMA	Most CDMA	/t9udb	T9udb_eng.dat, T9udb_spa.dat	Dictionary words entered by user for SMS and MMS

CDMA/GSM	Motorola	/motorola/browser/usr	various	Browser data
CDMA	Motorola	/brew/mod/syncom2	Folders i (inbox), o (outbox), s (saved), d (drafts); each has index folder with mb.idx	Basic SMS/MMS text
CDMA	Motorola	/brew/mod/syncom2	Each folder (d, o, i, s) has .att and .env files	Pointers to message and attachments
CDMA	Motorola	/motorola/shared/	Image and video files	Camera location
CDMA	Motorola	/nvm	nvm_security	Device lock
CDMA	Samsung	/cam	Various folders and files	Camera location
CDMA	Samsung	/nvm	dial	Phonebook
CDMA	Samsung	/nvm	dial_tbl	Call history file
CDMA	Samsung	/nvm	schedule	Calendar
CDMA	Samsung	/nvm	T9udb	User dictionary
CDMA	Samsung	/vmemo	.vm files	Voice memo files

Tip

Remember that the mobile forensic solution must be able to recover the EFS in order to use the table.

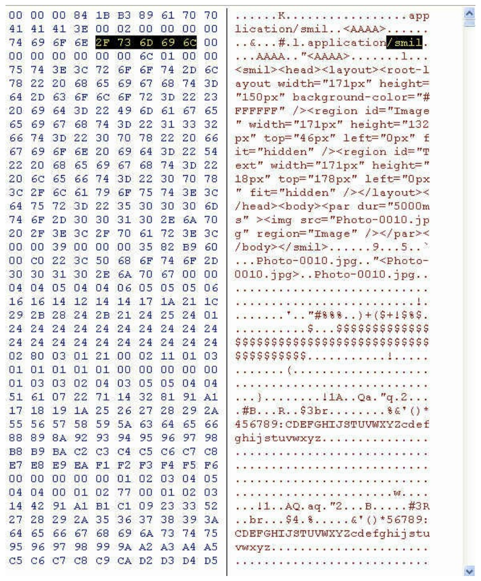
File Metadata Cues

Because of the variety of manufacturers and models, all the different files in a feature phone cannot be represented in a single table—unless the table were hundreds of pages long. Some feature phones include more than 4000 different files, most of which are system files, but a tremendous number of these files can hold the “smoking gun.” The examiner could search manually through many of the files, but a better approach is to use search terms to locate files that might not be represented by the parsed data extracted by the mobile forensic solution.

To be successful at locating data within the plethora of files in the feature phone file system, the examiner must first understand how to search the files. By using standard text strings and regular expressions, the examiner will be able to locate and examine additional files within the file system (see Figure 12-6, for example). The examiner will not only benefit from the data recovered, but he or she will gain a better understanding regarding how the mobile forensic solution navigates the feature phone file system. Table 12-3 represents some common search strings that can be used to locate various items within a feature phone file system.

TABLE 12-3 **Possible Search Terms for Various Artifacts on Feature Phones**

Artifact	Search Terms	Regular Expression
MMS	<smil	
MMS	<applicationsmil_1.smil>	
MMS text content	filename=textplain	
SMS Samsung	DEADBEEF	
CDMA lock codes		\x01\d\d\d\d\x01
IP addresses		\b\d{1,3}\.\d{1,3} \.\d{1,3}\.\d{1,3}\b
Voice memos	. qcp, .amr, .acc, .vm, RIFF	
Videos	.3gp2, .3gp	
Images	.jpg	
Settings files		<?xml version=



```
00 00 00 84 1B B3 89 61 70 70   ......K.................app
41 41 41 3E 00 02 00 00 00 00   lication/smil..<AAAA>......
74 69 6F 6E 2F 73 6D 69 6C 00   ..&...#.1.application/smil.
00 00 00 00 00 00 6C 01 00 00   ...AAAA.."<AAAA>.......l...
75 74 3E 3C 72 6F 6F 74 2D 6C   <smil><head><layout><root-l
78 22 20 68 65 69 67 68 74 3D   ayout width="171px" height=
64 2D 63 6F 6C 6F 72 3D 22 23   "150px" background-color="#
20 69 64 3D 22 49 6D 61 67 65   FFFFFF" /><region id="Image
65 69 67 68 74 3D 22 31 33 32   " width="171px" height="132
66 74 3D 22 30 70 78 22 20 66   px" top="46px" left="0px" f
67 69 6F 6E 20 69 64 3D 22 54   it="hidden" /><region id="T
22 20 68 65 69 67 68 74 3D 22   ext" width="171px" height="
20 6C 65 66 74 3D 22 30 70 78   18px" top="178px" left="0px
3C 2F 6C 61 79 6F 75 74 3E 3C   " fit="hidden" /></layout><
64 75 72 3D 22 35 30 30 30 6D   /head><body><par dur="5000m
74 6F 2D 30 30 31 30 2E 6A 70   s" ><img src="Photo-0010.jp
20 2F 3E 3C 2F 70 61 72 3E 3C   g" region="Image" /></par><
00 00 39 00 00 00 35 82 B9 60   /body></smil>......9...5..`
00 C0 22 3C 50 68 6F 74 6F 2D   ...Photo-0010.jpg.."<Photo-
30 30 31 30 2E 6A 70 67 00 00   0010.jpg>..Photo-0010.jpg..
04 04 05 04 04 06 05 05 05 06   ...........................
16 16 14 12 14 14 17 1A 21 1C   .........................!.
29 2B 28 24 2B 21 24 25 24 01   ......."..."#%%%..)+($+!$%$.
24 24 24 24 24 24 24 24 24 24   ..........$...$$$$$$$$$$$$$
24 24 24 24 24 24 24 24 24 24   $$$$$$$$$$$$$$$$$$$$$$$$$$$
02 80 03 01 21 00 02 11 01 03   $$$$$$$$$$...........!.....
01 01 01 01 01 00 00 00 00 00   .......(...................
01 03 03 02 04 03 05 05 04 04   ...........................
51 61 07 22 71 14 32 81 91 A1   ...}........!1A..Qa."q.2...
17 18 19 1A 25 26 27 28 29 2A   .#B...R..$3br........%&'()*
55 56 57 58 59 5A 63 64 65 66   456789:CDEFGHIJSTUVWXYZcdef
88 89 8A 92 93 94 95 96 97 98   ghijstuvwxyz...............
B8 B9 BA C2 C3 C4 C5 C6 C7 C8   ...........................
E7 E8 E9 EA F1 F2 F3 F4 F5 F6   ...........................
00 00 00 00 00 01 02 03 04 05   ...........................
04 04 00 01 02 77 00 01 02 03   ......................w....
14 42 91 A1 B1 C1 09 23 33 52   ...!1..AQ.aq."2...B.....#3R
27 28 29 2A 35 36 37 38 39 3A   ..br...$4.%.....&'()*56789:
64 65 66 67 68 69 6A 73 74 75   CDEFGHIJSTUVWXYZcdefghijstu
95 96 97 98 99 9A A2 A3 A4 A5   vwxyz......................
C5 C6 C7 C8 C9 CA D2 D3 D4 D5   ...........................
```

FIGURE 12-6 **Using Physical Analyzer to search manually for an MMS message; notice the searched-for string "smil".**

Dates and Times

Dates and times are important for most investigations, and sometimes dates need to be either verified or located. The problem that arises with dates and times within a feature phone are the many different formats available. Depending upon the device, the manufacturer, and even the version of the EFS, the date and time format could be different. Table 12-4 lists the most common date and time types for both GSM and CDMA feature phones along with the epoch that is referenced.

TABLE 12-4 **Common Epoch Dates**

Format	Epoch
UNIX Time	January 1, 1970
Qualcomm BREW/GPS	January 6, 1980
AOL, DOS, FAT12/16/32, exFAT	January 1, 1980

Note

The *epoch* is the reference point from which a date and time will start to display.

After a record is located and the examiner manually decodes a date and time, the examiner can look to the table and decide the format. After doing this repeatedly, the examiner will find that decoding becomes easier and will recognize a pattern. Feature phones generally follow a pattern, with the date and time at an offset either from the start of the file or from some point within the combined string if multiple records are in a single file. Table 12-5 lists the various date and time formats used by both GSM and CDMA feature phones based upon manufacturer. Most of the date and time formats, along with the epoch, are self-explanatory. The "AOL format," as coined by forensic expert Paul Sanderson, is represented by 4 bytes, much like UNIX. Various tools such as DCode and MFI HEX Assistant can help the examiner decode these dates independent of the forensic solution if an embedded HEX interpreter is not included.

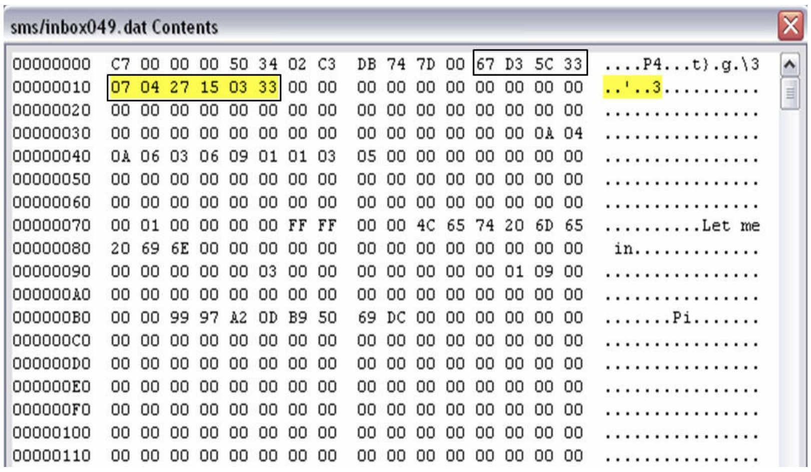
TABLE 12-5 **Various Date and Time Examples from Feature Phones**

Format	Example	Decoded
UNIX time	`2E 1A D9 44 (LE)`	Aug 8, 2006 23:11
BCD (Binary Coded Decimal)	`05 02 19 18 50 26 (YY MM DD HH MM SS)`	Feb 19, 2005 18:50:26
Qualcomm BREW/GPS	`67 D3 5C 33 (LE)`	Apr 27, 2007 15:04
AOL	`2B F0 D6 36 (LE)`	Feb 25, 2009 19:04
GSM (BCD reverse nibble)	`01 80 71 81 05 30 8A`	Aug 17, 2010 18:50:03 (GMT -7)
LG/Samsung	`9C 35 6B 7D (LE)`	Nov 6, 2006 22:28

The LG/Samsung format needs a little more explanation, because the 4-byte values must be converted to a binary number and separated into year, month, day, hour, and minutes:

```
9C 35 6B 7D (LE) converted to binary = 011111010110 1011 00110 10110 011100
                                       \__________/ \__/ \___/ \___/ \____/
                                          Y(2006)   M(11) D(6) H(22)  M(28)
```

Figure 12-7 shows an actual SMS within an LG CDMA phone inbox.dat file with two dates and times outlined. In this particular example, the date and time for the SMS is in Qualcomm BREW format (April 27, 2007 15:04) and BCD format (April 27, 2007 15:03:33), and for this type of file, the date and time are always 12 bytes from the start of the file using 0-based numbering. Once a format and the offsets are established, parsers can be built to locate and interpret many of these date and time formats. Cellebrite Physical Analyzer, as well as FTK and Oxygen Forensic Detective, enable the user to search for various formats within the binary files, often converting these formats on the fly. However, the examiner should be aware of the various formats that can be used within a feature phone's file system.

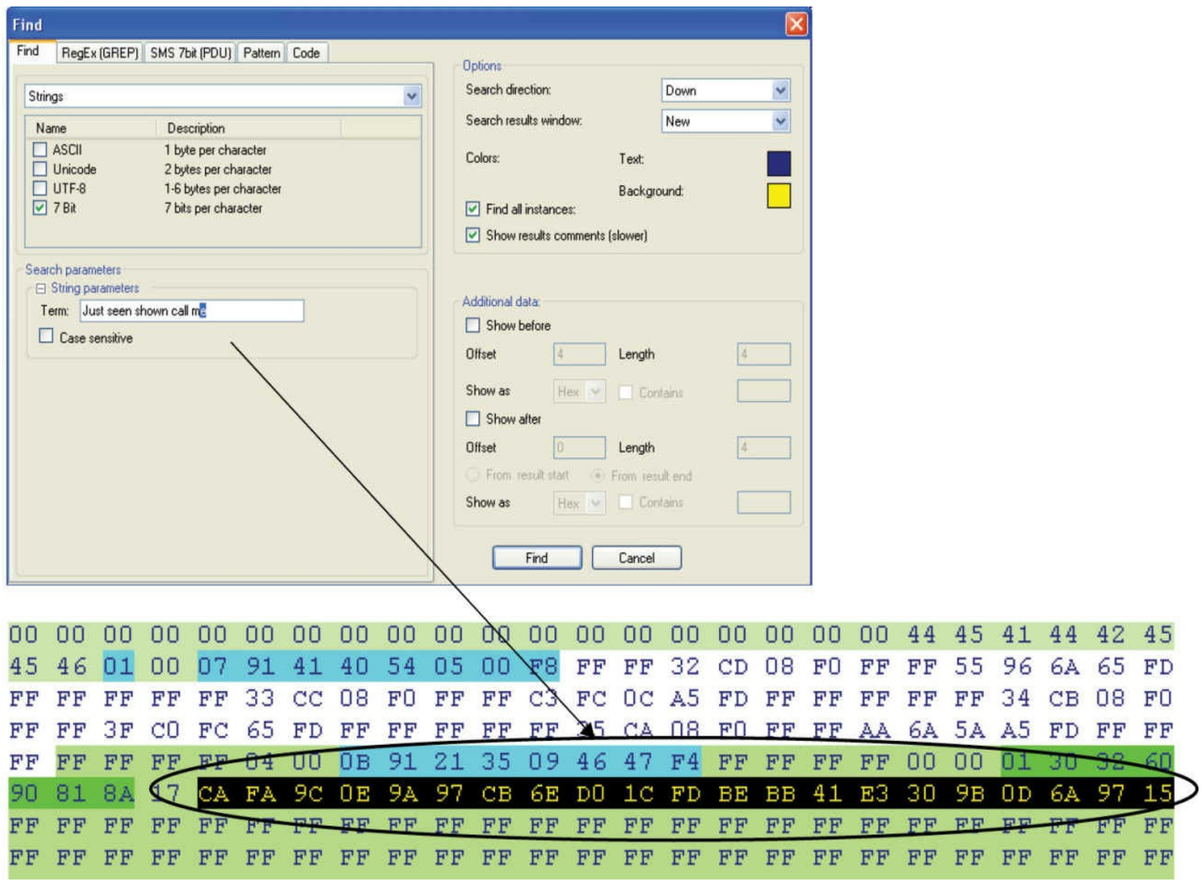


sms/inbox049.dat Contents

```
00000000  C7 00 00 00 50 34 02 C3  DB 74 7D 00 67 D3 5C 33  ....P4...t}.g.\3
00000010  07 04 27 15 03 33 00 00  00 00 00 00 00 00 00 00  ..'..3..........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 0A 04  ................
00000040  0A 06 03 06 09 01 01 03  05 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070  00 01 00 00 00 00 FF FF  00 00 4C 65 74 20 6D 65  ..........Let me
00000080  20 69 6E 00 00 00 00 00  00 00 00 00 00 00 00 00   in.............
00000090  00 00 00 00 00 03 00 00  00 00 00 00 00 01 09 00  ................
000000A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0  00 00 99 97 A2 0D B9 50  69 DC 00 00 00 00 00 00  .......Pi.......
000000C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

FIGURE 12-7 **An LG file system file from an inbox file; notice two dates and times within a single file, which is indicative of sent and received times for many feature phone files.**

Armed with this information, the examiner can not only verify that the automated tool parsed the data correctly, but when information is located that was not automatically parsed, the examiner can manually decode and present the data. Cellebrite Physical Analyzer decodes many different types of data automatically for the user (see Figure 12-8). However, like all other mobile forensic tools, not all items are located and decoded automatically. Using either Physical Analyzer or Oxygen Forensic Detective, the examiner can highlight the represented date and time and properly convert that information using a built-in converter. This data can then be bookmarked and recorded for review.


Find

Find | RegEx (GREP) | SMS 7bit (PDU) | Pattern | Code

Strings

Name	Description
☐ ASCII	1 byte per character
☐ Unicode	2 bytes per character
☐ UTF-8	1-6 bytes per character
☑ 7 Bit	7 bits per character

Search parameters
String parameters
Term: Just seen shown call me
☐ Case sensitive

Options
Search direction: Down
Search results window: New
Colors: Text: 🟦
Background: 🟨
☑ Find all instances:
☑ Show results comments (slower)

Additional data:
☐ Show before
Offset 4 Length 4
Show as Hex ☐ Contains
☐ Show after
Offset 0 Length 4
From result start ● From result end
Show as Hex ☐ Contains

Find Cancel

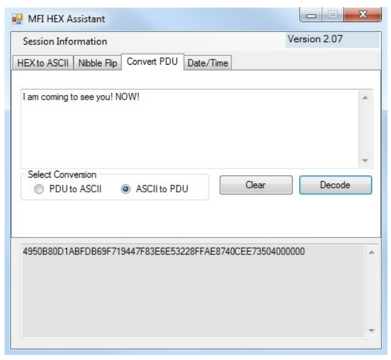
```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 44 45 41 44 42 45
45 46 01 00 07 91 41 40 54 05 00 F8 FF FF 32 CD 08 F0 FF FF 55 96 6A 65 FD
FF FF FF FF FF 33 CC 08 F0 FF FF C3 FC 0C A5 FD FF FF FF FF FF 34 CB 08 F0
FF FF 3F C0 FC 65 FD FF FF FF FF FF 35 CA 08 F0 FF FF AA 6A 5A A5 FD FF FF
FF FF FF FF FF 04 00 0B 91 21 35 09 46 47 F4 FF FF FF FF 00 00 01 30 92 60
90 81 8A 17 CA FA 9C 0E 9A 97 CB 6E D0 1C FD BE BB 41 E3 30 9B 0D 6A 97 15
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

FIGURE 12-8 **Using Cellebrite Physical Analyzer, the examiner can search for a string in several formats.**

Text Formats

Feature phone mobile data can be difficult to decode. The use of the 7-bit GSM alphabet or UCS-2 (2-byte Universal Character Set, often referred to as PDU, or Protocol Data Unit), use of Unicode, and formatting of phone numbers can cause confusion while deciphering data from feature phones. This also means that data in the mobile device file system cannot always be searched by using a string search.

For SMS messages in the PDU format, the data could be encoded in hexadecimal values, so the examiner must search using a converted string. The string can be converted with a tool such as MFI HEX Assistant, or the mobile forensic solution can automatically convert the string if the feature is supported. Figure 12-9 shows 7-bit data located from a search using MFI HEX Assistant.


MFI HEX Assistant

Session Information Version 2.07

HEX to ASCII | Nibble Flip | Convert PDU | Date/Time

I am coming to see you! NOW!

Select Conversion
PDU to ASCII [✓ ASCII to PDU]

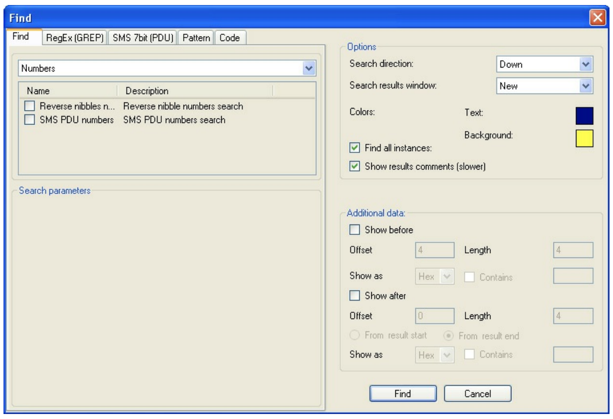
Clear Decode

```
4950B80D1ABFDB69F719447F83E6E53228FFAE8740CEE73504000000
```

FIGURE 12-9 **The search term is entered and 7-bit is selected, which yields a result within the HEX viewer.**

Another problem often encountered occurs when search strings are encoded as Unicode. Some mobile devices store the data as ASCII but in Unicode, meaning 2 bytes for every character. This means, then, that a standard string search for "dog" would not return a result simply because in the file system, the value is "d.o.g." Because "dog" has a secondary value, here represented by the dot, but actually x00, the search misses the characters. Tools such as Physical Analyzer and FTK enable the user to specify that the search should be in text and Unicode format, which would successfully locate the example term "dog."

For phone numbers, the examiner must remember that numbers can be in reverse nibble (see Figure 12-10) or BCD format. This is also referred to as the "GSM phone number format" by several mobile forensic tools. Tools such as Physical Analyzer enable the examiner to specify the format (GSM phone number) of the data to be searched when entering the phone number into the search box. The number can then be searched in the various formats simultaneously, yielding better results.



Numbers

Name	Description
☐ Reverse nibbles n...	Reverse nibble numbers search
☐ SMS PDU numbers	SMS PDU numbers search

Search parameters

Options
Search direction: Down
Search results window: New
Colors: Text: ■ Background: 🟨
☑ Find all instances:
☑ Show results comments (slower)

Additional data:
☐ Show before
Offset 4 Length 4
Show as Hex ☐ Contains
☐ Show after
Offset 0 Length 4
○ From result start ● From result end
Show as Hex ☐ Contains

FIGURE 12-10 **Numbers can be stored differently, so searching for numbers in various formats is important.**

Understanding that text-based data within a feature phone is stored in various formats will help the examiner achieve better success when conducting a search for specific communication. The examiner will be able to convert the string into a format to search both PDU (7-bit) alphabet and standard ASCII strings.

Deleted Data

Deleted data is often the most sought-after artifact within a feature phone file system. Deleted data can be located only if the EFS has been extracted. If only a surface ("tip of the data iceberg") collection has occurred, the examiner will be unable to conduct a search for deleted file artifacts.

Note

Feature phone deleted data is typically located outside of the formatted file system, and searches must be conducted on the formatted NAND/NOR binary file. If only a file system is available, the recovery of deleted artifacts on a CDMA or GSM feature phone is unlikely. There are some exceptions, however, such as the Motorola V3 CDMA series, in which the msging folder can store deleted messages, so searching should always be attempted.

BlackBerry Devices

In most situations, forensic examination of a BlackBerry device will involve the examination of a backup produced by either the forensic software or the associated BlackBerry desktop software.

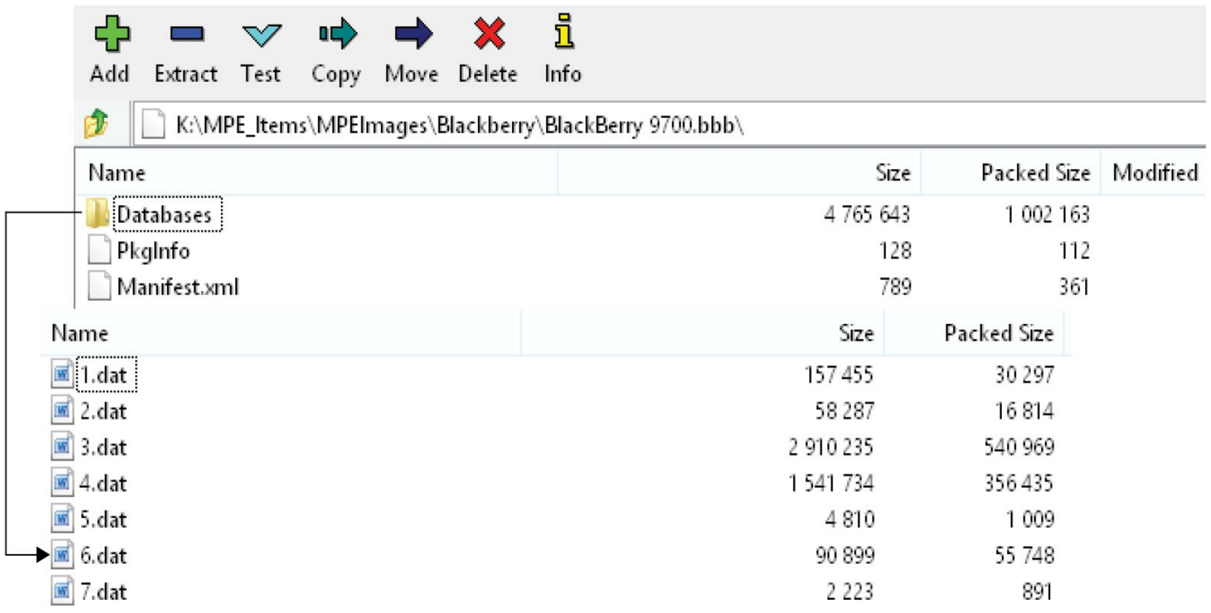
Note

With the BlackBerry market share and minimal exposure in the forensic examiner case load recently, the information in this chapter is presented as a historical education.

If using a mobile forensic solution, the examiner should recognize that many files within a BlackBerry backup are not automatically parsed by the solution. Mobile forensic tools will import the backup file and parse out the relevant database files into the user interface, often ignoring other files, either because they are not supported or because the data could not be decoded. At this point, the examiner should be competent enough to dive into some of the additional files that are located within the backup file, if needed.

A pre-BlackBerry 10 (BB10) device produces a set of database files contained within a single IPD or BBB file, and BB10 devices produce only a BBB file. The difference between the BBB files produced with the BlackBerry desktop software and those of the new BlackBerry Link software is encryption. BlackBerry Link software backups are fully encrypted with credentials independent of any device password. The IPD or BBB file should not be interpreted as the EFS as it existed exactly on the NAND flash within the BlackBerry. If the examiner is assessing the data at a physical level, the information found within the logical backup file should be correlated to what is seen at the flash level. In making this comparison, the examiner can use the logical backup as an aid in locating, and sometimes decoding, the information within the physical binary image.

BlackBerry desktop software prior to version 7.1x produced IPD files for Windows-created backups, and BBB files were produced in Mac backups. With version 7.1x, only BBB files were produced for both Windows and Mac. The difference between the IPD and BBB files involves how the files are constructed. An IPD file is a single file containing numerous tables, representing a resource field and its associated data. The first version of BBB used by the Mac was a compressed zip file that contained an IPD file. Windows, starting with version 7.1x, used BBB files when creating backups. The BBB file was still a compressed file, but the internal resources were now independent files with a .dat extension, a PkgInfo file, and a manifest file (see Figure 12-11). The PkgInfo file contained the version of the BlackBerry Backup Format, and the manifest file contained a listing of all .dat files in their corresponding resource files (see Figure 12-12).

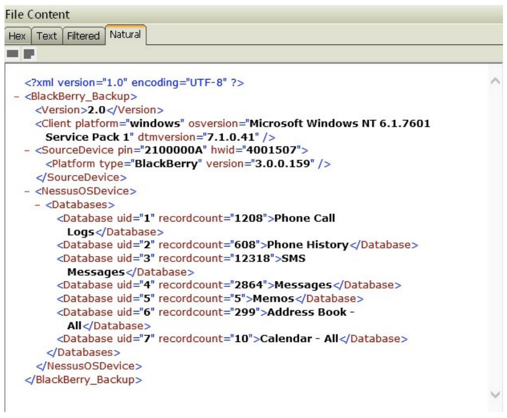


K:\MPE_Items\MPEImages\Blackberry\BlackBerry 9700.bbb\

Name	Size	Packed Size	Modified
Databases	4 765 643	1 002 163	
PkgInfo	128	112	
Manifest.xml	789	361	

Name	Size	Packed Size
1.dat	157 455	30 297
2.dat	58 287	16 814
3.dat	2 910 235	540 969
4.dat	1 541 734	356 435
5.dat	4 810	1 009
6.dat	90 899	55 748
7.dat	2 223	891

FIGURE 12-11 **The first BBB file that existed outside of a Mac backup included a Databases folder with .dat files containing the user databases.**



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <BlackBerry_Backup>
    <Version>2.0</Version>
    <Client platform="windows" osversion="Microsoft Windows NT 6.1.7601
      Service Pack 1" dtmversion="7.1.0.41" />
  - <SourceDevice pin="2100000A" hwid="4001507">
      <Platform type="BlackBerry" version="3.0.0.159" />
    </SourceDevice>
  - <NessusOSDevice>
    - <Databases>
        <Database uid="1" recordcount="1208">Phone Call
          Logs</Database>
        <Database uid="2" recordcount="608">Phone History</Database>
        <Database uid="3" recordcount="12318">SMS
          Messages</Database>
        <Database uid="4" recordcount="2864">Messages</Database>
        <Database uid="5" recordcount="5">Memos</Database>
        <Database uid="6" recordcount="299">Address Book -
          All</Database>
        <Database uid="7" recordcount="10">Calendar - All</Database>
      </Databases>
    </NessusOSDevice>
  </BlackBerry_Backup>
```

FIGURE 12-12 **The manifest file contained the number of records along with the type of data within the .dat file.**

BlackBerry devices can also contain a microSD card that includes data such as application files, documents, media, and chat logs.

BlackBerry "Tip of the Iceberg Data"

Automated logical parsing of a BlackBerry device prior to BB10 typically recovers more information than is recovered from feature phones, including e-mail. Following is a list of the "tip of the iceberg" data recovered from pre-BB10 devices:

Contacts	Calendar	Tasks
Notes	Call log	SMS
MMS	E-mail	Media
Browser data	PIN messages	Searches
Corrected text	Map locations	Hot lists

Note

Oxygen Forensics and Cellebrite documentation suggest including social media and non-BBM chat application support as typical collection items.

This information is a subset of the available information that exists within the IPD or BBB file produced by a pre-BB10 device. Additional recoverable data is available within the BlackBerry database, but the heavy lifting required to uncover this material often makes it impractical to do so. In general, the material located in the additional databases is deemed of minimal value considering the hours it would take to look through all the records.

Note

The BlackBerry device first brought e-mail to the mobile device.

BlackBerry Database Breakdown

The standard IPD file contains many records that correspond to the IPD and the first edition BBB file from a Mac backup. The BBB files produced with a Mac are compressed files containing an IPD file (Databases.ipd) and, if the media card is part of the backup, a second file called emmc.zip. Table 12-6 represents the information as detailed by BlackBerry in its Knowledge Base article KB03974 (http://support.blackberry.com/kb/articleDetail?ArticleNumber=000003974). Using this list of databases located within the IPD and BBB, the examiner can manually investigate some of the databases that have been automatically parsed and also those that have not.

TABLE 12-6 **Typical Database Files Located Within a BlackBerry IPD Backup**

Database	Functionality
Address Book - All	Reflects the total number of contact list entries for multiple contact lists
Address Book - Last Used Hints	Provides information on what phone number or e-mail address was last used to contact a specific contact
Address Book Options	Stores contact list settings
Alarm Options	Stores settings for the alarm application
Alarms	Stores customizations for alarm sounds set by the user
Application Permissions	Stores permissions configured under Options | Security Options | Application Permissions
Attachment Data	Stores attachments that have been viewed on the device
Attachment Data - Calendar	Stores attachments that have been viewed through the calendar application on the device
Attachment Options	Stores user-defined settings for the Attachment Viewer
AutoText/Word Substitution	Contains dictionary for automatic substitution of commonly misspelled words
AutoText Data Version	Contains version of Auto Text Data application
BBGroups	Stores BlackBerry Groups created within the BBM
BIS Account Data	Stores BlackBerry Internet Service account information that might be used by the Thick Client application in the BlackBerry Internet Service
BlackBerry Messenger	Stores contacts in BBM
Bluetooth Options	Stores settings for Bluetooth technology
Browser Bookmarks	Contains bookmarks for web pages in the browser applications
Browser Channels	Shows browser push service subscribed to by the user, such as weather setup service; lists channels browser is subscribed to, such as weather updates
Browser Data Cache	Holds cached data from the browser
Browser Folders	Enables user to sort bookmarks into various folders
Browser Messages	Contains browser push service notifications
Browser Options	Contains browser setting information
Browser Push Options	Contains push content settings for the BlackBerry browser
Browser Web Address	Contains list of the web addresses that have been typed into the browser

Calendar - All	Contains total number of entries for all calendars (multiple CICAL service records)
Calendar Options	Stores calendar settings
Camera Options	Contains database of settings customization in Options | Camera | Camera Options
Categories	Enables various addresses, appointments, tasks, and memos to be sorted by category
Certificate Options	Contains option list in certificates, whether a user set a certificate to Distrust or Revoke or leaves the default setting when no changes are made
Certificate Summary Data	Contains list of certificates stored on the device (Options | Security Options | Advanced Security Options | Certificates)
Clock Options	Contains customization settings or clock options via Clock | Options
Code Module Group Properties	Stores system software properties for the core system modules
Compatibility Settings	Contains compatibility mode settings used when the application has been designed for a physical input (non–touch screen) device, where the application expects to receive input events from the keyboard and the trackball
Configuration Channel	Stores configuration information that an administrator may assign using the Configuration Channel feature of BlackBerry Enterprise Server 5.0
Content Store	Stores pictures and other media content in the on-board device memory
CustomWordsCollection	Stores customized words saved to the custom word dictionary
Default Service Selector	Defines the default message service
Device Options	Stores setting changes made within Options on the device; some options have their own database
Diagnostic App Options	Contains test configuration settings in Self Test Application, initiated in Options | Status using the key sequence TEST
DocsToGoCommonPrefs	Contains setting information for the Documents to Go for device application suite
Email Filters - All	Stores settings for filters applied to incoming messages
Email Settings - All	Defines the way messages are sent and received

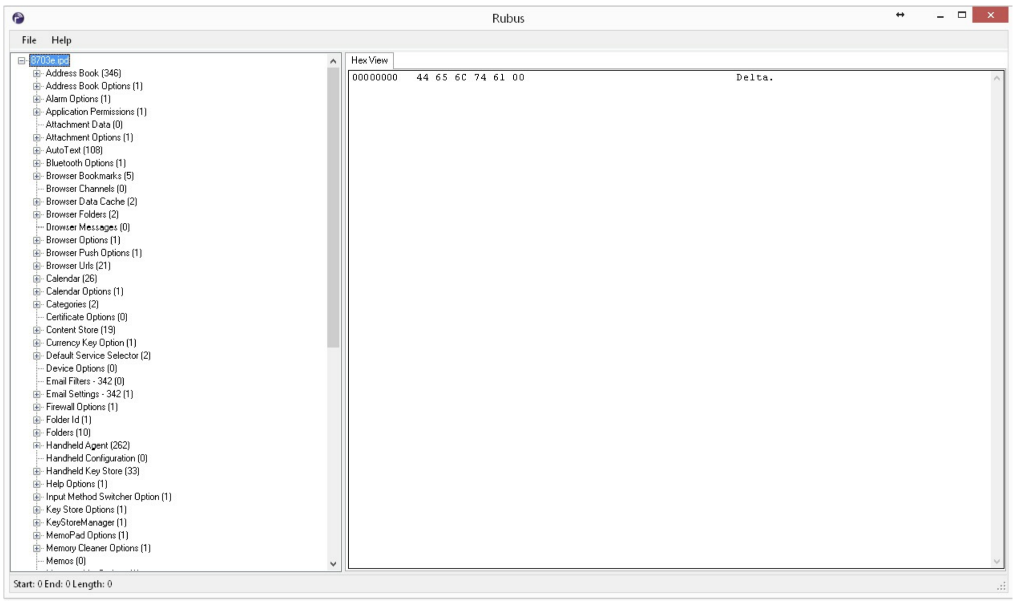
Enhanced Gmail Plug-in	Stores information used by the Enhanced Gmail plug-in
File Explorer Network Favorites	Stores list of file shares that have been marked as Favorites
File Explorer Network History	Stores list of past network file shares that have been accessed from the device
File Explorer Options	Includes options configured in Explorer, such as Show Hidden, or any new folders created
Firewall Options	Enables or disables the firewall
Folder ID	Determines which folder a message is filed into
Folders	Lists user-created folders in the message list
Full-predictive Options	Stores settings information pertaining to predictive text functionality
Handheld Agent	Contains diagnostic information about the device, including information sent to the BlackBerry Enterprise Server for reporting
Handheld Configuration	Incoming half of the device agent that sends data from the BlackBerry Enterprise Server to the device
Handheld Key Store	Securely stores encryption keys for encrypted communication and signing
Input Learning Data	Stores list of text that device uses to predict what user is attempting to type
Input Method Switcher Option	Stores selection of input language method when sequence key ALT-ENTER is initiated
Input System Properties	Stores text input and keyboard layout settings specified under Options | Language and Text Input
Key Store Manager	Manages the BlackBerry key store
Key Store Options	Configures options for the BlackBerry key store
LDAP Browser Options	Contains Lightweight Directory Access Protocol (LDAP) browser setting information
Location Based Services	Used mainly for corporate users in a BlackBerry Enterprise Server environment, either for Location Tracking or can be accessed by third-party applications that are developed by BlackBerry MDS Studio
Mailbox Icon Management Options	Stores information regarding inbox settings and delivery of e-mail messages to certain message icons
Map Locations	Stores locations added to Favorites in BlackBerry Maps

Map Settings	Stores configuration settings in BlackBerry Maps | Options
MemoPad Options	Stores MemoPad settings
Memory Cleaner Options	Stores configuration settings in Options | Security Options | Memory Options; set status to Enable
Memos	Contains the memos stored on the device
Message List Options	Defines the way messages are displayed in the messages application
Messages	Contains all the messages on the device
Messenger Options (GoogleTalk)	Contains setting information for Google Talk instant messaging client
Messenger Options (WLM)	Contains setting information for Windows Live Messenger instant messaging client
Messenger Options (ST)	Contains setting information for IBM Sametime instant messaging client
Messenger Options (OC)	Contains setting information for Microsoft Office Communicator instant messaging client
Messenger Options (Yahoo!)	Contains setting information for Yahoo! Messenger client
MMS Messages	Database that contains a history of sent or received Multimedia Messaging Service (MMS)
MMS Options	Stores MMS settings
Mobile Network Options	Contains information outlining current mobile network settings and configurations
On-Board Device Memory	Database for the on-board device memory that enables the user to store media files, normally in *.cab extension
Options	Stores miscellaneous configuration options
PasswordKeeper	Application that uses Advanced Encryption Standard (AES) technology to store password entries on the device; see KB19098
PasswordKeeper Options	Contains settings configured via Password Keeper | Options
PGP Key Store	Contains the PGP private key and public keys
Phone Call Logs	Logs phone calls made to and from the device
Phone History	Stores information pertaining to phone call history with specific participants (complete history of incoming and outgoing phone calls with selected recipients)

Phone Hotlist	Stores information on the last *x* number of calls placed from the device
Phone Options	Stores phone settings
PIM Folder List - All	Outlines complete list of PIM folders within the associated mailbox; can be set up like a filter—if there are multiple folders, user can filter contact lists based on specific folder
PIN Messages	Stores personal identification number (PIN) messages sent or received
Policy	Stores the IT policy for the device
Profiles	Contains various smart phone alerts, such as a vibration for a new calendar appointment or a tone when a new message arrives
Profiles Options	Selects the current profile for alerts
Purged Messages	Contains reference for messages deleted from the device
Quick Contacts	Stores telephone numbers in the Speed Dial list
Random Pool	Stores numbers to generate random numbers securely for encryption
Recent Contacts	Stores list of contacts that have recently been contacted (phone or e-mail), such as a quick contact reference tool-tip when creating an e-mail
Recipient Cache	Contains encryption profiles for the people with which the user communicates
RMS Databases	Stores information about registered applications
S/MIME Options	Stores configuration information for Secure Multipurpose Internet Mail Extensions (S/MIME) settings
Saved Email Messages	Contains all saved messages stored on the device
Searches	Stores settings configured in search application
Secure Email Decision Maker	Automatically selects secure e-mail sending method (determined by whether the original sender sent a signed/encrypted e-mail—the device determines whether to send a signed/encrypted/plain text e-mail back)
Service Book	Stores all of the service records that enable the device to access various services
Setup Wizard Options	Contains options configured for personal settings for the device
SheetToGoPrefs	Contains setting information for the Sheet To Go application
SlideshowToGoPrefs	Contains setting information for the Slideshow To Go application
Smart Card Options	Stores saved settings configured for the BlackBerry Smart Card reader

SMS Messages	Contains Short Message Service (SMS) text messages sent to and from the device
Sounds	Contains various notification sounds, such as a vibration for a new calendar appointment or a tone when a new message arrives on the device; for BlackBerry Device Software 5.0
Spell Check Options	Stores settings configured in Options | Spell Check
SureType Options	Stores SureType keyboard settings specified under Options | Language
Tasks	Lists task items on the device
Tasks Options	Saves settings in Task | Options
Text Messages	Combines total number of SMS text messages received (database in BlackBerry Device Software 6.0)
TCP/IP Options	Stores TCP/IP configuration settings
Theme Settings	Stores settings for icon arrangement on device Home screen
Time Zones	Stores the time zone table
TLS Options	Configures Transport Layer Security settings
Trusted Key Store	Contains the trusted keys for the device
VideoRecorder Options	Saves settings in Video Camera | Options
Voice Activated Dialing Options	Stores settings configured in Options | Voice Dialing
WAP Push Messages	Contains information from Wireless Application Protocol (WAP) push services
WLAN Profiles	Saves Wi-Fi profiles created in Wi-Fi settings
WordToGoPrefs	Contains setting information for the Word To Go application
WTLS Options	Contains settings for Wireless Transport Layer Security

Using Rubus software (now deprecated) from CCL Forensics (cclgroupltd.com) or AccessData FTK, the examiner can deconstruct the IPD file (see Figure 12-13). Once this file is deconstructed, the examiner can locate the proper database and decode the underlying data.



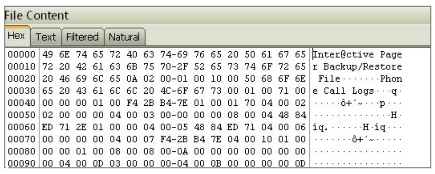
```
8703e.ipd
  Address Book (346)
  Address Book Options (1)
  Alarm Options (1)
  Application Permissions (1)
  Attachment Data (0)
  Attachment Options (1)
  AutoText (108)
  Bluetooth Options (1)
  Browser Bookmarks (5)
  Browser Channels (0)
  Browser Data Cache (2)
  Browser Folders (2)
  Drowser Messages (0)
  Browser Options (1)
  Browser Push Options (1)
  Browser Urls (21)
  Calendar (26)
  Calendar Options (1)
  Categories (2)
  Certificate Options (0)
  Content Store (19)
  Currency Key Option (1)
  Default Service Selector (2)
  Device Options (0)
  Email Filters - 342 (0)
  Email Settings - 342 (1)
  Firewall Options (1)
  Folder Id (1)
  Folders (10)
  Handheld Agent (262)
  Handheld Configuration (0)
  Handheld Key Store (33)
  Help Options (1)
  Input Method Switcher Option (1)
  Key Store Options (1)
  KeyStoreManager (1)
  MemoPad Options (1)
  Memory Cleaner Options (1)
  Memos (0)
```

```
00000000   44 65 6C 74 61 00                              Delta.
```

Start: 0 End: 0 Length: 0

FIGURE 12-13 **IPD deconstruction in CCL Rubus**

For Windows-produced BBB files from BlackBerry Desktop version 7, all the databases listed in Table 12-6 do not exist. The BBB file contains the manifest file that lays out the information within the .dat files. The manifest (Figure 12-14) contains the version of BBDM (dtmversion), the device PIN (SourceDevice PIN), and the databases that have been backed up into the BBB file. Each database listed in the manifest provides the number of records in the file (recordcount) along with the database name. Each .dat file is its own IPD file, which is evident by the file header of each .dat file, Inter@ctive Pager Backup. If a mobile forensic solution does not allow the automated ability to parse one or any of the included .dat files, these files can be manually interrogated to obtain user information. It is important for the examiner to look to the manifest file to determine the data that actually exists within the BBB file. In doing so, he or she has a reasonable expectation of what will and will not be recoverable with the automated software.



```
00000 49 6E 74 65 72 40 63 74-69 76 65 20 50 61 67 65 Inter@ctive Page
00010 72 20 42 61 63 6B 75 70-2F 52 65 73 74 6F 72 65 r Backup/Restore
00020 20 46 69 6C 65 0A 02 00-01 00 10 00 50 68 6F 6E  File······Phon
00030 65 20 43 61 6C 6C 20 4C-6F 67 73 00 01 00 71 00 e Call Logs···q·
00040 00 00 00 01 00 F4 2B B4-7E 01 00 01 70 04 00 02 ·····ô+´~···p···
00050 02 00 00 00 04 00 03 00-00 00 00 08 00 04 48 84 ··············H·
00060 ED 71 2E 01 00 00 04 00-05 48 84 ED 71 04 00 06 íq.······H·íq···
00070 00 00 00 00 04 00 07 F4-2B B4 7E 04 00 10 01 00 ·······ô+´~·····
00080 00 00 01 00 08 00 08 00-0A 00 00 00 00 00 00 00 ················
00090 00 04 00 0D 03 00 00 00-04 00 0B 00 00 00 00 0D ················
```

FIGURE 12-14 **The .dat file header from a BBB file indicating the typical IPD header information**

BlackBerry Data Formats and Data Types

As discussed, manual exploration is sometimes necessary—for example, when an automated tool fails to retrieve the necessary data or to verify the validity of the parsed information. Before embarking on a manual exploration, the examiner must be familiar with the various formats BlackBerry uses within the IPD and BBB files. Much like the rest of mobile device forensics, various formats are used throughout the device backups; they are listed in Table 12-7 with information gathered from the *BlackBerry Developers' Journal*, Mike McDonough's IPD Parse (https://sites.google.com/site/ipdparse/home), and Yogesh Khatri's Forensic Blog (www.swiftforensics.com). Table 12-7 will be further examined and used in the "Manual Decoding" section next.

TABLE 12-7 **Data Formats Used for Manual Interpretation Within a BlackBerry Backup File**

Artifact	Category	Format	Epoch	Example
Phone call log	Date/Time	HEX value little-endian 8 bytes	1970/1/1	4A AE 6C 12 2F 01 00 00 2011/04/1 18:56:43 GMT
Phone call log	Text	HEX to ASCII		+12085559876
SMS	Date/Time	HEX value little-endian 8 bytes	1970/1/1	BC 5C 75 D7 36 01 00 00 2012/04/22 00:30 GMT
SMS	Text	HEX to ASCII		This person called.
Phone history	Date/Time	HEX value little-endian 8 bytes	1970/1/1	CC 58 72 E5 36 01 00 00 2012/04/24 17:42 GMT
Phone history	Text	HEX to ASCII		+14547990987
Calendar	Date/Time	HEX value little-endian 4 bytes	1900/1/1	74 10 78 03 2010/08/28 19:00 GMT
Calendar	Text	HEX to ASCII		Emik's Birthday
Messages (E-mail)	Date/Time	HEX value little-endian 4 bytes	First bytes = Date Second bytes = Time	25 1C D9 61 2010/01/05 12:14 GMT
Messages (e-mail)	Text	HEX to ASCII/ Unicode		Test Message…
PIN messages	Date/Time	HEX value little-endian 4 bytes	First bytes = Date Second bytes = Time	A1 20 21 53 2012/5/1 10:25 GMT

Manual Decoding

BlackBerry uses little-endian decoding throughout the IPD and version 2 BBB files but uses two epoch dates. The UNIX epoch of January 1, 1970, is used as the basis for most databases within the BlackBerry backup image, with the exception of Calendar and Tasks, which use January 1, 1900, and e-mail and PIN messages, which use 1996 as an epoch year. The date and time fields for BlackBerry IPD files are not well documented. A few can be found, but most work comes in to recognizing the various formats and using a tool such as the MFI HEX Assistant or DCode to convert the dates and times. As mentioned, there can be a lot of heavy lifting involved.

Note

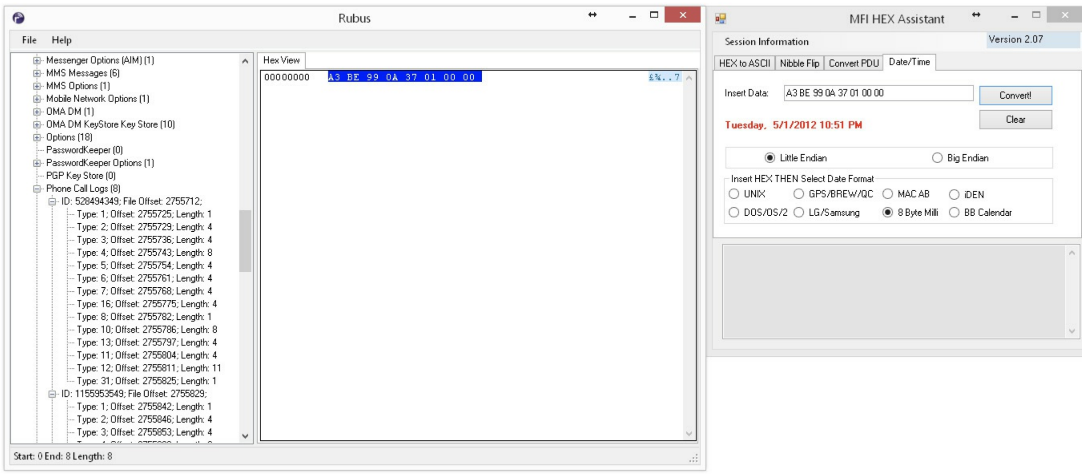
Many more databases are contained within the IPD and BBB, but using the information in this section can help the examiner in identifying information within many databases not listed.

A better explanation of the dates and times for the data within Table 12-7 is covered next, including data types with specific information and guidance on locating the date and time, along with how to convert the information.

Note

Rubus (since deprecated) from CCL Forensics and the MFI HEX Assistant were used for this section for IPD files, and FTK and the MFI HEX Assistant were used for v2 BBB files. After loading the IPD or BBB file into the solution of choice, the examiner can view the raw backup.

- **Phone call logs** In Rubus, the second field and the first byte indicate the type (missed-02, outgoing-01, incoming-00), and the fourth field is the date and time (see Figure 12-15). This is an 8-byte value that, when read as a little-endian value to obtain a 64-bit integer value, can be assessed as milliseconds from 1970/1/1.



```
Messenger Options (AIM) (1)
MMS Messages (6)
MMS Options (1)
Mobile Network Options (1)
OMA DM (1)
OMA DM KeyStore Key Store (10)
Options (18)
PasswordKeeper (0)
PasswordKeeper Options (1)
PGP Key Store (0)
Phone Call Logs (8)
  ID: 528494349; File Offset: 2755712;
    Type: 1; Offset: 2755725; Length: 1
    Type: 2; Offset: 2755729; Length: 4
    Type: 3; Offset: 2755736; Length: 4
    Type: 4; Offset: 2755743; Length: 8
    Type: 5; Offset: 2755754; Length: 4
    Type: 6; Offset: 2755761; Length: 4
    Type: 7; Offset: 2755768; Length: 4
    Type: 16; Offset: 2755775; Length: 4
    Type: 8; Offset: 2755782; Length: 1
    Type: 10; Offset: 2755786; Length: 8
    Type: 13; Offset: 2755797; Length: 4
    Type: 11; Offset: 2755804; Length: 4
    Type: 12; Offset: 2755811; Length: 11
    Type: 31; Offset: 2755825; Length: 1
  ID: 1155953549; File Offset: 2755829;
    Type: 1; Offset: 2755842; Length: 1
    Type: 2; Offset: 2755846; Length: 4
    Type: 3; Offset: 2755853; Length: 4

00000000   A3 BE 99 0A 37 01 00 00        £¾..7

Start: 0 End: 8 Length: 8
```

```
Insert Data:  A3 BE 99 0A 37 01 00 00

Tuesday,  5/1/2012 10:51 PM
```

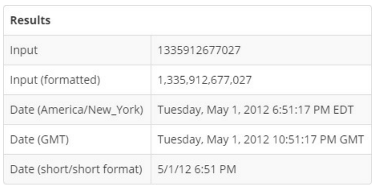
[✓ Little Endian] Big Endian

Insert HEX THEN Select Date Format: UNIX, GPS/BREW/QC, MAC AB, iDEN, DOS/OS/2, LG/Samsung, [✓ 8 Byte Milli], BB Calendar

FIGURE 12-15 **Call log date and time in the fourth field and converted with the MFI HEX Assistant**

Note

The value A3BE990A37010000 in LE (little-endian) is 000001370A99BEA3. If this is input into the scientific calculator as HEX and then converted to decimal, the value is 1335912677027. This value can then be plugged into www.fileformat.info/tip/java/date2millis.htm and, upon conversion, the date is displayed (see Figure 12-16).

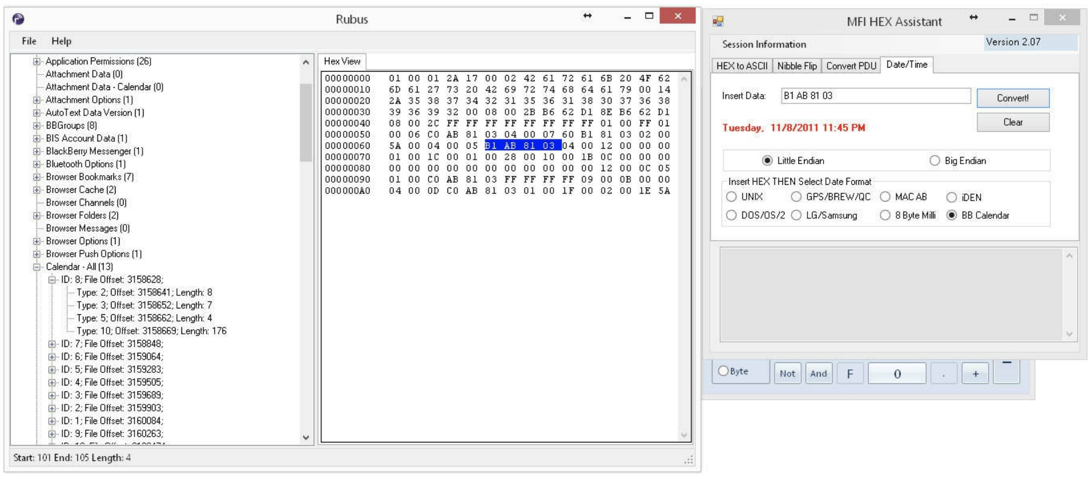


Results	
Input	1335912677027
Input (formatted)	1,335,912,677,027
Date (America/New_York)	Tuesday, May 1, 2012 6:51:17 PM EDT
Date (GMT)	Tuesday, May 1, 2012 10:51:17 PM GMT
Date (short/short format)	5/1/12 6:51 PM

FIGURE 12-16 **Use www.fileformat.info on the Web to convert the HEX value into a readable date.**

- **Phone history** In Rubus, the third field is the date and time field and the fourth field is of the type exactly like Phone Call Logs. The 8-byte value can be converted just like phone call logs.

- **SMS messages** In Rubus, the field labeled 11 and the first byte will indicate the direction of the SMS (incoming-01 and outgoing-00), and the field labeled 16 will be the 8-byte date and time in the same format as described for phone call logs.
- **Calendar** In Rubus, the data is located in the field labeled 10 (see Figure 12-17). There are three dates in the Calendar (Start, End, and Reminder), and all dates use 4 bytes that must be converted to little-endian to obtain an integer. The integer will represent the number of minutes from 1900/1/1.



```
Application Permissions (26)
Attachment Data (0)
Attachment Data - Calendar (0)
Attachment Options (1)
AutoText Data Version (1)
BBGroups (8)
BIS Account Data (1)
BlackBerry Messenger (1)
Bluetooth Options (1)
Browser Bookmarks (7)
Browser Cache (2)
Browser Channels (0)
Browser Folders (2)
Browser Messages (0)
Browser Options (1)
Browser Push Options (1)
Calendar - All (13)
    ID: 8; File Offset: 3158628;
        Type: 2; Offset: 3158641; Length: 8
        Type: 3; Offset: 3158652; Length: 7
        Type: 5; Offset: 3158662; Length: 4
        Type: 10; Offset: 3158669; Length: 176
    ID: 7; File Offset: 3158848;
    ID: 6; File Offset: 3159064;
    ID: 5; File Offset: 3159283;
    ID: 4; File Offset: 3159505;
    ID: 3; File Offset: 3159689;
    ID: 2; File Offset: 3159903;
    ID: 1; File Offset: 3160084;
    ID: 9; File Offset: 3160263;
Start: 101 End: 105 Length: 4
```

```
00000000   01 00 01 2A 17 00 02 42 61 72 61 6B 20 4F 62
00000010   6D 61 27 73 20 42 69 72 74 68 64 61 79 00 14
00000020   2A 35 38 37 34 32 31 35 36 31 38 30 37 36 38
00000030   39 36 39 32 00 08 00 2B B6 62 D1 8E B6 62 D1
00000040   08 00 2C FF FF FF FF FF FF FF FF 01 00 FF 01
00000050   00 06 C0 AB 81 03 04 00 07 60 B1 81 03 02 00
00000060   5A 00 04 00 05 B1 AB 81 03 04 00 12 00 00 00
00000070   01 00 1C 00 01 00 28 00 10 00 1B 0C 00 00 00
00000080   00 00 00 00 00 00 00 00 00 00 00 12 00 0C 05
00000090   01 00 C0 AB 81 03 FF FF FF FF 09 00 0B 00 00
000000A0   04 00 0D C0 AB 81 03 01 00 1F 00 02 00 1E 5A
```

Insert Data: B1 AB 81 03

Tuesday, 11/8/2011 11:45 PM

[✓ Little Endian] Big Endian

Insert HEX THEN Select Date Format: UNIX, GPS/BREW/QC, MAC AB, iDEN, DOS/OS/2, LG/Samsung, 8 Byte Milli, [✓ BB Calendar]

FIGURE 12-17 **Calendar date and time format within field 10 using Rubus**

- **Messages** E-mail messages were examined within Rubus, and because there are many fields of interest in messages, they are compiled in Table 12-8.

TABLE 12-8 **E-mail Messages Decoded Using Rubus**

Field Number	Type	Location	Format
0	Unread - F2 08 Read - F2 00 Sent - F3 00 Unsent - E3 00	bytes 3–4	
0	Date/Time (three will be found; last time is received/sent)	bytes 29–40	4 byte LE (Date/Time)
1	To		ASCII/Unicode
5	From		ASCII/Unicode
11	Subject		ASCII/Unicode
12	Message if text		ASCII/Unicode
22	Attachments		ASCII/Unicode
24	Message if formatted		HTML

The date and time method for converting the dates is rather intensive. The date and time is a 4-byte value: the first 2 bytes represent the date and the second 2 bytes represent the time. To convert this date and time to a Julian date, the examiner must take the 2 bytes into little-endian order: 81 1E becomes 1E 81, and this value must then be converted into a binary number. The binary number 0001111 0100 00001 must be divided up into the most significant (on the left) 7 bits—0001111 represents the days from 1996, the next 4 bits, 0100, represent the month (1–12), and the final 5 bits represent the day (1–31).

```
0001111 0100 00001
\_____/ \__/ \___/
Y(2011) M(4)  D(1)
```

The binary values will be converted to a decimal number to obtain the decoded date. The binary number, as presented, would be 0001111 = 15 (1996 + 15 = 2011), 0100 = 4 (April), 00001 = 1 (1). The date for this message is April 1, 2011.

The time is broken down in much the same manner. Here the time is 18 06, and switching to little-endian would be 06 18 and the binary representation is 00000 110000 11000.

```
00000 110000 11000
\___/ \____/ \___/
H(0)   M(48) S(24)
```

To decode the time, the examiner, starting at the MSB, will take 5 bits that represent the hour, the next 6 bits that represent the minutes, and the remaining 5 bits that represent the seconds. Converting the binary to decimal and understanding the time is GMT time using a 24-hour clock, the time is converted to 0:48:24 hours.

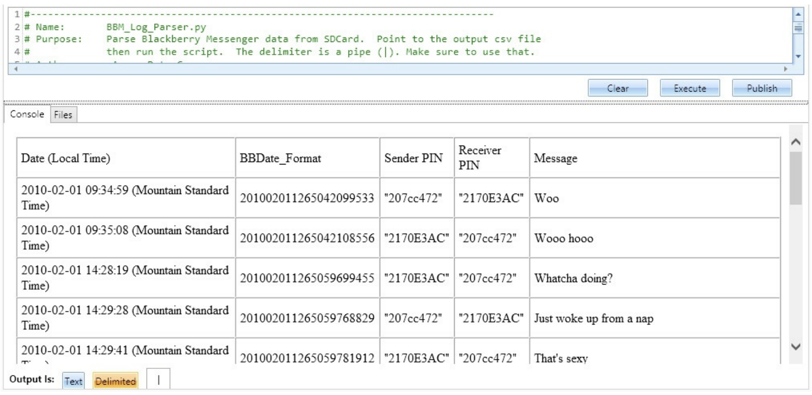
- **PIN messages** In Rubus, PIN messages are decoded much like messages, described previously. The significant fields are described in Table 12-9, with the only significant difference being the unsent PIN message represented by EB 00. The date and time location and format are identical to those of messages and can be followed to decode the values in field 0.

TABLE 12-9 **PIN Message Significant Field with Type and Format**

Field Number	Type	Location	Format
2	Contact List (PIN, Status, Name, UID)		ASCII/Unicode
3	BBM User Name		ASCII/Unicode
4	Contact UID		ASCII/Unicode
10	User PIN		ASCII/Unicode
20	User Image (JPG)		Binary

- **BlackBerry Messenger message logs** BlackBerry Messenger has evolved over the many years BlackBerry has been in existence. BlackBerry OS up to BB10 enables the user to store chat logs on the device or media card if the user selects this feature in settings, but this setting is not the default. For BB10 devices, the user turns on or off the chat history function. BBM chat logs are not backed up into the IPD or v1 and v2 BBB files. However, the BBM contact list for devices up to BBM v7 are backed up and available within the IPD/BBB (BlackBerry KB18406), and its format is referenced in Table 12-8, as decoded within Rubus. BBM 7x contacts and settings are synched using the BlackBerry ID and are not stored within a backup file (BlackBerry KB20554).

If the user has enabled the option to save BBM message logs, the logs can be recovered in CSV within the content store on the device or on the external media card. The folder containing the BBM data is identified on the device and external card at <home>im\BlackBerry Messenger\<Users PIN>. Within the <history> folder are several CSV files that are uniquely named by the PIN of the user they were communicating with. There is also a <backup> folder that contains the BBM contact list (.con) and backups of the contact list based upon a date (.bak). The chat history .csv file contains the BBM conversation in the following order: date, sender PIN, receiver PIN, message. The date is formatted rather bizarrely, with the first eight most significant digits representing the year, month, and day, and the remaining digits representing the milliseconds from 1970/1/1 and in GMT. This information is shown in Figure 12-18 as parsed using AccessData MPE+ and the pythonScripter.


```
#------------------------------------------------------------------------------
# Name:       BBM_Log_Parser.py
# Purpose:    Parse Blackberry Messenger data from SDCard.  Point to the output csv file
#             then run the script.  The delimiter is a pipe (|). Make sure to use that.
```

Date (Local Time)	BBDate_Format	Sender PIN	Receiver PIN	Message
2010-02-01 09:34:59 (Mountain Standard Time)	201002011265042099533	"207cc472"	"2170E3AC"	Woo
2010-02-01 09:35:08 (Mountain Standard Time)	201002011265042108556	"2170E3AC"	"207cc472"	Wooo hooo
2010-02-01 14:28:19 (Mountain Standard Time)	201002011265059699455	"2170E3AC"	"207cc472"	Whatcha doing?
2010-02-01 14:29:28 (Mountain Standard Time)	201002011265059768829	"207cc472"	"2170E3AC"	Just woke up from a nap
2010-02-01 14:29:41 (Mountain Standard	201002011265059781912	"2170E3AC"	"207cc472"	That's sexy

FIGURE 12-18 **MPE+ and pythonScripter output from running the BBM parsing script against a backup file produced by a BlackBerry device**

Note

One the best *free* Python scripting tools available is PyScripter (https://sourceforge.net/projects/pyscripter/) if you are working only in Windows, with Eric Python IDE (http://eric-ide.python-projects.org) coming in second, simply because it is a cross-platform tool, albeit a full development environment.

If a BBM chat logging service has not been set by the user, the examiner must attempt to obtain either a non-invasive or an invasive physical collection of the device. With a physical collection, the device's internal system can then be examined, and BBM chat is accessible even if the user has not enabled the logging function from the bbm.db file. The data formats along with the data that was previously stored (device and owner information, contacts [.con/.bak], and chat) within the IPD/BBB and contact database are now contained within the bbm.db file. Within some BlackBerry backup files, another database, BBM

Conversations, can be located and used to enhance the search ability by the BlackBerry device. This database location and file \BBM Conversations\conversations.csv is not included in every backup, and the company that developed BlackBerry Backup Extractor, Reincubate, can parse the .csv file that is formatted much like the .csv backup file as previously discussed. This software is still available (www.blackberryconverter.com/), but it will parse only the legacy IPD files.

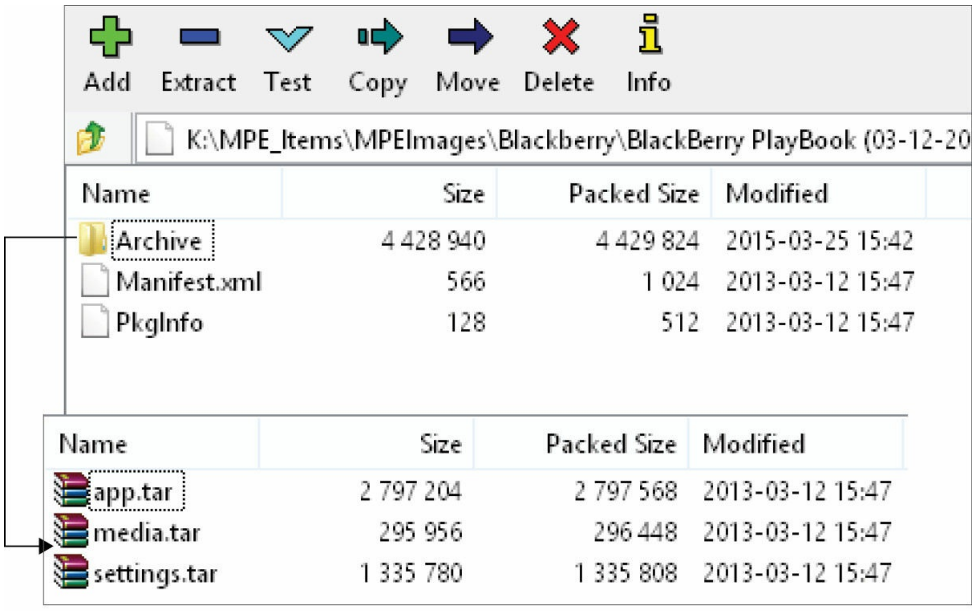
BlackBerry 10 File System

With BB10, many things drastically changed, starting with the operating system, which changed from the BlackBerry Mobile OS (Java MicroEdition-J2ME) to the QNX Neutrino platform. The QNX Neutrino platform is designed to run on several servers concurrently, with only 100,000 lines of code, compared to a single large program such as Linux (9 million lines) or Windows (70 million lines). This is said to improve the operability and reliability of the BlackBerry platform. Table 12-10 outlines the application working areas and repositories an examiner should be familiar with when conducting analysis of BB10 device apps.

TABLE 12-10 **BB10 App Working Directory Locations (from https://developer.blackberry.com/native/documentation/cascades/device_platform/data_access/file_system.h**

Folder	Description
app	Files installed with the application, packaged with the .bar file
app/native	Application binary and other application resources
app/native/assets	Application assets, such as .qml files, images, and media
data	Application's private data; the application's home directory
db	Application's database files
logs	System logs for an application; stderr and stdout are redirected to this directory
shared	Subfolders that contain shared data grouped by type; an application can write to subfolders of this directory only if the access_shared permission is specified
shared/books	eBook files that can be shared between applications
shared/camera	Images taken using the Camera application that can be shared between applications
shared/documents	Documents that can be shared between applications
shared/downloads	Web browser downloads that can be shared between applications
shared/Dropbox	Files located in the user's Dropbox account
shared/misc	Files that don't belong in any other category and can be shared between applications
shared/music	Music files that can be shared between applications
shared/photos	Photos that can be shared between applications
shared/print	News and print articles that can be shared between applications
shared/videos	Videos that can be shared between applications
shared/voice	Audio recordings that can be shared between applications
sharewith	Files that the application can share with other applications by using the Invocation framework
tmp	The application's temporary working files; application should remove these files regularly—the BB10 OS may remove these files when the application isn't running

The examination of the BB10 device can occur at different levels. Currently the backup that is produced for the BlackBerry PlayBook is different from the backup from the other BB10 devices. The BBB file from the BlackBerry PlayBook is a compressed file that can be opened with 7-Zip or WinZip, and it contains the same two files (covered previously) included with the v2 BBB (manifest and PkgInfo) file, but it also includes three new tar files, app.tar, media.tar, and settings.tar, that are unencrypted QNX files (see Figure 12-19). All BB10 devices produced after the BlackBerry PlayBook produce the same BBB files with a few exceptions. The associated tar files are now encrypted with the BlackBerry ID and password, and the manifest file contains the BlackBerry ID associated with each archive type, located in the perimeter type for the archive (for example, Application [app], Media [media], and Settings [settings]). The BlackBerry PlayBook files are unencrypted QNX files, while the rest of the BB10 devices have encrypted QNX files. If the examiner knows the Link password, both Elcomsoft and Oxygen Forensic Analyst enable the backup to be imported and produce an export that contains call logs, applications, phonebook (contacts), messages, and a file system containing the decrypted app, settings, and media folders.


Add
Extract
Test
Copy
Move
Delete
Info
K:\MPE_Items\MPEImages\Blackberry\BlackBerry PlayBook (03-12-20

Name	Size	Packed Size	Modified
Archive	4 428 940	4 429 824	2015-03-25 15:42
Manifest.xml	566	1 024	2013-03-12 15:47
PkgInfo	128	512	2013-03-12 15:47

Name	Size	Packed Size	Modified
app.tar	2 797 204	2 797 568	2013-03-12 15:47
media.tar	295 956	296 448	2013-03-12 15:47
settings.tar	1 335 780	1 335 808	2013-03-12 15:47

FIGURE 12-19 **BBB file from a BlackBerry PlayBook, with the Archive folder instead of the Database folder, as with previous BBB files**

Tip

For BlackBerry backups (OS 7 and earlier) that are encrypted, Elcomsoft also enables the bypass of the password using a brute-force attack of plain text passwords.

Here's another option: If the password is known or the device is not locked, the examiner can plug the BB10 device into a computer and mount the device as a drive. The BB10 will show two drives: One drive will provide access to the internal memory of the BB10 device and anything on the shared path, including books, camera, documents, downloads, Dropbox, miscellaneous music, photos, print, videos, and voice. The other mounted network drive will be the removable storage that will duplicate some of the same folders observed on the mounted internal storage area, but these folders contain unique data. The external SD card should be analyzed for any deleted files using a forensic solution and a built-in file carving utility. This acquisition is conducted using the Media Transfer Protocol (MTP) and was outlined in earlier chapters. This area does not contain PIM data—only files and folders used by applications to store and share data. To obtain PIM data, the examiner must access the logical or physical file system of the BB10 device.

A limited amount of user information is automatically parsed from the backup, even if the examiner is using a forensic solution that can obtain the logical file system or a physical binary image. This means that a lot of manual work is required with IPD and BBB files, but the rewards are worth the effort. With BB10 devices, an incredible amount of information is available to the examiner. These file systems now contain apps with a wealth of information within the SQLite databases. SQLite databases will be covered in Chapter 14, but Table 12-11 represents great locations for finding valuable data within the BB10 device. (This table is not an all-inclusive list, and some areas are not covered here because they were included in Table 12-10.) For date and time artifacts, the time is stored in milliseconds, from 1970/1/1, as is the case with IPD and BBB files.

TABLE 12-11 **Artifact Locations for BB10 File System Examinations**

Artifact	Path	Considerations	Comments
GEO location	/accounts/1000/sys/placesservice	Places.db3	Mapping database with latitude and longitude information for device user
User added dictionary	/accounts/1000/sys/input/spell_check	personal_dictionary.wrd	Words added to dictionary by user
Facebook	/accounts/1000/appdata/com.rim.bb.app.facebook	fb-messages.db mail.db friends.db	Messages contain messages and threads; mail contains data on mail interactions with app
WhatsApp	/accounts/1000/appdata/com.whatsapp.WhatsApp	messageStore.db	Call logs and messages
Android PIM	/accounts/1000/appdata/sys.android	databases = autofill.db browser2.db webview.db	Many other databases, including contacts, messages, downloads, and more
BBM	/accounts/1000\appdata/sys.bbm	master.db bbgroups.db	Contacts, text messages, calls
Browser	/accounts/1000/appdata/sys.browser	Chrome/Databases.db Chrome/localstorage/local_X.localstorage/ webviews/creditials.db.archivedautofill.db Databases.db	Cached WebKit, searches, and usernames

Messages/ SMS/e-mail	/accounts/1000/ appdata/sys.pim .messages	unifed.db	Uses table category and UIB Item to match up conversational data from many different types of content
MMS/SMS	/var/db/	messages.db attachments folder	MMS and SMS with links to attachments folder
MediaStore	/accounts/1000/db/	mmlibrary.db	Path information for all media on device internally
MediaStore	/accounts/1000/db/ sdcard_dbcache	<media card serial>.db	Path information for all media on media card even if removed
Media associated to PIM	/accounts/1000/ pimdata/	Various folders contacts, messages, events associated to data in /sysdata/db and referenced by number	Media and messages used by PIM database; messages from SMS and various applications cached here
User usage	/accounts/1000/ sysdata/pim/ analytics/	analytics.db	Statistics on user (such as e-mails sent, contacts)
PIM storage	/accounts/1000/ sysdata/pim/db/	Many SQLite databases	Stores built-in PIM used and cross-linking data for application use of user PIM data
PIM calendar	/accounts/1000/ sysdata/pim/db/	1.db	Uses Calendar Event table
PIM contacts	/accounts/1000/ sysdata/pim/db/	2.db	Uses Contacts table
PIM unified contacts	/accounts/1000/ sysdata/pim/db/	4.db	Data for contacts used in various apps and settings; has deleted field in Unified Contact table
PIM call logs	/accounts/1000/ sysdata/pim/db/	8.db	Uses Call Detail table for data; call table has unread attribute
PIM tasks	/accounts/1000/ sysdata/pim/db/	22.db	Uses Tasks table

With a logical file system extraction, the examiner will find user data of value in the settings.tar file, as observed in Figure 12-19. App storage data is located in /accounts/1000/appdata/ in the settings.tar file. In a physical collection, the /accounts/1000/appdata folder is located under root, as shown in Figure 12-20. The actual app code will be located in the app file in a logical extraction and in the app folder in a physical collection. The examiner can scan the app code for malware or viruses if needed to verify that no malicious apps were installed.

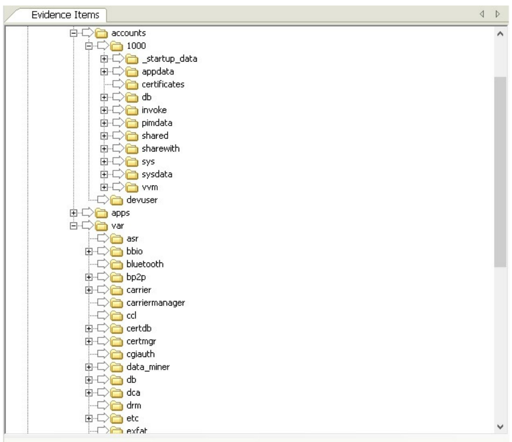


FIGURE 12-20 **File system view of the BB10 device from within FTK**

Note

There are many similarities between the files found in a BB10 file system and files found in both iOS and Android devices. Figure 12-21 shows a SQLite database, common to both iOS and Android. The investigation into the file types is covered in greater detail in Chapter 14. As a matter of history of the BlackBerry device and company is the fact that the company stopped manufacturing mobile devices and outsourced the hardware in 2016. A year later, BlackBerry was using Android as the underlying OS within the Priv, and then the KEYone device. Processing of a BlackBerry device has simply been relegated to processing an Android device and its known file system.

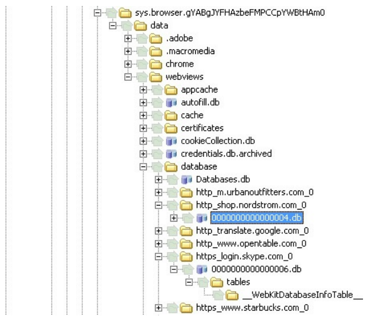


FIGURE 12-21 **Navigation into the file system to uncover SQLite databases within the sys.browser application subfolders**

Windows Phone

Windows Phones are used far less than both Android and iOS devices, but they do make their way into mobile forensic labs across the globe once in a while. Mobile forensic solutions and an automated feature set to obtain the PIM data are limited, if not nonexistent. To access these devices, the examiner must be able to access the internal file system of the device beyond the MTP data that mobile forensic solutions currently offer.

Windows Phone "Tip of the Iceberg Data"

When the Windows Phone 7 device is not locked, two logical file systems are available to the examiner—the internal storage area and the external media, if inserted. Forensic tools use MTP to obtain the file system, which is then brought into the mobile forensic solution. Some user files are stored on the media area, including documents, images, video, and audio. Some PIM data, such as the phonebook, is supported, but if the examiner is looking for messages, call logs, or other user data, access to a backup of a Windows Phone 7 or SkyDrive (now OneDrive) is necessary. However, when a file system is obtained, several Python scripts and EnCase EnScript programs can be run against the Windows Phone file system to gather "tip of the iceberg data." Much like Windows Phone 7, 8, and 8.1, current solutions for Windows 10 Mobile will use MTP for onboard data from the media area, and other data will need to be recovered from the user's cloud account for the Windows device. As covered in Chapter 10, data for the Windows 10 Mobile device is limited if access to the file system cannot be made.

Windows Phone File System

A forensic examination of a Windows Phone should include the internal file system. Currently, to access valuable user information, the examiner must obtain a physical image, either by non-invasive or invasive methods. Most examiners use JTAG methods for the Windows Phone because JTAG offers a non-invasive method of access to the device without requiring complete disassembly, and many Windows Phone models are supported. A Windows Phone 7 device, depending on the OS version, can be collected with tools such as TouchXperience and Una to obtain a file system–level extraction and registry dump. For Windows Phone 8 devices, most examiners rely on JTAG, but Cellebrite tools currently enable the examiner to ascertain a binary image of select Windows Phone 8 devices as well.

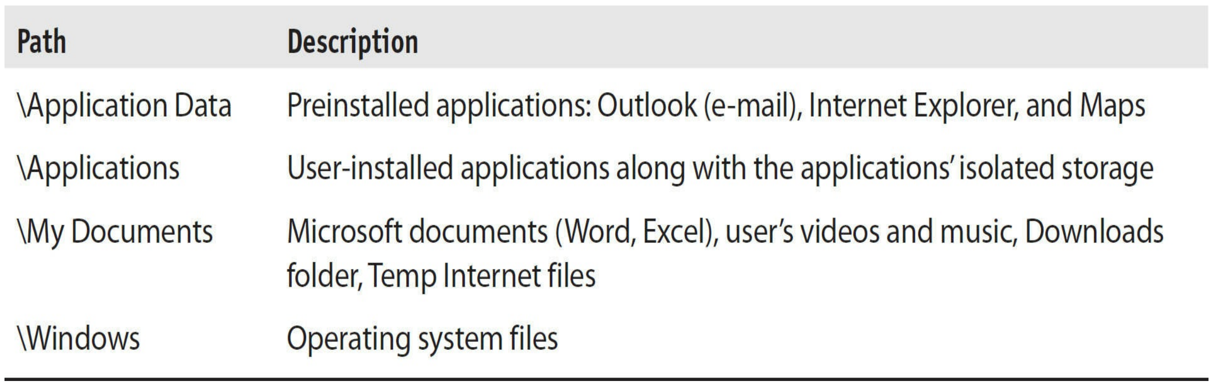
After the images are obtained from the Windows Phone, the examination is much like the BlackBerry examination—extremely labor intensive. Instead of a QNX EFS, the underlying file system is NTFS. For computer forensic examiners, forensic tools that handle Windows Phone file systems more efficiently (such as EnCase, FTK, or X-Ways) can be used.

Because Windows Phone 7 devices were discontinued and all support by Microsoft was stopped in October 2014 in favor of Windows Phone 8 devices, limited references to Windows Phone 7 is made here, but several tables list important areas for investigation. The file system and artifact locations are different from those of the current Windows Phone 8 device file system, but they resemble those of the standard Windows desktop file system. Most of the Windows Phone discussion here is devoted to the Windows Phone 8. In late 2015, Microsoft wanted to move to a "cross-platform" solution, in which data from computers and mobile devices kept in constant synch; enter Windows 10 devices. Again, much like BlackBerry devices, the market share and the likelihood that the examiner will encounter this device are almost nonexistent. As a result, most commercial tools do not spend development cycles on supporting older Windows phones. That being said, the Windows 10 device will be discussed minimally within the chapter.

Windows Phone 7

The Windows Phone 7 file system resembles the Windows desktop system, with locations for applications, documents, and the operating system files, as outlined in Table 12-12.

TABLE 12-12 **Evidence Folders in Root of Windows Phone 7 File System**



Path	Description
\Application Data	Preinstalled applications: Outlook (e-mail), Internet Explorer, and Maps
\Applications	User-installed applications along with the applications' isolated storage
\My Documents	Microsoft documents (Word, Excel), user's videos and music, Downloads folder, Temp Internet files
\Windows	Operating system files

Obtaining common user data from a Windows Phone 7 device can also be accomplished by navigating to the device file system. Table 12-13 lists some common locations for various data, and some of the files, such as store.vol, are used within the Windows Phone 8 device and are explained in detail in the following sections.

TABLE 12-13 **Common Artifact Locations Within Windows Phone 7 Device**

Artifact	Path	Considerations	Comments
Conversation	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Call logs, SMS messages, e-mail subjects	All items that have appeared on the device screen, in chronological order
Contact	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Names, e-mail, phone numbers, business addresses	Culmination of records from apps (such as Facebook, Twitter), device contact list
Appointments	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Calendar appointments	Information in Unicode, ASCII, dates/times
Attachments	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Attachments for various applications	Name of attachment, type of artifact
Message	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	MMS and SMS	Message content, numbers, date and time information
Note	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Notes	Notes, content, date and time information
Tasks	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\store.vol	Tasks	Tasks, content, date and time information
E-mail	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\data\	Outlook client, enables use of IMAP services from Gmail, Exchange, others	Folders include .dat files; .dat files in folder 19 are e-mail messages in HTML format
Internet Explorer	\Application Data\ Microsoft\Outlook\Stores\ DeviceStore\data\	Various Internet files	TabsStorage.bin lists opened tabs in IE
Apps	\Applications\Data\<app GUID>\Data\IsolatedStore	All app information	Application cache, SQLite databases, other application settings

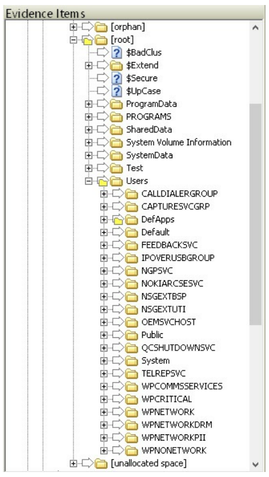
Location	\Application Data\Location\Cache	Wi-Fi and cellular location	If user enabled, lists locations for cellular and Wi-Fi
Registry	MainOS Windows\systerm32\config	Security Account Manager (SAM), security, software, system	Information on typed URLs
Maps	\Application Data\Maps	Map data	If user accessed maps, locations found in MapsSetData.dat

Most of the file types and formats carried over to the Windows Phone 8, but the path to the file(s) changed. As outlined on the XDA forum (http://forum.xda-developers.com/showthread.php?t=1072796), the store.vol file can be parsed, but the file must be renamed while it's stored on the device because the device operating system continuously has ownership of the file. If the device image is obtained via JTAG or another physical collection, this, of course, would not be the case. The renaming of the store.vol, in an effort to collect and examine data, would be necessary only if the examiner were connecting to the device live, via USB, and navigating the file system with Explorer.

Additional information in the store.vol file and other files located within the Windows Phone 7 file system is covered in greater detail in the next section.

Windows Phone 8

To help mitigate the deep dive into the Windows Phone 8 file system, Python scripts composed by Adrian Leong are available (through GitHub at https://github.com/cheeky4n6monkey/4n6-scripts) to help the examiner with the recovery of contacts, call logs, and SMS. These scripts can be run in PyScripter, Physical Analyzer, and XRY, to name a few, but the examiner must understand that the scripts are written to be output to a .csv file. If the mobile solution enables the running of Python scripts, the examiner must make sure that the output conforms to the solution, since the output can always be tweaked to fit the examiner's needs. Also, EnCase has an EnScript that was created from Leong's Python scripts at http://encase-forensic-blog.guidancesoftware.com/2014/10/encase-and-python-automating-windows.html. Having a file system accessible, as shown in Figure 12-22, will enable an examiner to utilize these powerful parsing scripts.



```
Evidence Items
[orphan]
[root]
    $BadClus
    $Extend
    $Secure
    $UpCase
    ProgramData
    PROGRAMS
    SharedData
    System Volume Information
    SystemData
    Test
    Users
        CALLDIALERGROUP
        CAPTURESVCGRP
        DefApps
        Default
        FEEDBACKSVC
        IPOVERUSBGROUP
        NGPSVC
        NOKIARCSESVC
        NSGEXTBSP
        NSGEXTUTI
        OEMSVCHOST
        Public
        QCSHUTDOWNSVC
        System
        TELREPSVC
        WPCOMMSSERVICES
        WPCRITICAL
        WPNETWORK
        WPNETWORKDRM
        WPNETWORKPII
        WPNONETWORK
[unallocated space]
```

FIGURE 12-22 **Internal file system of a Windows Phone 8 device displayed in FTK**

Table 12-14 outlines the file layout of the Windows Phone 8 data drive, and Table 12-15 shows the various evidence locations within the Windows Phone 8 file system. This is not an exhaustive list, but a starting point for the advanced analysis of the Windows Phone 8 file system.

TABLE 12-14 **Main File Layout for Windows Phone 8 Within the Data Partition**

Path	Description
\Programs	Windows Phone Store application code
\Users	Default user account, built-in services, and public data folders
\SystemData	System-wide files, such as logs and Device Update; data in this folder is not applicable to a particular application
\SharedData	System-wide shared data, or data shared between multiple applications
\Test	Test code and the data in its subdirectories

Windows Phone stores the majority of user data in an Extensible Storage Engine (ESE) database(s). ESE is a sequential storage technology developed by Microsoft that is remarkably different from other storage methods found on any of today's mobile devices. These storage databases contain numerous tables similar to other built-in databases with records and tables (see Figure 12-23). The database allows multiple pieces of transactional data, seemingly unrelated, to be stored in a single repository. A perfect example within the Windows Phone is the store.vol file. This file is located at \Users\WPCOMMSSERVICES\APPDATA\Local\Unistore (see Figure 12-24) and holds transactional data for contacts, messages, activity, conversations, appointments, and others. This method of data storage makes locating key information easy, as long as the information can be viewed. Using applications such as FTK and EnCase, the examiner can view the ESE files and the data contained within them.

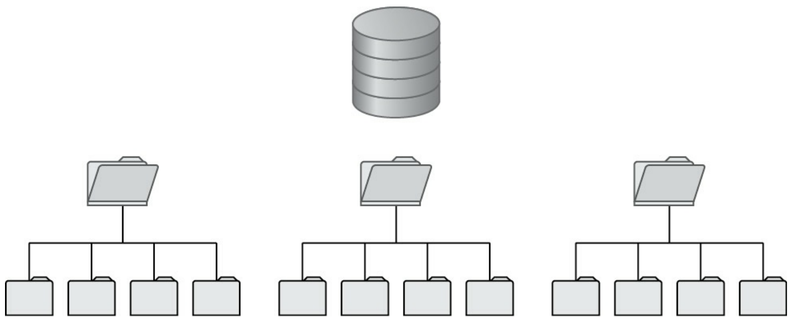
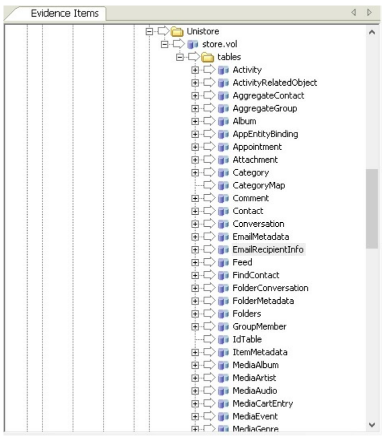


FIGURE 12-23 **ESE database structure with the root node on the top**

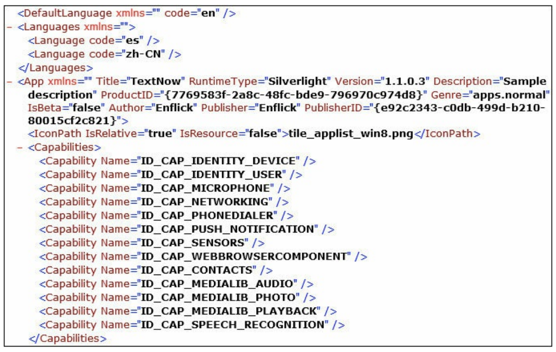


```
Evidence Items
Unistore
  store.vol
    tables
      Activity
      ActivityRelatedObject
      AggregateContact
      AggregateGroup
      Album
      AppEntityBinding
      Appointment
      Attachment
      Category
      CategoryMap
      Comment
      Contact
      Conversation
      EmailMetadata
      EmailRecipientInfo
      Feed
      FindContact
      FolderConversation
      FolderMetadata
      Folders
      GroupMember
      IdTable
      ItemMetadata
      MediaAlbum
      MediaArtist
      MediaAudio
      MediaCartEntry
      MediaEvent
      MediaGenre
```

FIGURE 12-24 **The store.vol ESE contains a wealth of information for an examiner.**

The data for installed apps is located in various subfolders and can be stored in SQLite databases, JSON, XML, and other file types. App data on a Windows Phone can be stored in several locations in the device file system, depending on the way the app was designed to store and share data. Two locations for data storage reside at Users\DefApps\APPDATA\. Microsoft Phone built-in apps such as Facebook and Skype are located in a subfolder at \APPDATA\Local, prefixed with *Microsoft*. Within the \APPDATA\Local folder are also some of the app GUIDs that were located in the root Users\DefApps\APPDATA. Each of the \APPDATA\Local apps includes a subfolder called AC (Application Cache), which contains certificate and crypto key information for any of the apps found in the root of the User\DefApps\APPDATA folder and identified with a GUID. However, some of the apps, such as the Microsoft apps, contain additional folders that pertain to Internet usage. Microsoft added the ability for each app to have its own Internet artifacts, which include its own cache, cookies, and history. These folders are called INetCache (Internet cache), INetCookies (Internet cookies), and INetHistory (Internet history). These are extremely important, especially the INetCache folder, which can store valuable chat messages when the application uses a web interface; Windows Phone devices cache these conversations in this folder. Also within the app folder system is the Local folder, which contains any data the app wants to persist across restarts, including databases files, settings files, and more. This folder can be found in the app folders in both the DefApps\APPDATA\ and the DefApps\APPDATA\Local folder and can contain valuable evidence.

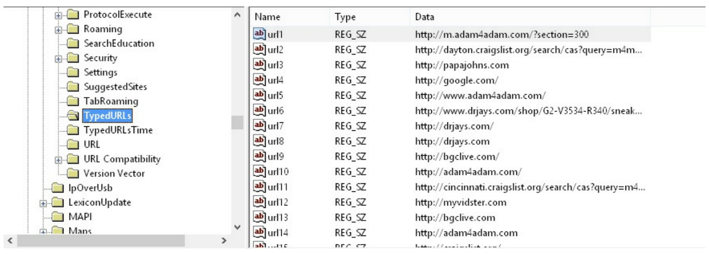
Many of the app names are GUIDs—the unique identifiers for the product that are declared in the app's WMAppManifest.xml (shown in Figure 12-25). The name of the app can be matched with the GUID by locating the WMAppManifest.xml file located in the \PROGRAMS\<Product GUID>\Install folder. The GUID is much like the naming convention observed in an iTunes backup, and the examiner can obtain additional information along with the actual working name of the application using that file path.



```
  <DefaultLanguage xmlns="" code="en" />
- <Languages xmlns="">
    <Language code="es" />
    <Language code="zh-CN" />
  </Languages>
- <App xmlns="" Title="TextNow" RuntimeType="Silverlight" Version="1.1.0.3" Description="Sample
    description" ProductID="{7769583f-2a8c-48fc-bde9-796970c974d8}" Genre="apps.normal"
    IsBeta="false" Author="Enflick" Publisher="Enflick" PublisherID="{e92c2343-c0db-499d-b210-
    80015cf2c821}">
    <IconPath IsRelative="true" IsResource="false">tile_applist_win8.png</IconPath>
  - <Capabilities>
      <Capability Name="ID_CAP_IDENTITY_DEVICE" />
      <Capability Name="ID_CAP_IDENTITY_USER" />
      <Capability Name="ID_CAP_MICROPHONE" />
      <Capability Name="ID_CAP_NETWORKING" />
      <Capability Name="ID_CAP_PHONEDIALER" />
      <Capability Name="ID_CAP_PUSH_NOTIFICATION" />
      <Capability Name="ID_CAP_SENSORS" />
      <Capability Name="ID_CAP_WEBBROWSERCOMPONENT" />
      <Capability Name="ID_CAP_CONTACTS" />
      <Capability Name="ID_CAP_MEDIALIB_AUDIO" />
      <Capability Name="ID_CAP_MEDIALIB_PHOTO" />
      <Capability Name="ID_CAP_MEDIALIB_PLAYBACK" />
      <Capability Name="ID_CAP_SPEECH_RECOGNITION" />
    </Capabilities>
```

FIGURE 12-25 **Applications have app manifest files that can reveal a lot about an installed application. This example shows a manifest file from TextNow, a popular chatting app.**

Another location in which apps may store data, but which is not recommended by Microsoft, is within the \SharedData path. WhatsApp, for example, stores backup information within the \SharedData\OEM\Public\WhatsApp\Backup path, and a complete backup of the messages and contacts can be recovered from the SQLite database. Also within the \SharedData\Comms folder is both a messaging and unistore folder, among other communication type apps. The unistore folder stores transient data from the store.vol ESE database that is shared among other apps or device functions. The messaging folder contains transient MMS files that are also available to other apps or device functions because of their location within the \SharedData folder. The \SharedData folder can be a good source for evidentiary data, because data deleted within an application can be retained in the \SharedData folder. Much like a Windows computer, the Windows Phone also contains a registry. In Figure 12-26, the examiner has full access to the familiar TypedURLs registry key within the Windows Phone Registry.



Name	Type	Data
url1	REG_SZ	http://m.adam4adam.com/?section=300
url2	REG_SZ	http://dayton.craigslist.org/search/cas?query=m4m...
url3	REG_SZ	http://papajohns.com
url4	REG_SZ	http://google.com/
url5	REG_SZ	http://www.adam4adam.com/
url6	REG_SZ	http://www.drjays.com/shop/G2-V3534-R340/sneak...
url7	REG_SZ	http://drjays.com/
url8	REG_SZ	http://drjays.com
url9	REG_SZ	http://bgclive.com/
url10	REG_SZ	http://adam4adam.com/
url11	REG_SZ	http://cincinnati.craigslist.org/search/cas?query=m4...
url12	REG_SZ	http://myvidster.com
url13	REG_SZ	http://bgclive.com
url14	REG_SZ	http://adam4adam.com

FIGURE 12-26 **Even Windows Phone registry files are accessible.**

An examiner must be prepared to analyze the Windows Phone file system in its entirety. Microsoft will continue to advance the device operating system, so continual updating of the various locations of artifacts will be necessary to recover the maximum amount of evidence. As with the BB10 device, an examiner must be diligent and recognize that the examination of these devices is time consuming, but the results can be fantastic.

Windows 10 Phone

As mentioned at the start of the Windows device section, I will not devote a lot of time to the Windows 10 device simply because of the current demand by examiners and the community support by tools. The update to the Windows 10 Mobile OS occurred in early 2015 and was an update to the Lumia line to include the 635, 730, and 830. Currently, all modern versions of the Lumia line support Windows 10, and almost all devices above 512MB will now support Windows 10 so long as they are moving from Windows 8 and 8.1.

The Microsoft Knowledge Base article (KB4090912), published in March 2018, outlines the current update and build information along with information on previous builds. However, how about the underlying file system? Information gathered on the layout of the file system shows that the main storage format is still the ESE database like previous versions of the Windows Phone OS. More importantly, the artifact table for Windows Phone 8/8.1 (Table 12-15) is still consistent for the Windows 10 device. The main storage point for PIM data will reside in the store.vol ESE database. Getting at the data is still an issue.

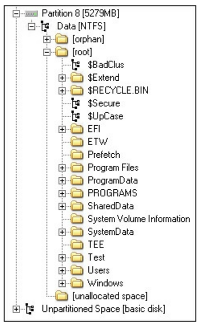
TABLE 12-15 **Compiled and Known Locations for Critical User Data Within a Windows Phone 8**

Artifact	Path	Considerations	Comments
Account Managers	\Users\WPCOMMSSERVICES\APPDATA\Local\BackupVols\	CommsBackup.xml	User login and passwords for various accounts
Call History	\Users\WPCOMMSSERVICES\APPDATA\Local\USERData\Phone	Call history table	Start and stop time, number, resolved contact name
Conversation	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Call logs, SMS messages, e-mail subjects	All items that have appeared on the device screen, in chronological order
Contact	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Names, e-mail, phone numbers, business addresses	Culmination of records from apps (such as Facebook, Twitter) and device contact list
Appointment	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Calendar appointments	Information in Unicode, ASCII date and time information
Attachments	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Attachments for various applications	Name of attachment, type of artifact
Message	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	MMS and SMS	Message content, numbers, date and time information
Notes	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Notes	Notes, content, date and time information
Tasks	\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol	Tasks	Tasks, content, date and time information
Internet Explorer	\Users\DefApps\APPDATA\INTERNETEXPLORER	Folders for INetCache, INetHistory	XML, HTML, and other files pertaining to Internet usage
Internet Explorer	\SharedData\InternetExplorer\Favorites		URL information for IE cached favorites
Apps	\Users\DefApps\APPDATA\Local\Packages\<GUID>\AC	Location for app certificates and additional security	Installed apps

Apps	\Users\DefApps\APPDATA\Local\Packages\	Location for Microsoft.<appname> apps app	Internet artifacts in INetCache, INetHistory
Apps	\Users\DefApps\APPDATA\<app GUID>\Local	Third-party apps	GUID or app name
Pictures	\Users\Public\Pictures	\Camera Roll\Saved Pictures\Screenshots	Photos from device, saved and shared
Videos	\Users\Public\Videos		Videos from device
Registry	MainOS Windows\systerm32\config	SAM, security, software, system	Information on typed URLs
App Identify	\PROGRAMS\<PACKAGE GUID>\Install	WMAppManifest.xml	Matches GUID with app name and app capabilities

If encryption has not been enabled on the device, the examiner can perform a chip-off and conduct decoding and analysis using Oxygen Forensic Detective and Cellebrite. Manual examination and validation of the ESE database files can be conducted using NirSoft ESEDatabaseView application (www.nirsoft.net/utils/ese_database_view.html). However, the Windows 10 device can be rooted as long as the device is unlocked and a complete raw file system exposed using tools from Windows Phone Internals (https://wpinternals.net).

As with an Android device, the Windows Phone Internals site will allow users to obtain root access, unlock bootloaders, and install custom ROMs. Note that the site creator and author state using the tools can have adverse effects on the device, however, to include the BSOD (Blue Screen of Death) common to many Microsoft products and the resetting of data. Use the tools with caution. Figure 12-27 shows the data partition of a Windows 10 device running 10.0.14393.0.



```
Partition 8 [5279MB]
  Data [NTFS]
    [orphan]
    [root]
      $BadClus
      $Extend
      $RECYCLE.BIN
      $Secure
      $UpCase
      EFI
      ETW
      Prefetch
      Program Files
      ProgramData
      PROGRAMS
      SharedData
      System Volume Information
      SystemData
      TEE
      Test
      Users
      Windows
    [unallocated space]
Unpartitioned Space [basic disk]
```

FIGURE 12-27 **Data partition in AccessData FTK Imager after raw extraction using Windows device internals**

One interesting change to the Windows 10 device versus previous versions is the addition of an "x," which is assumed to denote ten (10), to many of the internal filenames. The manifest files of previous apps within Windows devices had been named WMAppManifest.xml and are now simply AppxManifest.xml files. This naming convention of files and folders remains the only observed change to the artifacts as outlined earlier for

Windows Phone 8 devices. Again, with the limited number of Windows devices currently entering the forensic lab, the next addition of this book will likely not include an update of these devices.

Chapter Summary

No longer can an examiner rely on “tip of the iceberg data” when conducting an investigation. Even feature phones contain valuable data outside of the normal PIM data. Nevertheless, many examiners believe that data collected using an automated tool includes all the available data, even if the entire file system is extracted! After looking into the file system, however, most examiners learn that there is life outside of automation when conducting mobile device collections. The examiner must not only recognize that additional data can be recovered, but he or she must also be prepared to interpret the information as a software application would. By understanding the various formats and locations, an examiner can determine whether a deeper analysis of the mobile device file system should be considered. Armed with knowledge of the various formats of data for feature phones, UICC, BlackBerry, and Windows Phone devices, the examiner will be prepared to conduct a thorough file system examination.

BlackBerry 10 and Windows devices present new challenges to the mobile forensic community. The collection of their file systems is extremely challenging, and when the internal files and folder structure have never been documented, every examination becomes a research and development exercise. The information contained in this chapter for BlackBerry and Windows devices will most likely be the last update of these two dying mobile platforms and will not be included in the next version of the book. BlackBerry has already stopped producing devices and changed its OS to Android, and Microsoft has hinted that its mobile platform will remain but will technically be a mirror of the full desktop OS version. As always, the tables provided in this chapter are starting points in an examiner’s journey to uncover even more information from these devices.

As an examiner continues to grow in understanding that the analysis of a mobile device file system is not only for automated tools, the more discoveries within the BlackBerry and Windows file systems will be made, even if the technology is not as prevalent as other mobile operating systems. Hopefully, by using the tables and figures in this chapter, the examiner will be able to start off quickly, showing immediate results for any investigation or case.

13 Advanced iOS Analysis

In 2017, the International Data Corporation (IDC) reported that although Android and iOS mobile devices dominated the mobile market worldwide, it was the first year a decline (of 0.7 percent) in shipments was observed. In 2018, Android consistently has been leading, with an 84.8 percent share, with Apple at 15.1 percent (www.idc.com/promo/smartphone-market-share/os). That leaves little doubt that these two types of devices also dominate the workload of most mobile device forensic examiners. This chapter deals specifically with the examination of the iOS file system structure, which is available with a simple logical extraction or via a backup import created with iTunes.

Automated tools can be used to extract a lot of "tip of the iceberg" iOS device data, which satisfies many investigators. However, considering the total number of files that are available for examination on an iOS device, the number of files analyzed and presented using an automated tool pales in comparison to the number that can be viewed using a logical extraction. The problem lies with the examiner who believes in the "what they see is what they get" theory, who is satisfied with a report created by clicking Print at the end of an automated extraction. However, an iOS device is like no other smart device regarding how data is backed up, and using a simple logical extraction will include more information, such as critical databases, settings, and property lists in the backup—and this type of data is digital gold to the examiner.

Examining actual binary files from which the user data was parsed, along with additional files not traversed by an automated tool, offers the examiner the opportunity to perform a deep analysis of the iOS device's file system. This can include hashing and comparisons, verification and validation, and manual decoding and file carving—to name a few of the possibilities. By having the entire file system available, the examiner can perform a thorough examination without needing to conduct a non-invasive or an invasive physical extraction. This chapter outlines the critical file structures and formats that make up the meat of the iOS file system. The goal of this chapter is to encourage the examiner to move away from wholeheartedly trusting an automated tool to conducting an iOS mobile device examination.

Note

With today's iOS devices containing A5 and later chipsets, a non-jailbroken physical bit-by-bit collection using a USB is impossible, so a logical file system collection is the only available method. Because of this physical collection limitation, it is critical that the examiner conduct a logical file system extraction for all iOS mobile devices. Some sources report that a physical collection can be obtained using tools such as Elcomsoft iOS Forensic Toolkit, GrayKey by Grayshift, or Cellebrite's professional services CAIS program, but this is not entirely accurate. These tools cannot be used to perform a physical partition collection as they can with A4 chipsets; they simply enable collecting a jailbroken device's internal file system. Once the device is in a state in which a raw file system can be collected, most commercial tools, including UFED Touch 2, Oxygen Forensic Detective, and XRY, can collect the file system and artifacts.

This chapter will not discuss the hardware configurations for Apple devices, because many other publications provide information regarding the hardware and internal software versions. By referring to the

Apple Developer web site (https://developer.apple.com/), an examiner can consult the iOS Device Compatibility Reference page to obtain critical information on the internal components and devices. Table 13-1 shows links to detailed technical specs for various iOS devices.

TABLE 13-1 **Apple Support Technical Specification Links**

Device Family	Link to Specification Table
iPhone	https://support.apple.com/specs/iphone
iPad	https://support.apple.com/specs/ipad
iPod Touch	https://support.apple.com/specs/ipod
Apple TV	https://support.apple.com/specs/appletv
Apple Watch	https://support.apple.com/specs/applewatch

This chapter will focus on how the examiner can identify the variety of evidence located within the iOS file system, including data that can be used immediately in an investigation and information that will lend credibility to the processes used by automated tools. This information will help the examiner better understand how the extraction processes work and will provide information regarding documenting the procedure if the information is needed in court proceedings. The examiner will gain a better working understanding of the iOS device and a greater understanding of how to uncover and process the wealth of information contained within the iOS device file system.

The iOS File System

Apple mobile devices share some of the OS X foundation, but they also use a different framework, which means OS X applications will not run on iOS. Both are based on the UNIX file system, and although the structures of the file systems are similar between iOS and OS X, there are differences in the way each stores apps and user data. In an iOS mobile device, apps interact with the file system. The interactions are limited ("sandboxed") by design, so each has a container or a number of containers with specific roles (containers include the App Bundle, App Storage, and iCloud Data, as shown in Figure 13-1). iOS and OS X devices both use a type of hierarchical file system (HFS) prior to iOS 10.3 and High Sierra, respectively. An iOS device uses HFSX/HFS+ (Mac OS Extended), and OS X computers use HFS+. Apple indicates that the only difference between HFS+ and HFSX is that the latter contains case-sensitive filenames, which is commonly observed in iOS devices. However, in March 2017, iOS 10.3 received the update to Apple File System (APFS), and in September 2017, the macOS was also moved to APFS. Apple's sandboxing remains consistent, and Figure 13-1 is still relevant to the app layout, storage, and communication in APFS.

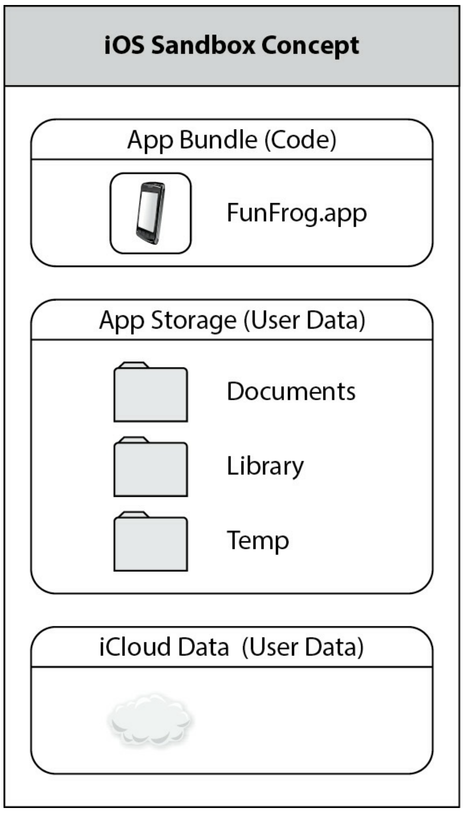


FIGURE 13-1 **The containers of the iOS sandbox**

As a course on iOS history for devices prior to iOS 10.3, mobile forensic tools conducted non-invasive physical collections of iPhone devices, and these solutions had to interpret the predicted binary HFS/HFSX file system. Early on, the forensic tools could interpret HFS+ file systems, but when the forensic tools looked for the telltale H+ around the 0x400 offset of the disk image, the process failed. Because the HFS/HFSX file system contains *HX* (Figure 13-2), the examiner had to change the *X* to a + in order for common computer forensic tools to mount the file system properly. The software would also have to negotiate and rebuild the file system from the raw disk image. As a result, to display and mount the directory structure properly, the solutions had to interpret the correct block size during collection and later decoding, which progresses in sizes of 512 bytes (for example, 4096, 8192). If the block size was not interpreted, the directory structure could not be properly formatted—which of course meant the examiner couldn't work with the file system. With today's tools, however, this is no longer an issue, because they all interpret the file system correctly and structure and display it properly to the examiner.

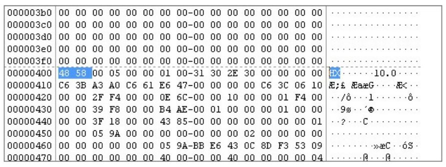
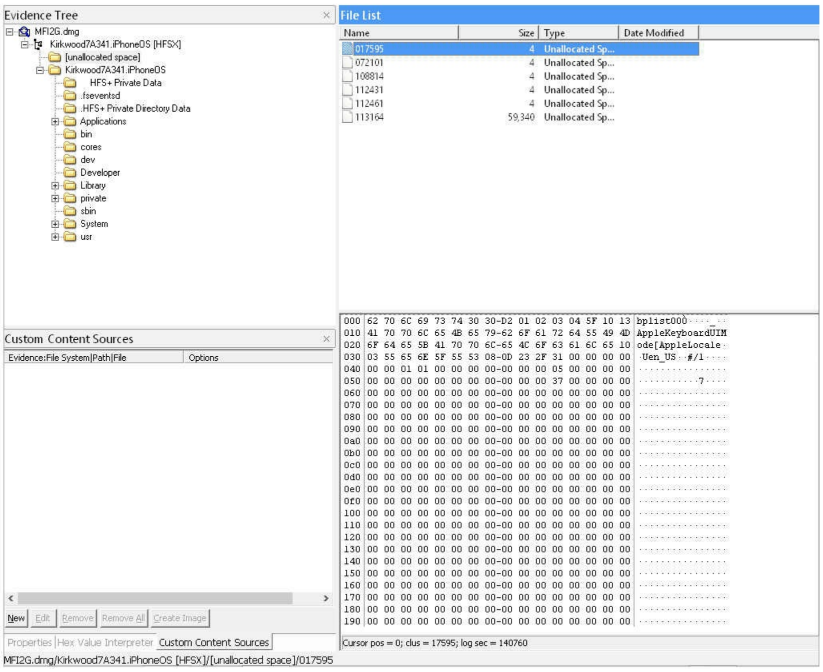

```
000003b0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000003c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000003d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000003e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000003f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
00000400 48 58 00 05 00 00 01 00-31 30 2E 30 00 00 00 00 HX······10.0····
00000410 C6 3B A3 A0 C6 61 E6 47-00 00 00 00 C6 3C 06 10 Æ;£ ÆaæG····Æ<··
00000420 00 00 2F F4 00 00 0E 6C-00 00 10 00 00 01 F4 00 ··/ô···l······ô·
00000430 00 00 39 F8 00 00 B4 AE-00 01 00 00 00 01 00 00 ··9ø··´®········
00000440 00 00 3F 18 00 00 43 85-00 00 00 00 00 00 00 01 ··?···C·········
00000450 00 00 05 9A 00 00 00 00-00 00 00 02 00 00 00 00 ················
00000460 00 00 00 00 00 00 05 9A-BB E6 43 0C 8D F3 53 09 ········»æC··óS·
00000470 00 00 00 00 00 00 40 00-00 00 40 00 00 00 00 04 ······@···@·····
```

FIGURE 13-2 **The *HX* marker is highlighted, indicating the HFS/HFSX format; the *X* had to be changed to a + during the early years of examining iPhones.**

Devices prior to those containing the A5 chip could undergo a non-invasive collection of the entire raw disk. This collection would yield both the system and user partitions within the iOS device. These partitions were much like the standard hard drive, with a directory structure, file slack, and unallocated areas. The unallocated space, or free area, within the iOS partition was a gold mine for deleted media, including images and videos. Granted, the videos had to be stitched back together because they were stored as separate image files, but recovering data from the unallocated space was still a dream come true with iOS devices, as shown in Figure 13-3.



Name	Size	Type	Date Modified
017595	4	Unallocated Sp...	
072101	4	Unallocated Sp...	
108814	4	Unallocated Sp...	
112431	4	Unallocated Sp...	
112461	4	Unallocated Sp...	
113164	59,340	Unallocated Sp...	

```
000 62 70 6C 69 73 74 30 30-D2 01 02 03 04 5F 10 13 bplist00Ò····_··
010 41 70 70 6C 65 4B 65 79-62 6F 61 72 64 55 49 4D AppleKeyboardUIM
020 6F 64 65 5B 41 70 70 6C-65 4C 6F 63 61 6C 65 10 ode[AppleLocale·
030 03 55 65 6E 5F 55 53 08-0D 23 2F 31 00 00 00 00 ·Uen_US··#/1····
040 00 00 01 01 00 00 00 00-00 00 05 00 00 00 00 00 ················
050 00 00 00 00 00 00 00 00-00 00 37 00 00 00 00 00 ··········7·····
060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
080 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0a0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0b0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
100 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
180 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
190 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
```

```
Cursor pos = 0; clus = 17595; log sec = 140760
MFI2G.dmg/Kirkwood7A341.iPhoneOS [HFSX]/[unallocated space]/017595
```

FIGURE 13-3 **An iPhone raw disk image with unencrypted unallocated space containing many deleted files**

This all ended with iOS 5, however, when Apple changed the way data was encrypted on disk along with data protection class keys and a file system key. With iOS 5, examiners and software developers alike were required to obtain the keybag and keys to decrypt the extracted user partition at the file level based upon file permissions to rebuild and ultimately analyze the extracted partition. Software vendors that had previously supported a non-physical collection of iOS devices quickly created modified versions of Jean Sigwald's code (https://code.google.com/p/iphone-dataprotection) that was in the wild. Soon user data was again visible to the examiner, but unfortunately the key needed for unallocated space was not available. As a result, when the unallocated space was extracted, it remained encrypted. The files are encrypted at the file level, and without having a valid file path and key and data protection key, the files cannot be decrypted.

Since then, new techniques have provided some benefits. For example, an article presented by Aaron Burghardt and Adam J. Feldman in 2008 (www.dfrws.org/2008/proceedings/p76-burghardt.pdf) discussed using the journal file within a macOS partition to locate and identify catalog file entries for deleted files, with limited results. This technique was extended by Jean-Baptiste Bédrune and Jean Sigwald within the iphone-dataprotection code on the google.com code repository using a Python script for iOS devices. As indicated by Sigwald, and also from personal experience, the journal file is extremely small, and only a small number of files can be recovered. To this day, this limitation means that the unallocated space is generally safe from an examiner's data carving tools.

When devices containing the A5 chip (iPad 2 was first, followed by the iPhone 4S) first hit the market, examiners realized they had a problem. Placing the devices into Device Firmware Update (DFU) mode still worked, but every attempt to use the same function with an automated tool no longer allowed for a collection. The iOS device, while still in DFU mode, did not accept the custom RAM disk, and examiners were blocked from acquiring the iOS device using this non-invasive physical technique. Although devices without the dreaded A5 chip were still being sold and updated, forensic software vendors continually had to explain to their customers that a non-invasive approach was not possible unless the device was jailbroken. This dashed the hopes and dreams of a lot of examiners, for now they thought all had been lost: the iOS device—the most widely sold mobile device at that time—was inaccessible to a forensic examiner. This mentality stormed through the community, and it is still a belief held by many examiners. However, this is not always the case, even with the devices of today running iOS 11 and the A11 chip along with the new APFS.

It was in iOS 4 that Apple introduced a feature that enables the user to encrypt the backup produced by iTunes within the application's settings. This setting had to be initiated within iTunes by the user checking the box to encrypt the iOS device backup and then entering a password for the backup and subsequent restore of that device. (To date, Apple has not made it mandatory to set a backup password in iTunes.) This password would be used to encrypt the backup, instead of using the hardware key of the iOS device. Using this method, the iTunes backup password is stored within the iOS device keychain database, and the BackupKeyBag is encrypted with the backup password and in the Manifest.plist. Because the BackupKeyBag is accessible within the backup, it is possible to decrypt the backup without involving the iOS device itself.

The encryption of the backup processes poses a problem to the mobile forensic examiner on several fronts. One, the encryption of the backup is not a setting that a user can set or disable on the actual iOS device, so an examiner cannot turn off the encryption on the device itself. Two, this setting is not on by default, but if a user checked the box and secured the backup with a password, the connected iOS device would then produce an encrypted backup when the Apple Mobile Backup service was initiated. This second condition posed another problem for the mobile forensic community when dealing with iOS devices. Because mobile forensic tools used the Apple Mobile Backup service to initiate a backup to the forensic computer, an encrypted backup was of no use. This roadblock was averted by Sigwald, who produced open source code that effectively decrypted the backup with the known password, again putting mobile forensic solutions back in business—so long as the examiner had the iTunes password that was used by the device owner. With that password, the mobile forensic solution would decrypt the backup and analyze the user's data. There was still a potential problem, however: What if the examiner did not know the iTunes password or the owner would not give it up? Commercial tools from Oxygen Forensics, Inc., Cellebrite, and Elcomsoft have the ability to "attack" the BackupKeyBag that is used to encrypt the backup to uncover the password. Also, instructions on attacking the Manifest.plist file using hashcat is available (http://irq5.io/2017/03/07/cracking-itunes-backup-passwords-with-hashcat/) and is currently used by many forensic examiners around the world with great success. This version of hashcat currently lists support for iTunes backups up to the current version, iOS 11x. In both the commercial tools and open source solutions, once the password is discovered, it can be used to decrypt the backup that was previously extracted.

Note

As noted by Apple, additional data not typically contained in a standard backup is included in an encrypted backup, including keychain, health data, and browsing history. Some commercial tools, such as Oxygen Forensic Detective, will force a backup to be encrypted when it is determined to have not been enabled by the device owner. This is a feature to uncover and include this valuable data to the forensic tool.

The inability of the examiner to conduct a proper collection of an Apple device if backup encryption was turned on was once a serious threat to forensic examinations, and still poses issues if proper tools are not available. The Apple hacking community had the same problem, so they trudged on in an effort to discover other methods of accessing an iOS device. Several groups began to publish open source tools that would allow not only a backup of the iOS device using iTunes, but also using the Apple File Connection (AFC) protocol used by iTunes to move files on and off and for device-level communication. Because iTunes is given access only to specific portions of the device (jailed environment), AFC is limited to the media area only. However, a jailbroken device, as the community discovered, can have all files available using the AFC protocol/backup service. The service was altered and installed to a jailbroken device and AFC2 was born, which enabled access to the complete file system of the iOS device.

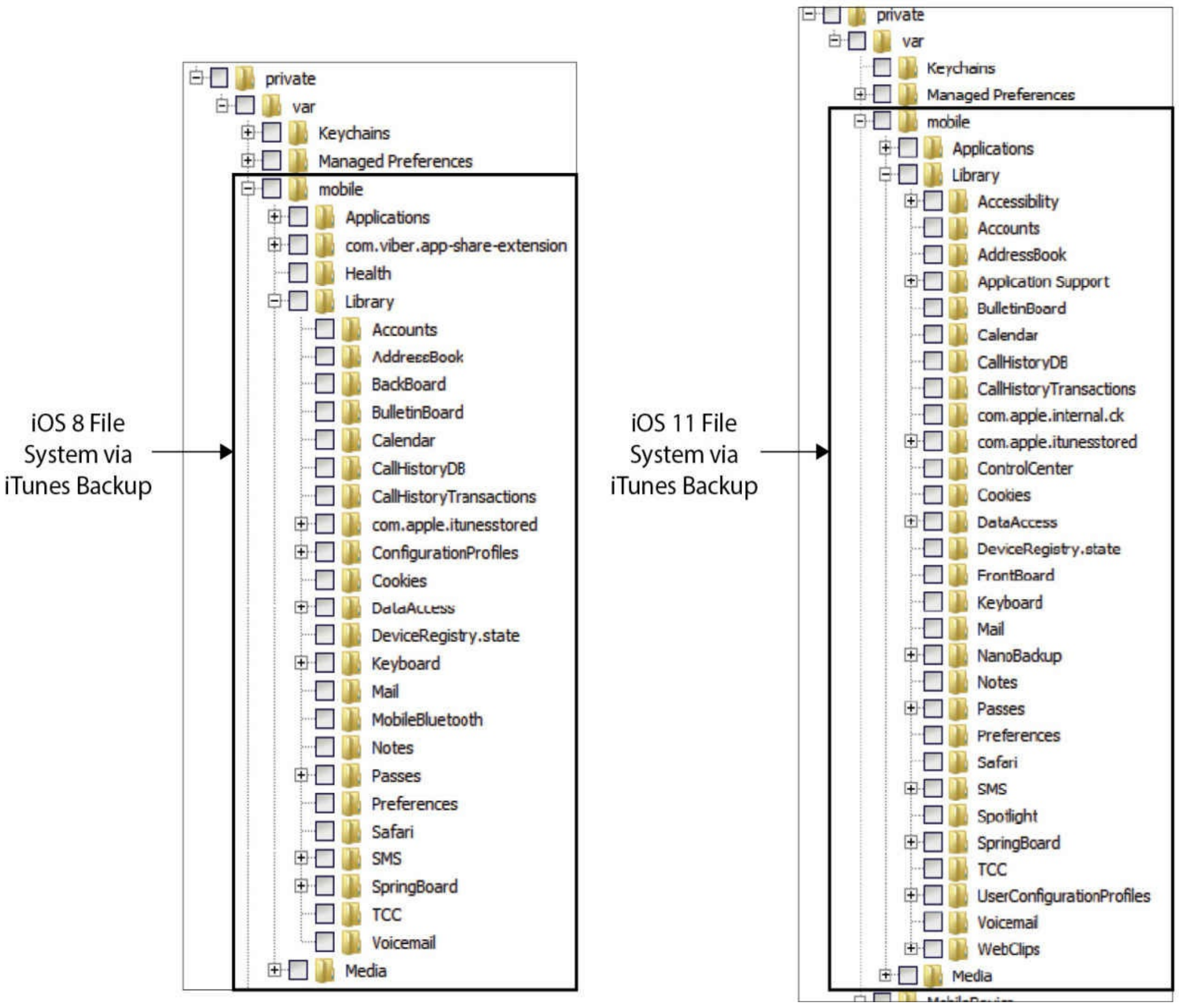
This method was not observed by the forensic community as viable, however, because of the implications often imposed by a jailbreak. But further research into the mobile device soon brought to light additional services such as com.apple.mobile.house_arrest and com.apple.mobile.file_relay. These services, primarily the file_relay service, had been developed by Apple as an internal testing mechanism for file transfer, but they had been a part of libimobiledevice code since 2009. This was not just any file transfer, but the transfer of user data–laden files, without the need for an iTunes connection. The type of data depends on the iOS version, but an iTunes backup was not invoked, and since Apple Mobile Backup was not initiated, the backup encryption was not triggered and any data transferred from the device would not be encrypted.

The house_arrest service enables access to the application folder and its contents. Mobile forensic solutions jumped on this new ability and began to incorporate this feature into their products. Examiners could then conduct a mobile device forensic exam on an iOS device even if the iTunes password was not known. Examiners were able to extract personal information manager (PIM) data and app data from these devices in a fully decrypted format. Mobile forensic vendors were back in business. Then, in 2015, an ex-employee of a mobile forensic company "outed" Apple on this "security flaw" and published how forensic companies were exploiting this "backdoor" and exposing customers' personal data. This self-serving revelation of a function already known by many researchers, and not new information by any stretch, effectively closed the door for law enforcement, enterprise, and corporate investigators if the file_relay service was going to be used to recover user data. Apple then released iOS 8, closing the door on the retrieval of PIM user data by any means other than using the built-in Mobile Backup function and limited the additional services. All iOS devices preceding OS version 8 still allow for connections to the house_arrest and file_relay services even if the iTunes password is not known. (Note that this self-serving individual later went to work for Apple as a researcher, most likely testing the capabilities of many of the forensic tools.)

Tip

A mobile forensic software solution must use the file_relay service to achieve a more robust file system collection, and not all mobile forensic software has this ability. The examiner should check with the software vendor before assuming that the software is capable of this function. UFED Physical Analyzer and Oxygen Forensic Detective both support this feature.

The logical collection of an iOS device without access to the additional services still contains an extraordinary amount of information. Examiners must be diligent in the recovery and discovery of the many files available within the iOS file system, as shown in Figure 13-4.


iOS 8 File System via iTunes Backup
iOS 11 File System via iTunes Backup

```
private
  var
    Keychains
    Managed Preferences
    mobile
      Applications
      com.viber.app-share-extension
      Health
      Library
        Accounts
        AddressBook
        BackBoard
        BulletinBoard
        Calendar
        CallHistoryDB
        CallHistoryTransactions
        com.apple.itunesstored
        ConfigurationProfiles
        Cookies
        DataAccess
        DeviceRegistry.state
        Keyboard
        Mail
        MobileBluetooth
        Notes
        Passes
        Preferences
        Safari
        SMS
        SpringBoard
        TCC
        Voicemail
      Media
```

```
private
  var
    Keychains
    Managed Preferences
    mobile
      Applications
      Library
        Accessibility
        Accounts
        AddressBook
        Application Support
        BulletinBoard
        Calendar
        CallHistoryDB
        CallHistoryTransactions
        com.apple.internal.ck
        com.apple.itunesstored
        ControlCenter
        Cookies
        DataAccess
        DeviceRegistry.state
        FrontBoard
        Keyboard
        Mail
        NanoBackup
        Notes
        Passes
        Preferences
        Safari
        SMS
        Spotlight
        SpringBoard
        TCC
        UserConfigurationProfiles
        Voicemail
        WebClips
      Media
```

FIGURE 13-4 **iOS logical file system view of iOS 8 and 11; many files and folders are still available for an examiner to analyze.**

iOS "Tip of the Iceberg Data"

The type of data collected by mobile forensic solutions from an iOS device is comparable to that collected on the majority of devices. There are, of course, small exceptions among solutions, but the collected data listed in Table 13-2, if available on the device, is always a part of a mobile forensic collection, barring a locked device, encrypted backup, or other impedance. The information in Table 13-2 can be used to identify the type of data and where the physical files are located so that further investigation can be conducted within the extracted file system when needed. Further investigation into the database files and other associated files can uncover usernames, passwords, and even deleted data.

TABLE 13-2 “Tip of the Iceberg Data” Commonly Collected by Automated Tools

User Data	File Type	Path in File System
Contacts	SQLite database	private/var/mobile/Library/AddressBook/ Addressbook.sqlitedb
Call logs	SQLite database	(prior to iOS 7) private/var/mobile/Library/ CallHistory/CallHistory.db (iOS 7) private/var/wireless/CallHistory/call_ history.db (iOS 8–11x) mobile/Library/CallHistoryDB/ CallHistory.storedata
SMS	SQLite database	private/var/mobile/Library/SMS/sms.db
MMS	SQLite database	private/var/mobile/Library/SMS/sms.db
Calendar	SQLite database	private/var/mobile/Library/Calendar/Calendar .sqlitedb
Notes	SQLite database	private/var/mobile/Library/Notes/notes.sqlite
Images	Individual .jpg files named IMG_<Sequence #>	private/var/ mobile/Media/DCIM/1XXAPPLE
Videos	Individual .mov files named IMG_<Sequence #>	private/var/ mobile/Media/DCIM/1XXAPPLE
Browser bookmarks	SQLite database	private/var/ mobile/Library/Safari/Bookmarks.db

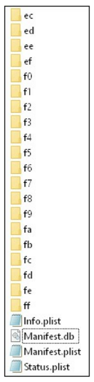
Table 13-2 contains only surface-level data; a wealth of additional data is contained within the file system. The next section discusses the file system arrangement, with major file and folder locations in a typical iOS file system.

Tip

With the introduction of iOS 8, devices that contained migrated data from iOS 7 have two database files in two different locations: the call_history.db for iOS 7 is stored at private/var/wireless/CallHistory, and CallHistory.storedata for iOS 8x is at mobile/Library/CallHistoryDB. Examiners should be aware that some tools do not obtain both databases. The call_history.db can contain historical calls before migration of the data and some duplicated calls also found in the iOS 8 CallHistory.storedata database. Because of this, the examiner should complete a manual inspection of these databases.

File System Structure

An iOS file system obtained by both open source and commercial mobile forensic tools relies primarily on the Apple Mobile Backup function. Because of this, the output by tools and interpretation of the file system are similar across the board. The files and folders have changed from the introduction of iOS 10, as shown in Figure 13-5. Commercial tools use the manifest.plist to obtain the file system structure and build the structure and display within the file system view of the tool. Also, within the manifest.db file, which is a SQLite database, the domain and path information can be obtained for the built-in apps for iOS devices. Note that most tools do not show the "true" path as documented in the plist file. For example, the true path for a third-party application would be /var/containers/Bundle/Application/, but commercial tools place the applications in /var/mobile/Application within the file browser view. Development teams explain that this is due to the legacy parsing across versions of iOS, and changing the path could break the way data is parsed.



```
ec
ed
ee
ef
f0
f1
f2
f3
f4
f5
f6
f7
f8
f9
fa
fb
fc
fd
fe
ff
Info.plist
Manifest.db
Manifest.plist
Status.plist
```

FIGURE 13-5 **The iOS Backup format was changed at iOS 10 to include a manifest.db and folders containing the backed-up bundles.**

This section represents the major "landmarks" that an examiner may see within the collected files and folders, along with some additional folders that might be discovered if the file_relay and house_arrest functions are used by the forensic software solution. This function was utilized prior to iOS 8 and has resurfaced with the use of a jailbreak or tools such as Grayshift's GrayKey device.

App Data

Figure 13-1 showed the sandbox concept and partitions. This storage method helps the examiner realize that valuable app data is contained within the Documents, Library, and Temp folders within the App Storage area. Table 13-3 outlines evidence locations based on the storage of user data and app data. These folders are stored directly under the main application folder within the iOS device file system. The application folder for iOS is located at /private/var/mobile/Applications, and the folder for iOS 8x is at /private/var/mobile/Containers/Data/Application. This path is consistent up to iOS 11 as well, but as discussed previously, a commercial tool will modify the true path to conform to parsers. Table 13-3 can help steer the examiner to the most likely location for digital artifacts within the application folders. Note that the application name within a raw iOS file system is a globally unique identifier (GUID), known as a BundleID, and that the GUID can change upon application updates; the examiner should not rely on the GUID to identify a specific application. Most mobile forensic software titles will translate the GUID to the bundle identifier if the manifest is used and available.

TABLE 13-3 **Major Landmarks Within the App File and Folder Structure Important for an Investigator**

Folder	Data Description
Documents	This folder contains user-generated content, where a developer would place data that can be accessible and exposed to the app user and data that the app cannot re-create. Often contains the SQLite databases used and available to the user along with plist files detailing user settings.
Documents/ Inbox	This folder is used often by app developers to enable the app to access files that the app opened from outside requests. An example would be a mail application that places e-mail attachments into the directory. The app can read and delete the files, but it cannot create or edit files within this directory. This folder is backed up by iTunes if used.
Library	The app uses this top-level directory to store data that it does not want to expose to the user; it is not advisable to store user data here that can be directly interacted upon by the user. This folder uses a subdirectory structure, and the app developer can create custom folders. Any folder, with the exception of the Caches subdirectory, is backed up by iTunes.
Library/ Application Support	This folder contains subfolders and files used by the app for function support. This is where the examiner can find advertisement support, database files supporting features, and additional app settings. This area should be examined, especially with browsers and apps not using SQLite databases. Chrome for iOS, for example, stores bookmarks, history, and other information within the Application Support/Google/Chrome/Default folder.
Library/Caches	The Caches folder is often a gold mine. The folder is obtained either via the house_arrest or file_relay service on a jailbroken device. This folder is not available in an iTunes backup. The folder is described later in the "App Caches" section of this chapter.
Library/Cookies	This subfolder contains the Cookies.binarycookies file, which stores the persistent session cookies used by the app. This file is discussed later in this section.
Library/ Preferences	This subfolder contains the application preferences. For example, mobile Safari contains all the search strings within the com.apple.mobilesafari .plist file. Other property lists in this folder provide user login details and application details.
tmp	This is temporary storage for the app. This data seldom persists unless the app developer is not purging the data, which is generally not the case. However, if the app is still running in the background when an examiner completes a collection, temporary data could be available, including media, text, and settings files. If available, the examiner should investigate.

Note

Apple development documentation insists that developers use the reverse DNS format for apps: the web site extension (.com, .net, .tv, .info, .co.uk, and so on), followed by the web site name, ending with the application name (such as com.oxygen.oxyagent). Tying an application to its bundle identifier is often as easy as looking into the manifest property list in the backup that has been created.

The Documents folder within the app file system structure contains, as mentioned in Table 13-3, the user-facing data, which is available to the user via the app. Here the examiner will find SQLite databases the app developer used for persistence, along with folders containing stored files. The Documents folder and its subdirectories store data that the user of the device can add, delete, and alter via the app. It can also contain additional custom folders for the storage of images, configuration settings, and SQLite files. The storage file for the app does not have to be a SQLite database and can be proprietary binary files whose format depends on the app developer's preference for storage, but the method must conform to the Apple SDK. The Documents folder can also contain property lists, XML, text files, and more. Analysis of custom binary files, text files, or property lists may involve building a custom parser, since most mobile forensic solutions cannot account for every file type within an iOS system. The examiner should understand as well that there are limitations with the folder availability, file availability, and file-level encryption—that is, this folder and associated files are not always available.

In iOS 5 and earlier versions, the app developer was not allowed to exclude folders from the backup, but this limitation was lifted in iOS 5.1 and has continued to the current iOS version (11). The app developer can use `NSURLIsExcludedFromBackup` or `kCFURLIsExcludedFromBackupKey` file system properties to exclude files or directories from backups. This can be extended to system-defined directories, but doing this is not recommended by Apple. However, the examiner will recognize that some apps do not have a Documents folder, or they may have the Documents folder but do not use SQLite databases or other user-defined data. This would indicate that the app developer used the exclusion property, and reliance on the Caches folder will be critical.

The /Library/Cookies folder contains the Cookies.binarycookies file that stores cookies for the particular app; a similar folder is found in most app file systems. A cookie is used to identify the user on a particular web site or web server. Cookies are of particular interest to the examiner because they can show that the user of the iOS device accessed particular web sites using that app. The Cookies folder can also hold a second Cookies.binarycookies file that is appended with a `-corrupt` flag. The `-corrupt` flag is rumored to be a failed authentication or corrupted file and is marked as such, so a new file is created. The Cookies.binarycookies file and the file marked `-corrupt` both can be decoded using some helpful Python scripts.

A script from satishb3 (www.securitylearn.net) was used as the basis to create the code that follows to operate with Python 3. It has been modified to decode, parse, and output to a file called CookieMonster.csv. The examiner can view the decoded data within a text reader or spreadsheet application. To run the script, the examiner can use any forensic solution that incorporates a Python scripting tool or an open source tool such as PyScripter. The examiner must have Python 3 installed to run the script and will use Cookies.binarycookies as the argument. Information and descriptions are incorporated throughout the following code.

We start by importing the needed Python modules that will be used in the script:

```
import sys
from struct import unpack
from io import BytesIO
from time import strftime, gmtime
import csv
```

This will be the file the script will parse. The `sys_argv`, shown next, is the file following the name of the script if the script is run from the command line or a system argument variable (for example, `mypythonprogram.py Cookies.binarycookies`, where `Cookies.binarycookies` is the `sys_argv`).

```
FilePath= sys.argv[1]

try:
      binary_file=open(FilePath,'rb')

except IOError as e:
      print ('File Not Found :'+ FilePath)
      sys.exit(0)
```

This portion of the code will look at the header of the file for `'cook'` to determine whether it is a valid cookie file per this script:

```
file_header=binary_file.read(4)
file_header=file_header.decode()

if str(file_header)!='cook':
      print ("Not a Cookies.binarycookies file")
      sys.exit(0)
```

Next, the script begins to read through the file to determine the number of pages (cookies) that will exist in the binary file and reports them back to the script as a variable to be used later. Once the pages are determined, the script will know how many cookies are within the file.

```
num_pages=unpack('>i',binary_file.read(4))[0]
page_sizes=[]
for np in range(num_pages):

      page_sizes.append(unpack('>i',binary_file.read(4))[0])
pages=[]
for ps in page_sizes:
      pages.append(binary_file.read(ps))
```

Next, the conversions of each page (cookie) occurs. The cookie name, creation date, expiry date, and domain the cookie is from are all decoded and extracted from the binary page:

```
for page in pages:

    page=BytesIO(page)
    page.read(4)
    num_cookies=unpack('<i',page.read(4))[0]
    cookie_offsets=[]
    for nc in range(num_cookies):
    cookie_offsets.append(unpack('<i',page.read(4))[0])

    page.read(4)
    cookie=b''

    for offset in cookie_offsets:
            page.seek(offset)
            cookiesize=unpack('<i',page.read(4))[0]
            cookie=BytesIO(page.read(cookiesize))
            cookie.read(4)

flags=unpack('<i',cookie.read(4))[0]
cookie_flags=''
            if flags==0:
                cookie_flags=''
            elif flags==1:
                cookie_flags='Secure'
            elif flags==4:
                cookie_flags='HttpOnly'
            elif flags==5:
                cookie_flags='Secure; HttpOnly'
            else:
                cookie_flags='Unknown'

            cookie.read(4)

            urloffset=unpack('<i',cookie.read(4))[0]
            nameoffset=unpack('<i',cookie.read(4))[0]
            pathoffset=unpack('<i',cookie.read(4))[0]
            valueoffset=unpack('<i',cookie.read(4))[0]
            endofcookie=cookie.read(8)
```

Dates listed are MAC times with an epoch of January 1, 2001, so an adjustment is made. The date depends upon the application examined, so this value should be analyzed upon output.

```
            expiry_date_epoch= unpack('<d',cookie.read(8))[0]+978307200
            expiry_date=strftime("%a, %d %b %Y ",gmtime(expiry_date_
epoch))[:-1]
            create_date_epoch=unpack('<d',cookie.read(8))[0]+978307200
            create_date=strftime("%a, %d %b %Y ",gmtime(create_date_
epoch))[:-1]
```

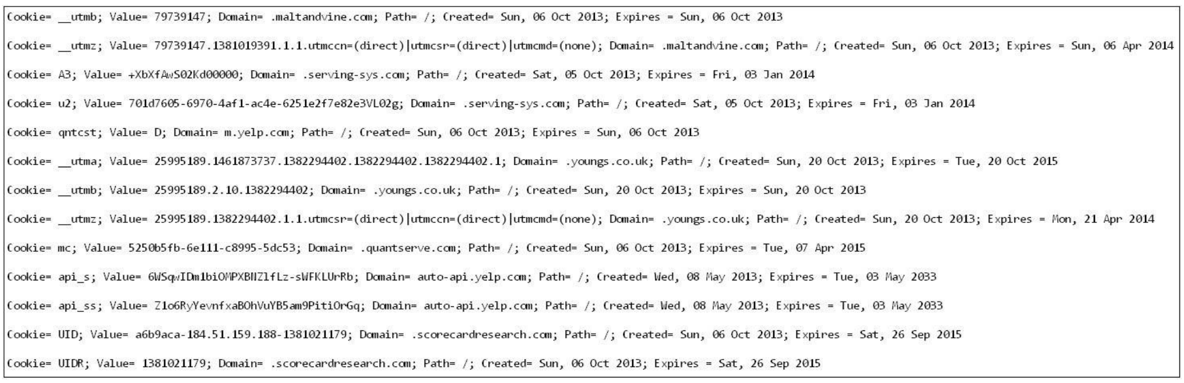
The data from each page is systematically extracted, starting with the URL, then the name, the path, and finally the actual cookie value:

```
cookie.seek(urloffset-4)
url=b''
u=cookie.read(1)

while unpack('<b',u)[0]!=0:
    url=url+bytes(u)
    u=cookie.read(1)
url=url.decode('utf-8')

cookie.seek(nameoffset-4)
name=b''
n=cookie.read(1)
while unpack('<b',n)[0]!=0:
    name=name+bytes(n)
    n=cookie.read(1)
name=name.decode()

cookie.seek(pathoffset-4)
path=b''
pa=cookie.read(1)
while unpack('<b',pa)[0]!=0:
    path=path+bytes(pa)
    pa=cookie.read(1)
path=path.decode()

cookie.seek(valueoffset-4)
value=b''
va=cookie.read(1)
while unpack('<b',va)[0]!=0:
    value=value+bytes(va)
    va=cookie.read(1)
value=value.decode()
```

The CookieMonster.csv file is created at the location shown in the following code listing. The information is systematically written to the open file that loops until the final page (cookie) has been located and printed, and the dates obtained are also added to the string. The file is then closed along with the Cookies.binarycookies file.

```
            f = open('C:\\CookieMonster.csv', 'a')
            f.writelines ('Cookie= {0};Value= {1};Domain= {2};Path= {3};
Created= {4}; Expires= {5} \n\n'.format(name,value,url,path,create_
date,expiry_date))
f.close()
binary_file.close()
```

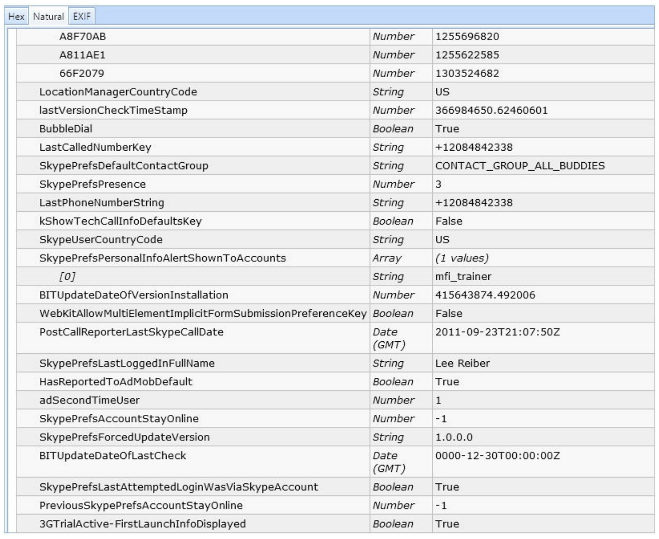
Once the process completes, the examiner can retrieve the file from the location specified within the code; here it is at C:\. This script output as shown in Figure 13-6 includes some valuable data for the examiner, especially if web history is of interest in an investigation.



```
Cookie= __utmb; Value= 79739147; Domain= .maltandvine.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Sun, 06 Oct 2013

Cookie= __utmz; Value= 79739147.1381019391.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none); Domain= .maltandvine.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Sun, 06 Apr 2014

Cookie= A3; Value= +XbXfAwS02Kd00000; Domain= .serving-sys.com; Path= /; Created= Sat, 05 Oct 2013; Expires = Fri, 03 Jan 2014

Cookie= u2; Value= 701d7605-6970-4af1-ac4e-6251e2f7e82e3VL02g; Domain= .serving-sys.com; Path= /; Created= Sat, 05 Oct 2013; Expires = Fri, 03 Jan 2014

Cookie= qntcst; Value= D; Domain= m.yelp.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Sun, 06 Oct 2013

Cookie= __utma; Value= 25995189.1461873737.1382294402.1382294402.1382294402.1; Domain= .youngs.co.uk; Path= /; Created= Sun, 20 Oct 2013; Expires = Tue, 20 Oct 2015

Cookie= __utmb; Value= 25995189.2.10.1382294402; Domain= .youngs.co.uk; Path= /; Created= Sun, 20 Oct 2013; Expires = Sun, 20 Oct 2013

Cookie= __utmz; Value= 25995189.1382294402.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); Domain= .youngs.co.uk; Path= /; Created= Sun, 20 Oct 2013; Expires = Mon, 21 Apr 2014

Cookie= mc; Value= 5250b5fb-6e111-c8995-5dc53; Domain= .quantserve.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Tue, 07 Apr 2015

Cookie= api_s; Value= 6WSqwIDm1biOMPXBNZ1fLz-sWFKLUrRb; Domain= auto-api.yelp.com; Path= /; Created= Wed, 08 May 2013; Expires = Tue, 03 May 2033

Cookie= api_ss; Value= Z1o6RyYevnfxaBOhVuYB5am9PitiOrGq; Domain= auto-api.yelp.com; Path= /; Created= Wed, 08 May 2013; Expires = Tue, 03 May 2033

Cookie= UID; Value= a6b9aca-184.51.159.188-1381021179; Domain= .scorecardresearch.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Sat, 26 Sep 2015

Cookie= UIDR; Value= 1381021179; Domain= .scorecardresearch.com; Path= /; Created= Sun, 06 Oct 2013; Expires = Sat, 26 Sep 2015
```

FIGURE 13-6 **Output and review of the file created by running the script against the cookies file from com.yelp.yelpiphone**

The /Library/Preferences folder is included in just about all app directory structures and contains the settings specific to the user and the app. The property list to examine is the plist file assigned to the bundle ID (for example, com.google.Drive.plist and com.skype.skype.plist, as shown in Figure 13-7). This plist file will often contain the user login name, last settings, last searches, date of last use, and other settings that could be critical to an examination. Settings files for many of the apps are also located in the Caches folder. Note that many of the property lists in the Preferences folders have been limited in the data to identify the user from iOS 10 to the current iOS version (11x).



A8F70AB	*Number*	1255696820
A811AE1	*Number*	1255622585
66F2079	*Number*	1303524682
LocationManagerCountryCode	*String*	US
lastVersionCheckTimeStamp	*Number*	366984650.62460601
BubbleDial	*Boolean*	True
LastCalledNumberKey	*String*	+12084842338
SkypePrefsDefaultContactGroup	*String*	CONTACT_GROUP_ALL_BUDDIES
SkypePrefsPresence	*Number*	3
LastPhoneNumberString	*String*	+12084842338
kShowTechCallInfoDefaultsKey	*Boolean*	False
SkypeUserCountryCode	*String*	US
SkypePrefsPersonalInfoAlertShownToAccounts	*Array*	*(1 values)*
[0]	*String*	mfi_trainer
BITUpdateDateOfVersionInstallation	*Number*	415643874.492006
WebKitAllowMultiElementImplicitFormSubmissionPreferenceKey	*Boolean*	False
PostCallReporterLastSkypeCallDate	*Date (GMT)*	2011-09-23T21:07:50Z
SkypePrefsLastLoggedInFullName	*String*	Lee Reiber
HasReportedToAdMobDefault	*Boolean*	True
adSecondTimeUser	*Number*	1
SkypePrefsAccountStayOnline	*Number*	-1
SkypePrefsForcedUpdateVersion	*String*	1.0.0.0
BITUpdateDateOfLastCheck	*Date (GMT)*	0000-12-30T00:00:00Z
SkypePrefsLastAttemptedLoginWasViaSkypeAccount	*Boolean*	True
PreviousSkypePrefsAccountStayOnline	*Number*	-1
3GTrialActive-FirstLaunchInfoDisplayed	*Boolean*	True

FIGURE 13-7 **A plist file for com.skype.skype.plist showing critical information on last calls, last login, and more**

The examiner could also encounter the /Library/WebKit folder or SQLite databases with the .localstorage extension. At iOS 8, changes began to surface and have continually advanced regarding how to store data locally. Think of it this way: if an app has an "offline" mode or makes data available to the user when it is offline, it needs storage. Mobile device apps, however, need a more efficient way to interact with the data. In iOS 11x, four different types of storage are used: Web Storage (localstorage), Web SQL Database, Indexed Database (IndexedDB), and File System. Data storage in previous versions was within the Caches folder, using the Web Storage format and with Safari using the Web SQL Database format. All the different types of client-side storage described so far use SQLite databases to store the transactional data, with the exception of File System, which uses JavaScript Object Notation (JSON). In the current versions of iOS, the databases are stored at /Library/WebKit/WebsiteData in folders IndexedDB, LocalStorage, and WebSQL, as shown in Figure 13-8. However, the data within each folder is laid out differently than before, with different parent databases.

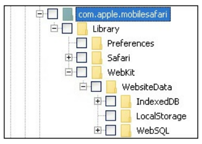
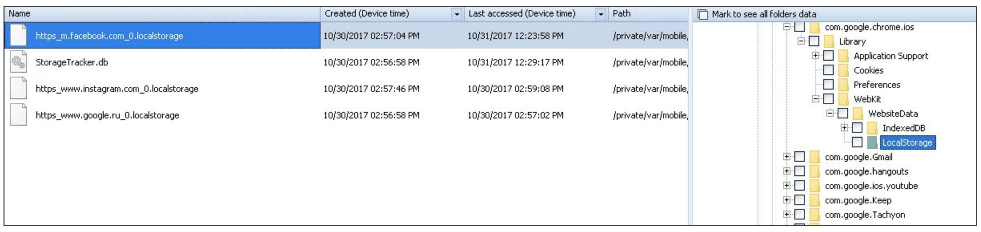


FIGURE 13-8 **Format under the WebKit folder within the iOS system, showing the respective folders that house the data from web sites locally**

Within the storage locations are a parent database and children files, folders, or both. The IndexedDB folder's parent is the IndexedDB.sqlite3 file that stores the path to the data stored offline by web sites visited by the user. The IndexedDB.sqlite3 file is created and stored within each site's folder. The LocalStorage folder's parent is the StorageTracker.db file that contains the name of the site as well as the path to the .localstorage file associated to the web site. Each of the .localstorage files are also SQLite databases and contain web site key-value pairs, and all are contained in the root of LocalStorage. The WebSQL folder contains the familiar Databases.db file, which contains not only the path but also the username for the web site. Within the WebSQL folder are the folders that contain the databases for the sites that are referenced in the Databases.db file.

Overall, the files and folders in the WebKit folder use the WebKit framework, which enables an application to use a built-in browser to display and store web content and settings. In doing this, the app will often cache the entire database that is used for that particular instance or site visited using the app. If the app uses an internal browser, or it is a browser type app, when the app launches the browser and navigates to a site such as Facebook, for example, the app will cache the mobile web site's browser settings and/or local data for that site (see Figure 13-9). The site name is part of the filename along with the extension .localstorage. Full web pages, searches, and other useful data can be located within the WebKit folder and respective files in the IndexedDB, LocalStorage, and WebSQL folders. It has been shown within the iOS device that Safari will utilize the WebSQL folder and the Databases.db file and associated web site referenced databases, while other apps will use the LocalStorage folder and the .localstorage file. Both hold good data for the investigator, but data in the WebSQL folder is considerably more robust because of additional tables and fields.



Name	Created (Device time)	Last accessed (Device time)	Path
https_m.facebook.com_0.localstorage	10/30/2017 02:57:04 PM	10/31/2017 12:23:58 PM	/private/var/mobile,
StorageTracker.db	10/30/2017 02:56:58 PM	10/31/2017 12:29:17 PM	/private/var/mobile,
https_www.instagram.com_0.localstorage	10/30/2017 02:57:46 PM	10/30/2017 02:59:08 PM	/private/var/mobile,
https_www.google.ru_0.localstorage	10/30/2017 02:56:58 PM	10/30/2017 02:57:02 PM	/private/var/mobile,

```
Mark to see all folders data
com.google.chrome.ios
    Library
        Application Support
        Cookies
        Preferences
        WebKit
            WebsiteData
                IndexedDB
                LocalStorage
com.google.Gmail
com.google.hangouts
com.google.ios.youtube
com.google.Keep
com.google.Tachyon
```

FIGURE 13-9 **Chrome, a browser application, uses WebKit functionality to offer the user a built-in web browser that caches data from each site within its directory structure.**

Note

The WebKit folder is still available within the main /private/var/mobile/Library folder for iOS, but it is seldom populated because most applications use the WebKit folder within their app folder structure.

Installed Apps

An examiner may often be curious as to whether an app had been installed and later removed from an iOS device. The Launch Services Daemon Identifiers file, com.apple.lsdidentifiers.plist, located at private/var/db/lsd/, indicates applications that have been installed on the device and launched by the lsd service and provides the actual vendor ID GUID assigned to the app(s). (Note that this vendor ID is not the product GUID; that would be found in the manifest file or the /Library/Application folder.) By comparing the list of apps in the com.apple.lsdidentifiers.plist to that in the Manifest.plist file and iOS file system, the examiner can determine whether the app once existed and has since been removed. (This has been verified in several tests in which an application has been installed, run, and subsequently deleted. Upon analyzing the backup, the bundle ID was still visible within the com.apple.lsdidentifiers.plist file but was no longer referenced within the file system or Manifest.plist.) The db folder and subsequent lsd folder is no longer available within the iTunes backup on any device later than iOS 9. The lsd folder and property list are still available within a raw file system extraction via jailbreak or the Grayshift GrayKey solution. However, several files are of importance with installed apps.

The key-value store, or kvs.sqlitedb, is located at private/var/mobile/Library/com.apple.itunesstored/ and within the downloaded-apps key is a binary plist. Extracting and displaying this property list will show all the apps that have been downloaded to the device from the iTunes Store. Furthermore, the KnownAccounts key in the same file lists all the names and e-mail addresses of users who have accessed the store. Within the same directory as the kvs.sqlitedb is the itunesstored2.sqlitedb. This file holds date and time data of the third-party apps on the device, which could be a great piece of evidence. Diving deeper into the apps, in the TCC.db file, located at private/var/mobile/Library/TCC, the user can identify the permissions that have been granted to the app, and using the clients.plist file from root/Library/Caches/location/ will identify the applications using location services along with great date and time information.

App Caches

An app on the iOS device, like any other program, uses an area on disk to store volatile and temporary data. As described earlier, app development dictates that data that should not be available to the user be stored in the Library folder and its subdirectories. The Caches folder is one of those subdirectories in which the app stores volatile data (such as web sites, media from web sites, and site-specific settings), and it is typically visible within the application only during the session. The folder can include children files in the folder root, but it can also store additional subfolders and files that are used by the app. The folders and files under the Caches folder are up to the app developer to create; Apple does give some direction and guidance regarding iCloud (https://developer.apple.com/icloud/documentation/data-storage/index.html)—but again, this is ultimately determined by the developer. Some consistencies within the Caches folder can steer the examiner, however, and within these files and folders is a wealth of information waiting to be recovered.

The Caches folder is located under the Library folder, which is a subfolder of the application main folder (see Figure 13-10). (The examiner should not confuse this with the Caches folder found under /private/var/mobile/Library.) Because the Caches folder is not contained within an iOS standard iTunes backup, the mobile forensic solution must use the house_arrest and file_relay services for recovery and collection to obtain a raw file system. Depending on the tool used, the bundle information is stored under the mobile/Containers/Bundle/Application path, with the data for the apps, to include the Caches folder, within the mobile/Containers/Data/Application folder. If the Caches folder is not visible after the extraction within these areas, the examiner should consider attempting extraction with a secondary tool to ensure that this important data is recovered.

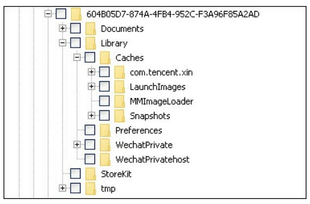


FIGURE 13-10 **The Caches folder expanded for com.tencent.xin**

Within the Caches folder, an app can store many custom folders, but apps also have some commonalities, as discussed in the following sections.

Bundle ID Folder

A good place to start is to look for a folder named after the bundle ID (such as com.<company><app name>). This does not necessarily have to be the same bundle ID that describes the app, but that is usually the case. Additional folders can also be in a bundle ID format—these are apps within the app, used by the main app and often advertising caches for images and other web details, analytics, and app crash details. The subfolder of importance will typically be the one with the bundle ID that is consistent with that of the actual app. If the bundle ID within the file browser of the mobile forensic tool is recognizable by its reverse-DNS format, this is a standard iTunes backup format normally. If a GUID is found at the path var/mobile/Containers/Bundle/Application/, that represents all the apps along with their code; at var/mobile/Containers/Bundle/Data/Application is another GUID, which represents the raw iOS file system as it is identified on the device. If the latter is observed, the following information on the Cache.db will pertain to the investigation, since at iOS 9 the availability on a standard iTunes backup was removed.

If reviewing a raw file system from either a jailbroken device or a device obtained using a tool such as GrayKey, the examiner needs to locate that bundle ID folder at var/mobile/Containers/Bundle/Data/Application. Once at the GUID, navigate to the Library/Caches/<Bundle ID> folder within. There, a Cache.db and sometimes an ApplicationCache.db will be available. The Cache.db file can also be located in another subfolder within the parent folder and typically contains five tables, with three of particular significance: cfurl_cache_response, cfurl_cache_blob_data, and cfurl_cache_receiver_data (Figure 13-11). The Cache.db file represents the data that the app has received from an outside source (such as the server or the Internet) and holds in cache for speed of loading if needed again.

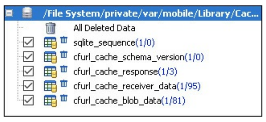
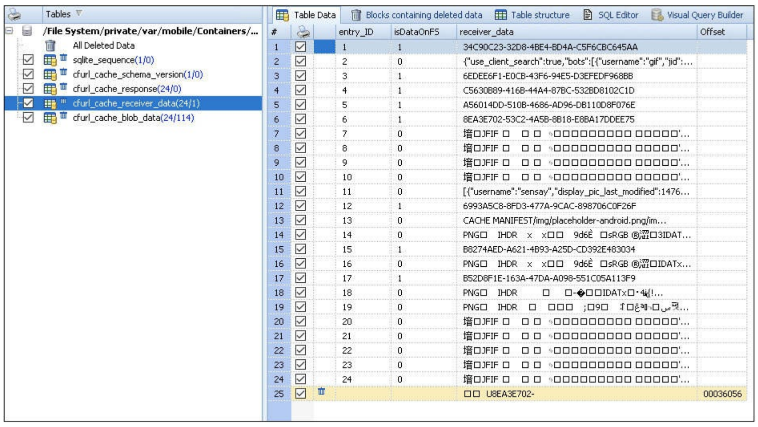

```
/File System/private/var/mobile/Library/Cac...
    All Deleted Data
☑   sqlite_sequence(1/0)
☑   cfurl_cache_schema_version(1/0)
☑   cfurl_cache_response(1/3)
☑   cfurl_cache_receiver_data(1/95)
☑   cfurl_cache_blob_data(1/81)
```

FIGURE 13-11 **Cache.db expanded in Oxygen Forensic Detective's SQLite Viewer**

The cfurl_cache_response table contains the data that was requested and the response, including the URL and the time of the request. The cfurl_cache_blob_data table contains BLOB (Binary Large Object) data with the response from the server. The cfurl_cache_receiver_data table contains the received data from the server in response to the server via the cfurl_cache_response table (Figure 13-12). Matching these tables can yield information that is not stored anywhere else. It is important for the examiner to pay attention to the isDataOnFs field within the cfurl_cache_receiver_data table. If the table contains a *1*, the data is stored within another folder on the iOS device within the Caches folder. A BLOB of 4096 bytes or larger will be stored locally and assigned a GUID, and all other files will be stored within the database and represented by a *0*. The files and the locations that are cached locally are covered in the next section.



Tables

/File System/private/var/mobile/Containers/...
- All Deleted Data
- sqlite_sequence(1/0)
- cfurl_cache_schema_version(1/0)
- cfurl_cache_response(24/0)
- cfurl_cache_receiver_data(24/1)
- cfurl_cache_blob_data(24/114)

Table Data | Blocks containing deleted data | Table structure | SQL Editor | Visual Query Builder

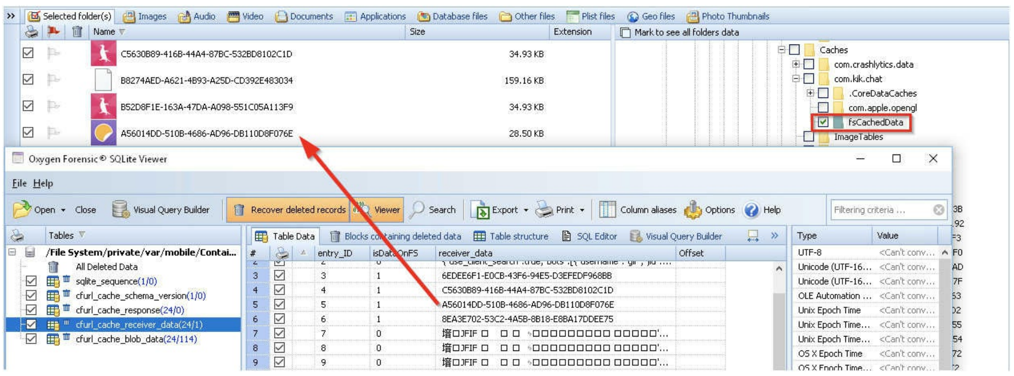
#			entry_ID	isDataOnFS	receiver_data	Offset
1	✓		1	1	34C90C23-32D8-4BE4-BD4A-C5F6CBC645AA	
2	✓		2	0	{"use_client_search":true,"bots":[{"username":"gif","jid":...	
3	✓		3	1	6EDEE6F1-E0CB-43F6-94E5-D3EFEDF968BB	
4	✓		4	1	C5630B89-416B-44A4-87BC-532BD8102C1D	
5	✓		5	1	A56014DD-510B-4686-AD96-DB110D8F076E	
6	✓		6	1	8EA3E702-53C2-4A5B-8B18-E8BA17DDEE75	
7	✓		7	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
8	✓		8	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
9	✓		9	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
10	✓		10	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
11	✓		11	0	[{"username":"sensay","display_pic_last_modified":1476...	
12	✓		12	1	6993A5C8-8FD3-477A-9CAC-898706C0F26F	
13	✓		13	0	CACHE MANIFEST/img/placeholder-android.png/im...	
14	✓		14	0	PNG□ IHDR x x□□ 9d6È □sRGB ®□3IDAT...	
15	✓		15	1	B8274AED-A621-4B93-A25D-CD392E483034	
16	✓		16	0	PNG□ IHDR x x□□ 9d6È □sRGB ®□IDATx...	
17	✓		17	1	B52D8F1E-163A-47DA-A098-551C05A113F9	
18	✓		18	0	PNG□ IHDR □ □-�□□IDATx□·4!...	
19	✓		19	0	PNG□ IHDR □ □□□ ;□9□ □...	
20	✓		20	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
21	✓		21	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
22	✓		22	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
23	✓		23	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
24	✓		24	0	揞□JFIF □ □ □ □□□□□□□□□□ □□□□□'...	
25	✓	🗑			□□ U8EA3E702-	00036056

FIGURE 13-12 **The Cache.db summary information shows that 24 records are still available in three tables within the database, and there are 114 recovered records.**

Note

It is extremely important that the examiner remember that the Cache.db file will be in the root of the bundle ID, but the structure may not conform to the schema described. App developers can, and many do, create schemas that will suit their particular applications, especially if the app does not use external servers or Internet resources.

The Caches folder also contains a subfolder used to store file data, including JPEGs, GIFs, PNGs, HTML files, JSON files, and others. This folder, fsCachedData, is located under the bundle ID in the Caches folder and is used to store files identified in the cfurl_cache_receiver_data table by a TRUE (1) Boolean value in the isDataOnFs field. Kik Messenger, for example, a widely used chat app, prescribes storing information locally, without a database, and this information is available in the fsCachedData folder. The files contained in the folder are directly referenced in the cfurl_cache_receiver_data table within the Cache.db file, as shown in Figure 13-13. The files within the Kik fsCachedData folder range from JPEG, to PNG, to JSON files. If the examiner matches the information contained within this table against the cfurl_cache_response table, he or she has all of the files that are referenced along with a timetable of the event, as shown in Figure 13-14. This can provide an incredible amount of information for any investigation when dealing with applications that do not have a main database. In fact, this might be the only data that can be recovered!



Selected folder(s) | Images | Audio | Video | Documents | Applications | Database files | Other files | Plist files | Geo files | Photo Thumbnails

		Name	Size	Extension
✓		C5630B89-416B-44A4-87BC-532BD8102C1D	34.93 KB	
✓		B8274AED-A621-4B93-A25D-CD392E483034	159.16 KB	
✓		B52D8F1E-163A-47DA-A098-551C05A113F9	34.93 KB	
✓		A56014DD-510B-4686-AD96-DB110D8F076E	28.50 KB	

Mark to see all folders data

- Caches
  - com.crashlytics.data
  - com.kik.chat
    - .CoreDataCaches
    - com.apple.opengl
    - fsCachedData
  - ImageTables

Oxygen Forensic® SQLite Viewer

File Help

Open | Close | Visual Query Builder | Recover deleted records | Viewer | Search | Export | Print | Column aliases | Options | Help | Filtering criteria ...

Tables

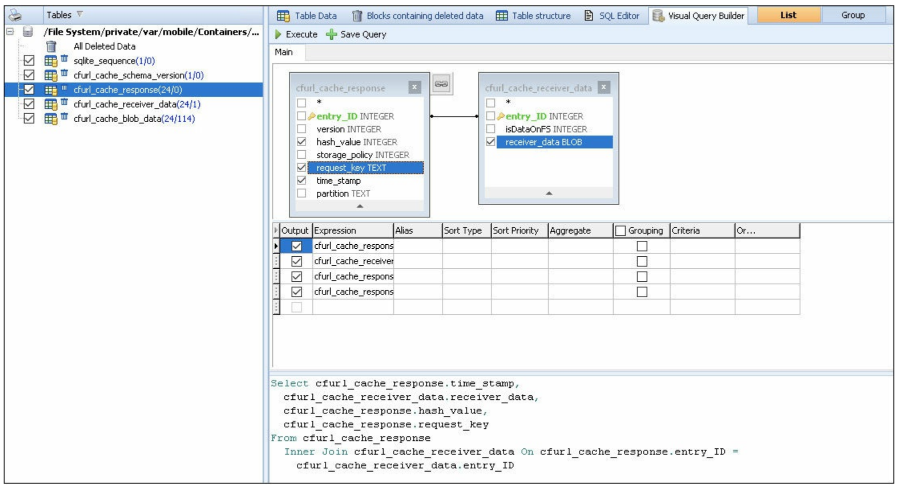
/File System/private/var/mobile/Contai...
- All Deleted Data
- sqlite_sequence(1/0)
- cfurl_cache_schema_version(1/0)
- cfurl_cache_response(24/0)
- cfurl_cache_receiver_data(24/1)
- cfurl_cache_blob_data(24/114)

Table Data | Blocks containing deleted data | Table structure | SQL Editor | Visual Query Builder

#		entry_ID	isDataOnFS	receiver_data	Offset
2	✓	2	0	{"use_client_search":true,"bots":[{"username":"gif","jid":...	
3	✓	3	1	6EDEE6F1-E0CB-43F6-94E5-D3EFEDF968BB	
4	✓	4	1	C5630B89-416B-44A4-87BC-532BD8102C1D	
5	✓	5	1	A56014DD-510B-4686-AD96-DB110D8F076E	
6	✓	6	1	8EA3E702-53C2-4A5B-8B18-E8BA17DDEE75	
7	✓	7	0	揞□JFIF □ □ □ □□□□□□□□□ □□□□□'...	
8	✓	8	0	揞□JFIF □ □ □ □□□□□□□□□ □□□□□'...	
9	✓	9	0	揞□JFIF □ □ □ □□□□□□□□□ □□□□□'...	

Type	Value
UTF-8	<Can't conv...
Unicode (UTF-16...	<Can't conv...
Unicode (UTF-16...	<Can't conv...
OLE Automation ...	<Can't conv...
Unix Epoch Time	<Can't conv...
Unix Epoch Time...	<Can't conv...
Unix Epoch Time...	<Can't conv...
OS X Epoch Time	<Can't conv...
OS X Epoch Time...	<Can't conv...

FIGURE 13-13 **The fsCachedData folder in Kik Messenger gives the investigator a folder full of GUID named files that can be matched to the GUID within the cfurl_cache_receiver_data table.**



/File System/private/var/mobile/Containers/...
- All Deleted Data
- sqlite_sequence(1/0)
- cfurl_cache_schema_version(1/0)
- cfurl_cache_response(24/0)
- cfurl_cache_receiver_data(24/1)
- cfurl_cache_blob_data(24/114)

cfurl_cache_response: *, entry_ID INTEGER, version INTEGER, hash_value INTEGER, storage_policy INTEGER, request_key TEXT, time_stamp, partition TEXT

cfurl_cache_receiver_data: *, entry_ID INTEGER, isDataOnFS INTEGER, receiver_data BLOB

Output	Expression	Alias	Sort Type	Sort Priority	Aggregate	Grouping	Criteria	Or...
✓	cfurl_cache_respons					☐		
✓	cfurl_cache_receiver					☐		
✓	cfurl_cache_respons					☐		
✓	cfurl_cache_respons					☐		

```
Select cfurl_cache_response.time_stamp,
  cfurl_cache_receiver_data.receiver_data,
  cfurl_cache_response.hash_value,
  cfurl_cache_response.request_key
From cfurl_cache_response
  Inner Join cfurl_cache_receiver_data On cfurl_cache_response.entry_ID =
    cfurl_cache_receiver_data.entry_ID
```

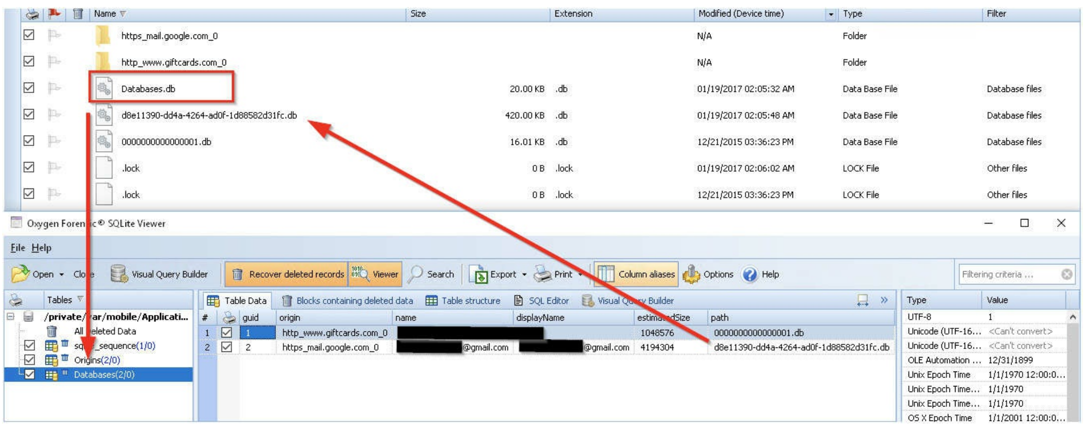
FIGURE 13-14 **When a SQL query is built to JOIN the cfurl_cache_response table and cfurl_cache_receiver_data, the data can be matched to date/time information and paths.**

Note

The Caches folder is no longer available within an iTunes backup as has been documented here, and it is available only within a raw file system extraction.

Additional Folders Under Caches

The examiner should be diligent in the examination of the files directly under the Caches folder, because a wealth of information is available. In an iTunes backup, the following will be located under the /WebKit/WebsiteData/WebSQL folder beneath the app folder and within the Applications folder. Database files within the Caches and WebKit folders are generally located in the root of the bundle ID folder, and these files are important to the examiner because many apps—most notably Google apps—store database files in this location. One very important database to look for under the Safari app (com.apple.mobilesafari) is Databases.db. This database contains a table with the name of the Gmail account and the size and path of the database file, all located within the Caches and WebKit/WebsiteData/WebSQL folders (see Figure 13-15).



Name	Size	Extension	Modified (Device time)	Type	Filter
https_mail.google.com_0			N/A	Folder	
http_www.giftcards.com_0			N/A	Folder	
Databases.db	20.00 KB	.db	01/19/2017 02:05:32 AM	Data Base File	Database files
d8e11390-dd4a-4264-ad0f-1d88582d31fc.db	420.00 KB	.db	01/19/2017 02:05:48 AM	Data Base File	Database files
0000000000000001.db	16.01 KB	.db	12/21/2015 03:36:23 PM	Data Base File	Database files
.lock	0 B	.lock	01/19/2017 02:06:02 AM	LOCK File	Other files
.lock	0 B	.lock	12/21/2015 03:36:23 PM	LOCK File	Other files

Oxygen Forensic® SQLite Viewer

/private/var/mobile/Applicati...
- All Deleted Data
- sqlite_sequence(1/0)
- Origins(2/0)
- Databases(2/0)

#	guid	origin	name	displayName	estimatedSize	path
1	1	http_www.giftcards.com_0	[redacted]		1048576	0000000000000001.db
2	2	https_mail.google.com_0	[redacted]@gmail.com	[redacted]@gmail.com	4194304	d8e11390-dd4a-4264-ad0f-1d88582d31fc.db

Type	Value
UTF-8	1
Unicode (UTF-16...	<Can't convert>
Unicode (UTF-16...	<Can't convert>
OLE Automation ...	12/31/1899
Unix Epoch Time	1/1/1970 12:00:0...
Unix Epoch Time...	1/1/1970
Unix Epoch Time...	1/1/1970
OS X Epoch Time	1/1/2001 12:00:0...

FIGURE 13-15 **Gmail Databases.db file pointing to the location of associated databases that store the user's e-mail and other data**

When the examiner locates these files, they can be correlated with the databases table and opened within a SQLite database viewer. Granted, the data from Gmail is only a snippet of the e-mail, but often this is the only information that is needed to assist in an investigation. This type of format clearly resembles the methods used by WebKit, discussed earlier in the chapter. The Databases.db file points to the local storage of offline cached e-mail.

Numerous custom folders could also be stored in the Caches folder on raw file systems and Library/Application Support or Library folders in iTunes backup extractions, and those that appear consistently are covered in this section. If an app is heavy into image display, either via online sources or server, folders typically named Media, ImageCache, or another derivative of "image" are often visible. These folders contain GUIDs that are referenced within a database within the main Caches folder. Most media has been moved out of the Caches folder unless it is only temporary data that can be purged and is accessible only with a raw file system, not iTunes backups. However, the media that is used, transmitted, and stored for apps can be located under the Library folder most frequently in a folder called Media. Safari does like to store thumbnails of the pages that are currently open, much like a snapshot, since data entered by the user is still visible. This information is available on an iTunes backup at /private/var/mobile/Applications/com.apple.mobilesafari/Library/Safari/Thumbnails/ and within a raw file system extraction along with a large cache of temporary Internet browsing data.

Also within the Caches folder, and only available on a raw file system extraction, is a Snapshots folder that contains another identical bundle ID equal to the main app bundle ID. Within this folder are the PNG files that have been saved to represent the app in different perspectives when the app was moved to the background. When an iOS app is minimized, it can be animated by first taking a screen capture of the currently active screen, creating a PNG, and shrinking the PNG file. The iOS saves this PNG file to the Snapshots folder.

A PNG file is created for each of the app's orientations (such as landscape, vertical, and inverted) when the app was minimized. Multiple PNG files can be found in this folder with an actual snapshot of data that might not be recoverable anywhere else on the device. Examiners have recovered e-mail, text messages, conversations, and more that the user had permanently deleted, but the snapshots remained, lending critical evidence to a case.

Additional File System Locations

App files are important to every forensic examination, but settings files and additional database files will also aid an examiner in analyzing an iOS device. These files are available within the file system using a standard logical collection unless otherwise noted. How these files are viewed is determined by the file type. This section discusses several main locations within the iOS file system that the examiner can quickly review for information. Some of the folder locations shown in Table 13-4 have already been covered and are mentioned; however, some of the paths contain several files and folders that are of significance to an examiner and are covered in detail.

TABLE 13-4 **iOS File System Parent Landmarks for Significant Evidentiary Files**

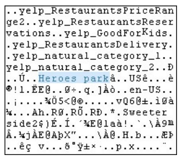
Folder	Data Description
private/var/mobile/	Contains Application, Library, and Media folders. With iOS 8 and to the current release (11x), the actual Containers folder is directly under mobile with subfolders Bundle, Data, and Media with a raw file system extraction after jailbreak or GrayKey.
private/var/preferences/SystemConfiguration	Contains the main user configuration files for Apple apps and services; these files are discussed in this chapter.
private/var/root/Library/Caches/locationd	Contains the consolidated.db, which includes geofence information, but with new devices there is generally limited information. The clients.plist contains every application that has used GPS location services and whether the app is authorized.
private/var/root/Library/Preferences	Contains many property lists that pertain to backups and phone services. This should not be confused with the app-level /Library/Preferences. Important plist files are outlined in a separate section.
private/var/wireless/Library	Contains Databases and Preferences. The CallHistory subfolder used to be here as well and stored historic and current data in a database, call_history.db, that is a leftover from iOS 7. This is visible only on an iOS 8 and later device if it has been restored with iOS 7 data.

Within the private/var/mobile/Library folder is not only the "tip of the iceberg data" contained within databases, but other significant databases in which an examiner may locate additional information:

- **/Library/Accounts/Accouts3.sqlite** This folder has usernames of application accounts, including date and time information.
- **/Library/Caches** This folder used to contain vast numbers of property lists and files that were cached by Apple Services, and these may be available if advanced services such as file_relay can be run; with a standard iTunes backup, the folder is not accessible with most forensic tools. If the device has been jailbroken or accessed via other means, this folder will be available with cached data and should be examined by the investigator.
- **/Library/Calendar/Extras.db** This folder contains current alarms and those that had been set but are no longer used on the device.
- **/Library/UserConfigurationProfiles/Truth.plist** At iOS 9, the path changed from /ConfigurationProfiles to /UserConfigurationProfiles. At iOS 10, and continued to the current version (11x) other files, PublicEffectiveUserSettings.plist and EffictiveUserSettings.plist, were added and include much the same data. Both file folders contain a key, forceEncrpytedBackup, that indicates whether the backup will be encrypted when calling an iTunes backup. Also, at iOS 10 and current (11x) versions, these files are found in

/private/var/mobile/Applications/systemgroup.com.apple.configurationprofiles/Library/ConfigurationProfi

- **/Library/Keyboard/** This folder contains any keyboard that the device has been set to use and its associated cache. The keyboard cache is identified as <language of keyboard>-dynamic-text.dat. This text file includes everything the user has typed in areas identified as "text fields." This recording takes place to assist with the iOS device's ability to auto-correct, which allows words to be suggested as the user types. If an app has a field marked "secure," the data that is typed into that field will not be populated in the dynamic-text.dat file. Secure fields typically include passwords, credit card numbers, and other personal data. Another file is the lexicon.dat file. This, much like the dynamic-text.dat, is created on a per-language basis and each contains words that can be made from contacts, custom dictionary, and other sources. The dynamic-text.dat and lexicon.dat file are no longer available to an examiner unless a raw file system is recovered in a jailbreak or other methods. However, there are several CloudUserDictionary.sqlite files that can be matched to the application that used text replacements and the programmed text replacement phrases and words that are available in the iTunes backup files.
- **/Library/Mail/Recents.db** This database file contains recent e-mail and SMS addresses that include name, e-mail address, and phone number along with the dates accessed and the last date accessed. This database is a gold mine for an examiner because the file grows indefinitely, and if a user deletes a contact, the information in Recents.db remains, regardless of the deletion.
- **/Library/MobileBluetooth/com.apple.MobileBluetooth.ledevices.paired.db** This database file is found only within iOS versions 8 to iOS 10x and contains the devices that have been paired with the iOS device. In iOS 11x, the location has changed to /private/var/mobile/Applications/systemgroup.com.apple.bluetooth/Library/Database/.
- **/Library/MobileBluetooth/com.apple.MobileBluetooth.ledevices.other.db** This database file is found only within iOS 11x devices and is quite interesting, because it shows the devices that are not paired but have been in range.
- **/Library/Maps/Bookmarks.plist** This binary plist contains bookmarks the user has added to the internal iOS mapping app. This file, for iOS 8, is located at /Containers/Application folder and is no longer stored in the /Library/Maps location.
- **/Library/Maps/GeoBookmarks.plist** This property list for iOS 9 to the current version (11x) contains a Key MSPBookMark that contains the bookmarks (Values) in hexadecimal representation that are easily converted to ASCII to reveal searches.
- **/Library/Maps/FailedSearches.mapsdata** This proprietary file can be read as text to gather searches conducted using the internal app for iOS or an app that uses the internal app. An example is Yelp: If Yelp uses the internal iOS map, any searches in Yelp that failed to find a location could be stored here, as shown in Figure 13-16. In iOS 8, this file is located within the /Containers/Application folder and is no longer stored in the /Library/Maps location. In iOS 10 to the current version (11x), the path is /private/var/mobile/Applications/com.apple.Maps/Library/ and is a binary plist file with embedded hexadecimal data.



```
..yelp_RestaurantsPriceRan
ge2..yelp_RestaurantsReser
vations..yelp_GoodForKids.
.yelp_RestaurantsDelivery.
.yelp_natural_category_1..
yelp_natural_category_2..Ð
..Ú..Heroes parkâ..USê...è
®!1.ËE@..Ø÷.q.]Àò..en-US..
.¡....¾Ö5<@©.....vQ6@±.iØà
¾...Ah.RØ.RÕ.RÐ.*.Sweeter
side2$)É.Í.´¾E@laà!.`.\À9m
Â.¼jÀE@AþX"...\À@.H.b...Æþ
..êç v...ð°ÿ±×·..p.x....¨.
```

FIGURE 13-16 **A search for "Heroes park" did not find the location and was deposited into the FailedSearches file.**

- **/Library/Maps/History.mapsdata** This proprietary file can be read as text to gather a history of routes used within the internal iOS Maps app. This file also contains data from apps that use the Maps program within the app. If a user used Yelp to locate a restaurant, for example, information about the restaurant, the directions, and even reviews would be located within this file. In iOS 8, this file is located within the /Containers/Application folder and is no longer stored in the /Library/Maps location. In iOS 10 to 11x, the path is /private/var/mobile/Applications/com.apple.Maps/Library/ and is a binary plist file with embedded hexadecimal data. The path /private/var/mobile/Applications/com.apple.Maps/Library/GeoHistory.mapsdata is the location of the SearchResults.mapsdata of older iOS versions.
- **/Library/Maps/SearchResults.mapsdata** This proprietary file lists the searches conducted within the internal iOS Maps app and also any app that uses the Maps app within the app. This file for iOS 8 is located within the /Containers/Application folder and is no longer stored in the /Library/Maps location. This has been deprecated and is now GeoHistory.mapsdata. The path /private/var/mobile/Applications/com.apple.Maps/Library/Pins.mapsdata is now available in the current version of iOS and shows the locations in plist format that a pin was dropped by the user.

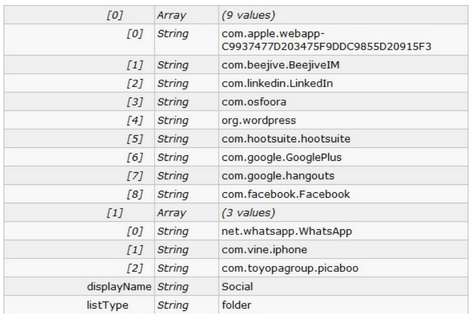
The Preferences folder contains many property list files. Only a small number have proved to hold valuable information for many examiners and are listed here:

- **/Library/Preferences/com.apple.conference.plist** This property list identifies the FaceTime settings for the device.
- **/Library/Preferences/com.apple.identityservices.idstatuscache.plist** This property list can also be a gold mine of information for deleted messages in FaceTime, iMessage, or e-mail. Identity services confirm the validity of a user's credentials as it travels across the ESS (Enterprise Shared Services) of Apple. Within this file are phone numbers and e-mail addresses along with a UNIX date of the lookup for clear credentials. Content is not included, but simply having the metadata of the occurrence is enough.
- **/Library/Preferences/com.apple.ids.service.com.apple.ess.plist** This file holds the accounts capable of authenticating on the Apple ESS and includes the AppleID, VettedAliases, Aliases, and LoginAs information. This key contains phone numbers and e-mail addresses of the device user.
- **/Library/Preferences/com.apple.ids.service.com.apple.madrid.plist** This is the identity services property list for iMessage. It contains e-mail addresses and phone numbers that have been used and are currently being used to access the ESS, along with the user's Apple ID.
- **/Library/Preferences/com.apple.ids.service.com.apple.private.alloy.<SERVICENAME>.plist** These files contain the user login information and service information for the various apple apps and services (for

example, FaceTime, iMessage, camera, and so on).

- **/Library/Preferences/com.apple.imservice.FaceTime.plist** This file identifies the credentials used for FaceTime.
- **/Library/Preferences/com.apple.imservice.iMessage.plist** This file identifies the credentials used for iMessage.
- **/Library/Preferences/ com.apple.itunesstored.plist** This file contains the date and time of updates to each specific app on the device.
- **/Library/Preferences/ com.apple.languageassetd.plist** This file contains the dictionaries, and even conversion dictionaries, that are set up on the device, including specialties, such as emojis.
- **/Library/Preferences/com.apple.locationd.plist** This file identifies the apps that have been accessed and are currently using location-based services. This file is still available in the current version of iOS (11x), but it does not identify the applications. The file discussed previously, Clients.plist, contains this valuable information. This file is located at /private/var/root/Library/Caches/locationd/.
- **/Library/Preferences/com.apple.madrid.plist** This file can be used to determine whether the user has ReadReceiptsEnabled set to on or off. When set to on, this indicates whether the receiver of the iMessage has read the message or not.
- **/Library/Preferences/com.apple.Maps.plist** This file identifies the last map location searched for by longitude and latitude and the date and time along with great data about the Apple maps. This was moved to /private/var/mobile/Applications/com.apple.Maps/Library/Preferences/ at iOS 8 and is still located here with release 11x. The MapsActivity Key and MapsActivityTimestamp Key are important artifacts.
- **/Library/Preferences/com.apple.MobileBluetooth.devices.plist** This file identifies the paired Bluetooth devices along with the MAC address and the LastSeenTime, indicating when the device was last observed by the iOS device. It was deprecated in iOS 9 and moved to Applications, as described earlier in the chapter.
- **/Library/Preferences/com.apple.MobileBluetooth.ledevices.plist** This file is another Bluetooth listing of identified Bluetooth Low Energy (BLE) devices that have been observed by the device. This settings file can be used by third-party apps that use BLE functions and are not compatible with classic Bluetooth protocols. A random MAC address is created, and a LastSeenTime indicates when the object was last observed by the iOS device. It was deprecated in iOS 9 and moved to Applications, as described earlier in the chapter.
- **/Library/Preferences/com.apple.mobile.ldbackup.plist** This file identifies the last backup date, if the backup was encrypted, and also if backing up to iCloud is enabled.
- **/Library/Preferences/com.apple.mobileipod.plist** This file identifies the music and, more importantly, the last media item played on the iOS device.
- **/Library/Preferences/com.apple.mobilephone.plist** This file identifies the last phone number that was displayed on the dialer screen. This will be visible even if the phone call was not initiated or completed.
- **/Library/Preferences/com.apple.mobilephone.settings.plist** This file identifies the call-forwarding number, if one is enabled, and other general settings. This was deprecated prior to iOS 8.

- **/Library/Preferences/com.apple.mobilephone.speeddial.plist** This file identifies favorites from the contact list. This was deprecated prior to iOS 8.
- **/Library/Preferences/com.apple.mt.<ACTION>.plist** These files are unique to current devices (iOS 11x) and include Killed, lastLaunch, and LaunchDelay. Killed lists the apps that were cancelled or killed while running, lastLaunch will give the date and time of the app's last launch, and LaunchDelay will give in seconds the delay in the application starting. The *mt* in the NDS naming appears to be representative to the Apple URL pattern for its web site and the new devices: see www.apple.com.
- **/Library/Preferences/com.apple.nanoweatherprefs.plist** This file identifies the selected city and country, along with the latitude and longitude, for the weather app.
- **/Library/Preferences/com.apple.osanalytics.addaily.plist** This file identifies the data usage, along with a date, for each app on the device. This could be good evidence to show how much the app was being used or if it was used on a certain day.
- **/Library/Preferences/com.apple.preferences.datetime.plist** This file identifies the preference of state and country for the device date and time.
- **/Library/Preferences/com.apple.preferences.network.plist** This file identifies the state of the network—for example, Wi-Fi was on or off.
- **/Library/Preferences/com.apple.preferences.plist** This file has some great information for the investigator on installed keyboards, number of apps, iCloud username, and much more.
- **/Library/Preferences/com.apple.TelephonyUtilities.plist** This file has some great information for the investigator on the apps that have access to the telephone. This would primarily the VOIP type apps such as Messenger, Telegram, Skype, and so on. Also, this file holds the key lastKnownFaceTimeCallerID, which can be extremely significant to your case.
- **/Library/Preferences/com.apple.vehiclePolicy.DNDMode.plist** This file can be used to see if the do not disturb setting for vehicles was enabled and also other significant details.
- **/Library/Preferences/com.apple.watchlistd.plist** This file lists the Apple Watch along with descriptors.
- **/Library/Preferences/UITextInputContextIdentifiers.plist** This file holds numbers from FaceTime and calls, as well as the keyboard that is used when interacting with these numbers (such as language and keyboard type).
- **/Library/SpringBoard/IconState.plist** This property list lays out the icons and folders for the apps that are currently displayed to the user on the iOS device. In Figure 13-17, a Social folder was created, and the icons directly above are the apps contained within this folder. Several keys within the property list can be important to the examiner. The key button bar at the top of the list identifies the apps that are located along the bottom of the main screen and accessible from any page on the iOS device. The key listType identifies a folder, and directly above is displayName, which identifies the name of the folder. The apps contained in the folder are directly above the displayName, starting with [0] and identified by the array number until reaching the displayName key.



[0]	*Array*	*(9 values)*
[0]	*String*	com.apple.webapp-C9937477D203475F9DDC9855D20915F3
[1]	*String*	com.beejive.BeejiveIM
[2]	*String*	com.linkedin.LinkedIn
[3]	*String*	com.osfoora
[4]	*String*	org.wordpress
[5]	*String*	com.hootsuite.hootsuite
[6]	*String*	com.google.GooglePlus
[7]	*String*	com.google.hangouts
[8]	*String*	com.facebook.Facebook
[1]	*Array*	*(3 values)*
[0]	*String*	net.whatsapp.WhatsApp
[1]	*String*	com.vine.iphone
[2]	*String*	com.toyopagroup.picaboo
displayName	*String*	Social
listType	*String*	folder

FIGURE 13-17 **Within the IconState.plist are the folders the user has created.**

- **/Library/SpringBoard/LockBackgroundThumbnail.jpg** This file is used as the image when the device is locked.
- **/Library/TCC/TCC.db** This database file identifies the apps on the iOS device (such as microphone, photos, Contacts, Calendar, and Twitter) and indicates to which services they have access. This can help the examiner identify rogue apps that might be attempting to gain elevated privileges.
- **/Library/Voicemail/voicemail.db** This database file identifies the voicemail date, sender, duration, and even when voicemail was deleted. Also within the /Library/Voicemail/ folder are the .amr files that contain the actual messages that are identified by the rowid in the database. Voicemails that are deleted can be recovered in this folder.

Note

Photo Stream, SMS, and Maps all have separate plist files in the /Library/Preferences/ folder and are prefixed by the com.apple.ids.service.com.apple file. These all contain information for proper authentication to the ESS.

Within the private/var/mobile/Media folder are pictures and videos, but this folder also is a storage place for additional files that may be of forensic significance:

- **/Media/Recordings/Recordings.db** This database file identifies voice recordings made with the iOS device along with the date and duration of the recording. The recordings themselves are also stored in the /Media/Recordings/ folder with an .m4a extension.
- **/Media/PhotoData/Photos.sqlite** Since the new advent of automatic photo categorization, this SQLite file can show the matches, which can be mapped to the images in the media folder.
- **/Media/PhotoData/CPLAssets** This folder stores the full-size images and videos that originated from iCloud.
- **/Media/PhotoData/Metadata/PhotoData/CPLAssets** The root folder contains images and additional child folders with images (both full and thumbnails) that originated from iCloud and have been optimized.
- **/Media/PhotoData/Metadata/PhotoCloudSharingData** This folder contains albums that are shared. The thumbnails for these are also located at /Media/PhotoData/Thumbnails/V2/PhotoData/PhotoCloudSharingData.
- **/Media/PhotoData/PhotoBulletins.plist** This file is used to show albums that have been shared and

accepted, which includes dates, times, and the recipient. This is great data if the subject is sharing illicit images.

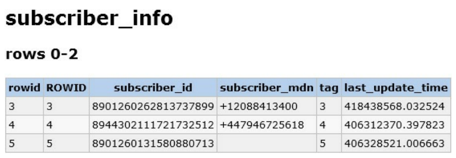
- **/Media/PhotoData/Thumbnails** This folder contains the thumbnail files of the images located in the DCIM folder and also can contain pictures that have been deleted. Not many tools can interpret the .thmb extension. These files are raw uncompressed images but can be viewed with an inexpensive application called iThmb Converter (www.ithmbconverter.com/en/download/) or directly within Oxygen Forensic Detective's interface.

Within the private/var folder is a preferences subfolder that contains additional iOS preference and network information. Here are some files that should be of interest to every examiner:

- **/Preferences/SystemConfiguration/com.apple.network.identification.plist** This property list contains the IP addresses used and assigned to the iOS device when communicating on both the cellular WAN and Wi-Fi. This file also lists the domain and IP addresses of the router, along with a timestamp of each event. This file does not exist in iOS 8.3 to 11.x devices. However, in the same directory, com.apple.wifi.plist is available
- **/Preferences/SystemConfiguration/com.apple.wifi.plist** This property list contains the Wi-Fi addresses to which the iOS device has connected when auto-joined, plus the name of the access point, the MAC address, and the type of security for the access point.
- **/Preferences/SystemConfiguration/preferences.plist** This property file lists various configuration preferences for communication, but also identifies the device name under the HostName and ComputerName keys.

Within the private/var/wireless folder is the CallHistory folder, where iOS 7 call data is located, but other subfolders also contain information that can help to identify not only app usage but also the exchange and substitution of subscriber information. This folder and subsequent database will exist only if the user has upgraded from iOS 7, since it was deprecated and again is stored under the Library folder.

- **/wireless/Databases/CellularUsage.db** This database file contains a subscriber_info table within the database that lists the subscriber ID (IMSI), the subscriber MDN (dialing number), and the last update date. The update date is when that IMSI and MDN were last used. This table also contains all SIM cards inserted and used within the device (see Figure 13-18). This is incredible information when attempting to determine subscriber information.



subscriber_info

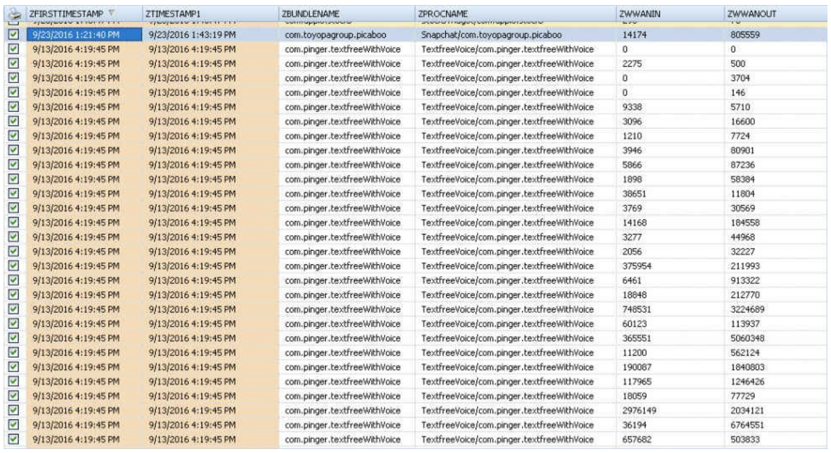
rows 0-2

rowid	ROWID	subscriber_id	subscriber_mdn	tag	last_update_time
3	3	8901260262813737899	+12088413400	3	418438568.032524
4	4	8944302111721732512	+447946725618	4	406312370.397823
5	5	8901260131580880713		5	406328521.006663

FIGURE 13-18 **The subscriber_info table in the CellularUsage.db shows that multiple SIM cards have been used in this iOS device.**

- **/wireless/Databases/DataUsage.db** This database file contains two significant tables within the database:

LiveUsage and Process. These tables contain the app name (bundle name), the process associated with the app, timestamps of usage, and the data coming in and out via the WAN. When these tables are put together using the foreign key OPT, the SQL query shows the activity of the app and its process (see Figure 13-19). This table can help the examiner determine whether a subject was using an app at the time of an event.



	ZFIRSTTIMESTAMP	ZTIMESTAMP1	ZBUNDLENAME	ZPROCNAME	ZWWANIN	ZWWANOUT
✓	9/23/2016 1:21:40 PM	9/23/2016 1:43:19 PM	com.toyopagroup.picaboo	Snapchat/com.toyopagroup.picaboo	14174	805559
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	0	0
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	2275	500
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	0	3704
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	0	146
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	9338	5710
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	3096	16600
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	1210	7724
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	3946	80901
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	5866	87236
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	1898	58384
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	38651	11804
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	3769	30569
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	14168	184558
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	3277	44968
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	2056	32227
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	375954	211993
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	6461	913322
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	18848	212770
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	748531	3224689
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	60123	113937
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	365551	5060348
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	11200	562124
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	190087	1840803
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	117965	1246426
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	18059	77729
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	2976149	2034121
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	36194	6764551
✓	9/13/2016 4:19:45 PM	9/13/2016 4:19:45 PM	com.pinger.textfreeWithVoice	TextfreeVoice/com.pinger.textfreeWithVoice	657682	503833

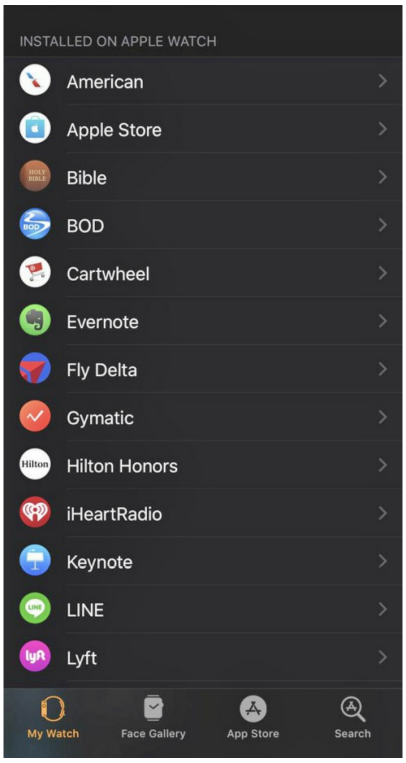
FIGURE 13-19 **Running a query on the DataUsage.db clearly shows critical information is available within this database file.**

- **/wireless/Preferences/com.apple.commcenter.plist** This property list identifies the ICCID along with the phone number assigned to the device, LastKnownICCID, CarrierBundleName, and more.
- **/wireless/Preferences/com.apple.commcenter.callservices.plist** This property list identifies the subscriber.id, last.account, and last.known.icloud.id. The last two keys are a hash of the actual account.

Group Shared Data

With the rush to bring a watch to market that has a workable interface Apple had to change the way the apps communicated. With a sandbox approach to apps, as discussed earlier, apps cannot communicate with one another to request or add data for obvious security reasons in a jailed device. So, typically, when an app is launched, it may ask the user for permission to access the contacts, camera, or other built-in Apple processes. The problem, however, lies in accessing data from other third-party applications with a device or Apple Watch. Shared folders were formed and are found in the /private/var/Application/ folder and prefixed with a group.<APP_DNS> format. This is another gold mine of data.

When a user downloads the Apple Watch app, and when setting up the watch, he or she can selectively install apps or install all the apps immediately (Figure 13-20). This means that any app that is compatible with Apple Watch will install, creating a shared relationship with the main app folder in the /private/var/Application/ folder.


INSTALLED ON APPLE WATCH

American

Apple Store

Bible

BOD

Cartwheel

Evernote

Fly Delta

Gymatic

Hilton Honors

iHeartRadio

Keynote

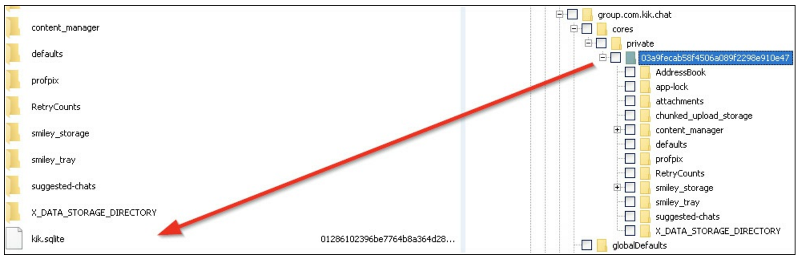
LINE

Lyft

FIGURE 13-20 **The Apple Watch application screen enables users to select apps to install on the watch, enabling a shared relationship.**

This is a data gold mine because of the development restrictions imposed by Apple on the sandboxing of data from apps. However, to make this data available to the Apple Watch, or other IoT devices, to communicate both by receiving data to the watch and sending replies to the watch, something happened. Apps now began adding full databases and working directories to the group shared folders! The most important part for the investigator to know is that the databases are not encrypted, and if the developer used the option not to include the backup in the main app folder, it does not affect the database or other data in the group shared area!

A perfect example is Kik Messenger. This messaging application is well known and used around the world. Its main database is located at private/var/mobile/Application/com.kik.chat, but it's currently available in the file system for forensic tools only if a full raw file system capture by jailbreak or other methods was conducted. This is because the developer elected not to include the database if a backup was called. However, within the group.com.kit.chat folder under /private/var/mobile/Applications, that main database, kik.sqlite, is clearly available to forensic tools, along with associated files (Figure 13-21).


```
content_manager
defaults
profpix
RetryCounts
smiley_storage
smiley_tray
suggested-chats
X_DATA_STORAGE_DIRECTORY
kik.sqlite                    01286102396be7764b8a364d28...
```

```
group.com.kik.chat
  cores
    private
      03a9fecab58f4506a089f2298e910e47
        AddressBook
        app-lock
        attachments
        chunked_upload_storage
        content_manager
        defaults
        profpix
        RetryCounts
        smiley_storage
        smiley_tray
        suggested-chats
        X_DATA_STORAGE_DIRECTORY
  globalDefaults
```

FIGURE 13-21 **From the shared storage under the group.com.kik.chat, the full database is available for**

parsing.

It appears that many of the apps that have been "installed" to the Apple Watch hold a fully synched database, as well as the associated files. Although some applications store only a property list (plist), this plist holds a massive amount of data. For example, Gmail, has a plist file located at private/var/mobile/Application/group.com.google.Gmail/Preferences that holds e-mail addresses sent and received. This area should always be examined because many forensic tools are not currently pulling data from these groups.

Using these locations, the examiner will be searching in the *what you don't see is what you need* area. Expert examiners will conduct this type of analysis and examination of mobile device data from an iOS device. Treating an iOS device as a small computing platform by examining, researching, and testing each file and location in an effort to uncover the meaning behind the data is a necessity.

iOS Evidentiary File Types

The examiner should understand the way data is stored and in what types of files it is stored. So far, this chapter has dealt with the location of artifacts; it now shifts to discuss the types of files that are commonly found within the iOS file system.

SQLite databases are the primary source of storage for apps and understandably a primary source of evidence for the examiner. User data types such as SMS, MMS, Contacts, Call Logs, Calendars, Notes, and Browser History, all discussed as "tip of the iceberg data," come from SQLite databases. But just because an automated tool pulls the information from the SQLite database does not necessarily mean that *all* the data has been collected from that particular database. To add to that statement, *all* data may have been collected, but it could be translated or decoded incorrectly. And what about the data contained within the many apps on the iOS device? The examiner was exposed to the various database locations and also cache areas that hold SQLite databases. These, too, will provide a wealth of information, so gaining an understanding of these files is necessary.

Property list, or plist, files are also within the iOS file system; these are used to store app settings, device settings, and user-specific settings. Next to SQLite databases, plist files are the second-most observed and examined file for the mobile forensic examiner. The plist files come in two forms—binary and XML. Both file types are covered in this section to help the examiner learn to interpret and view them using several different tools.

Several other file types are not as prevalent within the iOS file system, but these will also be covered to give the examiner an idea as to what is generally stored and what can be recovered to assist in an examination. Gaining knowledge about what these files can yield to an examination will help the examiner understand how these files store data and how to recover their hidden treasures.

SQLite Databases

According to SQLite.org, SQLite is the most widely deployed SQL database engine in the world. SQLite is used not only in iOS devices but also in Android devices, Windows Phone, and BlackBerry 10 devices, and in both Windows and Mac desktop applications. Many desktop applications use SQLite because of its simplistic deployment and use, and this is the case within mobile apps as well. SQLite does not need a client-server relationship like SQL, nor does it have to be installed. All tables, views, and other associated methods are included in a single file within a compact library. SQLite is a public-domain software library that is well documented and free for both commercial and private use.

The current build of SQLite, as of this writing, is 3.23.1. Because of its forensic implications, an examiner should become familiar with the structure of this common file type. A surface-level explanation of the SQLite file type is outlined here with the major structural pieces defined.

Note

Many SQLite database browsers will enable the examiner to view the contents of a SQLite database, so this section is not intended to inform the examiner on how to create a tool, decode, and visually display a SQLite database. The intention is to help the examiner understand the structure and to "see" that data will not be displayed within a SQLite database viewer if it is no longer referenced (that is, deleted). Many SQLite documents and books cover the structure, ideology, and methods, and the examiner can supplement the introduction and overview in this section by consulting those documents.

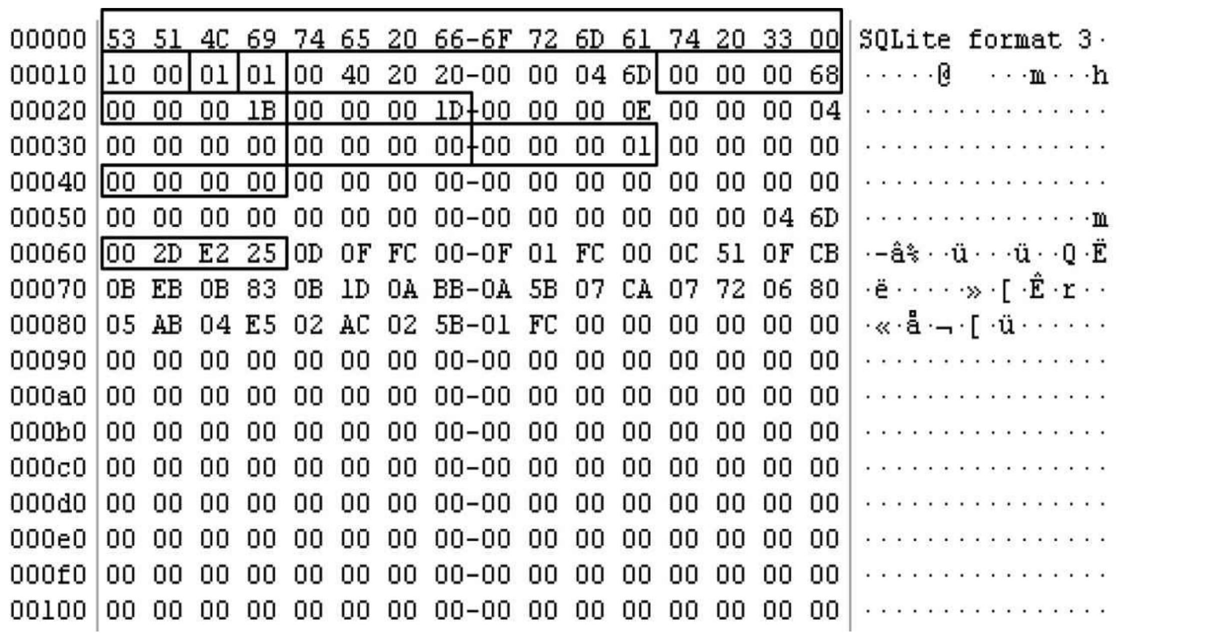
This section is intended to arm the forensic examiner with the information needed to recover valuable "hidden" data from these files and to help the examiner understand why, where, and how this can be accomplished. The recovery of data from the logical structure of tables and rows will be covered in Chapter 14.

A SQLite database is like a file that stores many records. SQLite.org defines the SQLite file as the "main database file." SQLite is *transactional*, which means that it stores data within a second file called a *rollback journal*; in other words, if transactions on the database are not completed properly, they are able to be rolled back. If a rollback journal is being used, the secondary file is created, associated, and in the same directory as the database file and has the same name, with *-journal* appended. If the SQLite file is in write-ahead log (WAL) mode, transactions will be stored in the associated WAL file (covered in detail in the next section). In the main database file, the records are stored in pages. The page size is determined by the 2-byte big-endian integer located at an offset 16 bytes from the start of the main database file. The page size is based on a power of 2 and is between 512 and 65,536 bytes.

The first 100 bytes of the database file is the file header, which is divided up into fields: the first 16 bytes contain SQLite format 3, followed by the 2 bytes for the page size at offset 16. These values in the header are all stored in big-endian format. The maximum page number is 2,147,483,646 and the minimum is one 512-byte page starting at page 1. The maximum database size could be around 140 terabytes if the maximum number of pages each held the maximum number of bytes! Table 13-5 shows the database header formats most critical to the examiner, which are visually represented in Figure 13-22.

TABLE 13-5 **SQLite Database Header Details as Outlined on SQLite.org**

Offset	Size (Bytes)	Description
0	16	The header string: "SQLite format 3\00".
16	2	The database page size in bytes.
18	1	File format: 1 for legacy journal and 2 for WAL.
19	1	File format: 1 for legacy journal and 2 for WAL.
28	4	Size of the database file in pages.
32	4	Page of the first freelist (free page list) trunk page.
36	4	Total number of freelist pages.
52	4	Page number of largest root B-tree page when in auto-vacuum or incremental vacuum. If 0, auto-vacuum is not enabled and the examiner can use only the freefile table, and parent/child tables cannot be mapped to a pointer map.
56	4	Text encoding used: UTF-8 = 1, UTF-16le = 2, UTF-16be = 3.
64	4	If any number other than 0 is used, incremental-vacuum mode is enabled.
96	4	SQLite version number.

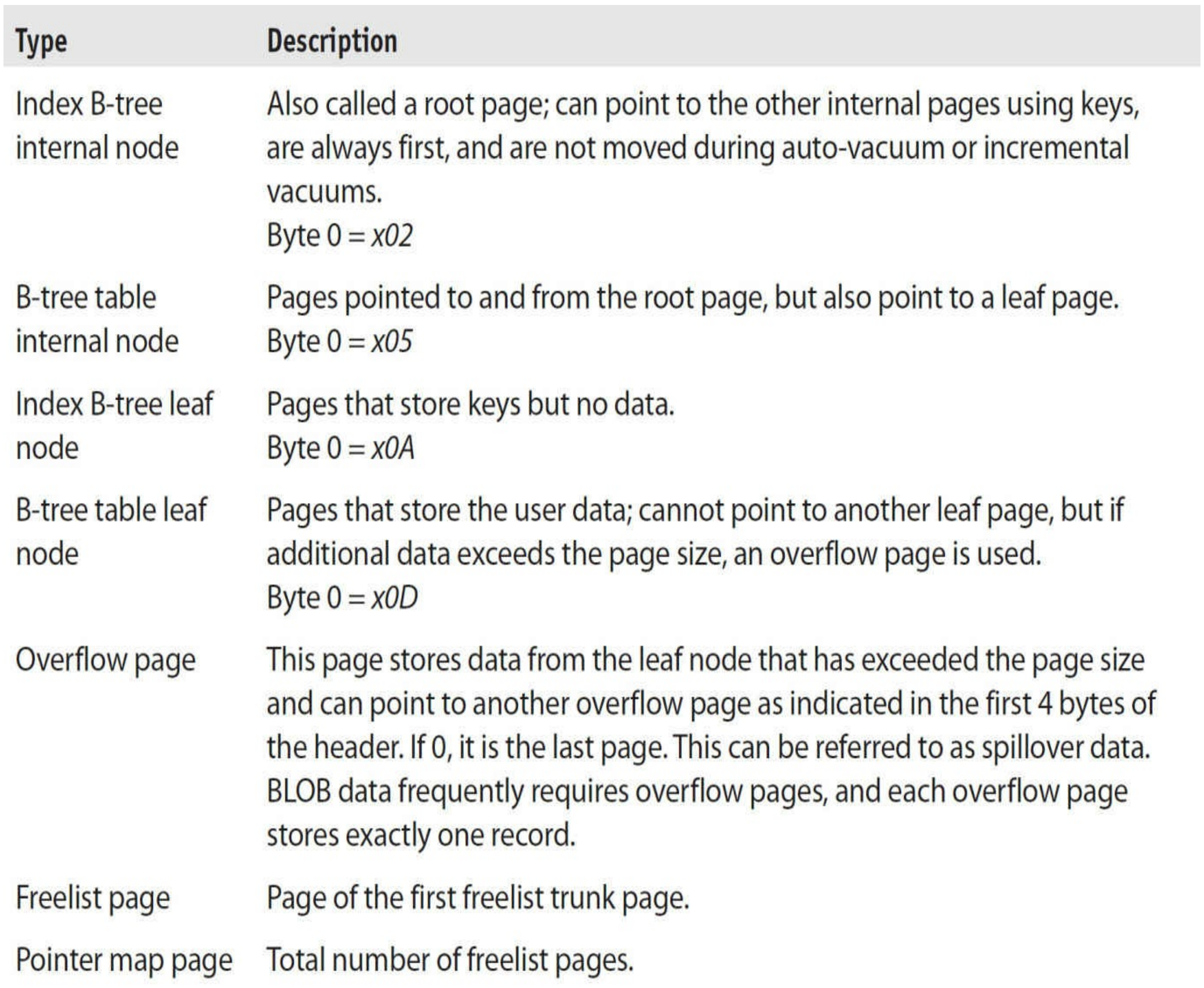


```
00000  53 51 4C 69 74 65 20 66-6F 72 6D 61 74 20 33 00  SQLite format 3·
00010  10 00 01 01 00 40 20 20-00 00 04 6D 00 00 00 68  ·····@   ···m···h
00020  00 00 00 1B 00 00 00 1D-00 00 00 0E 00 00 00 04  ················
00030  00 00 00 00 00 00 00 00-00 00 00 01 00 00 00 00  ················
00040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
00050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 04 6D  ···············m
00060  00 2D E2 25 0D 0F FC 00-0F 01 FC 00 0C 51 0F CB  ·-â%··ü···ü··Q·Ë
00070  0B EB 0B 83 0B 1D 0A BB-0A 5B 07 CA 07 72 06 80  ·ë·····»·[·Ê·r··
00080  05 AB 04 E5 02 AC 02 5B-01 FC 00 00 00 00 00 00  ·«·å·¬·[·ü······
00090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000a0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000b0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000c0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000d0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000f0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
00100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

FIGURE 13-22 **The SQLite header can tell the examiner many things during a manual attempt to recover information from various areas of the database file.**

A SQLite database contains several types of pages, outlined in Table 13-6. B-tree parent pages within a database point to their children, but the reverse relationship is not the same. Children can point to an overflow page, which can then point to another overflow page, and so on, but children cannot point back to their parent. Understanding the various table types can help the examiner identify the location and identity of pages that may contain freefile pages, which could ultimately hold deleted data. Using the freefile list to identify the location of the various freefile pages can help the examiner uncover substantial records not otherwise collected by most mobile forensic tools.

TABLE 13-6 **SQLite Page Types and Descriptions**



Type	Description
Index B-tree internal node	Also called a root page; can point to the other internal pages using keys, are always first, and are not moved during auto-vacuum or incremental vacuums. Byte 0 = *x02*
B-tree table internal node	Pages pointed to and from the root page, but also point to a leaf page. Byte 0 = *x05*
Index B-tree leaf node	Pages that store keys but no data. Byte 0 = *x0A*
B-tree table leaf node	Pages that store the user data; cannot point to another leaf page, but if additional data exceeds the page size, an overflow page is used. Byte 0 = *x0D*
Overflow page	This page stores data from the leaf node that has exceeded the page size and can point to another overflow page as indicated in the first 4 bytes of the header. If 0, it is the last page. This can be referred to as spillover data. BLOB data frequently requires overflow pages, and each overflow page stores exactly one record.
Freelist page	Page of the first freelist trunk page.
Pointer map page	Total number of freelist pages.

The page header in the SQLite database file can be examined to assist in identifying the type of page, the first free block on the page, the number of cells on the page, and much more information. The page header comprises 8 bytes for leaf pages and 12 bytes for interior pages in big-endian format. The fields for the header are outlined in Table 13-7. (Table 13-6 identifies the type of pages along with the first byte that identifies the page type.)

TABLE 13-7 **Page Header Format as Described in SQLite.org Documentation**

Offset	Size (Bytes)	Description
0	1	The type of page, as listed in Table 13-6, represented by the first byte
1	2	Two-byte integer is the start of the first free block, 0 if there are no free blocks
3	2	Two-byte integer giving the number of cells on the page
5	2	Start of the cell content area; 0 value is interpreted as 65,536 bytes
7	1	Number of fragmented free bytes in the cell content area
8	4	Value appears only in the header of interior tree pages and is the page number of the rightmost pointer (map)

Using the file type along with the storage areas, the examiner can identify the contents within the page and uncover data that might not be referenced by the database active index. This will be evident as the examiner becomes more familiar with the freelist and free pages within the SQLite database.

The freelist is a list of the unused pages within a database (see Figure 13-23). However, if the offset 32 (number of freelist pages) 4-byte value is *0*, then the freelist does not contain any unused pages within the database. This freelist and the subsequent free pages can be important to an investigator. The free pages are not active and are where the database stores the data that is no longer referenced within the SQLite table. An examiner may understand this as deleted data. Information the user of the app has chosen to delete is removed from the table references and moved to the free pages within the freelist area. Freelist pages can either be trunk pages or leaf pages. The freelist file header, as dissected in Table 13-8, points to a trunk page, which can then point to multiple leaf pages. When a SQLite page becomes inactive, it is added to the freelist and will not be released to the database file system. When information is added to the database by the user, the database will take free pages from the freelist to store the data, and the table becomes active and no longer referenced in the freefile index. If the list is empty, the database will obtain new pages and then add them to the database file.

TABLE 13-8 **Freelist Header Breakdown to Assist When Locating Inactive Pages**

Offset	Bytes	Description
0	4	Integer is the page number of the next freelist trunk page, or *0* if this is the last freelist trunk
4	4	Integer that is the number of leaf page pointers to follow
8	4	Subsequent integers are page locations for free pages that will follow the number of leaf page pointers specified in bytes 4–7

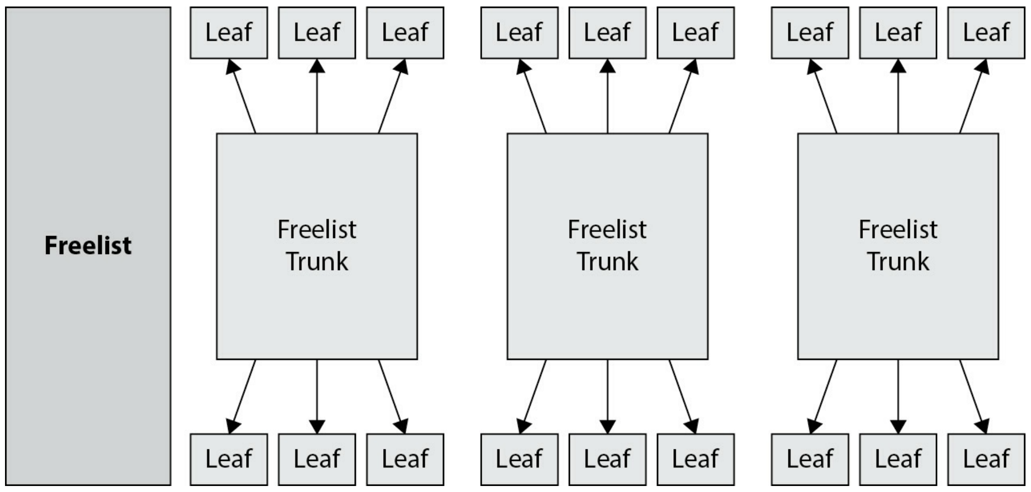


FIGURE 13-23 **The freelist is an index to trunk files that point to leaves where the data is stored.**

The examiner should also understand both the auto-vacuum and the pointer map of the SQLite database. *Auto-vacuum* is the ability of a SQLite database to stay small. Auto-vacuum is not the same as vacuum in SQLite: Vacuum cleans the main database by copying the contents to a temporary file and then reloading the file from the copy; this action removes deleted data (free pages) and completely cleans the database structure. Auto-vacuum does not eliminate the free pages; it moves them. This is very crucial information to an examiner regarding the recovery of deleted data from a SQLite database. However, to understand how the database locates information within the table and moves information around, the examiner should also understand the pointer map (ptrmap) pages.

The *pointer map* and pages are inserted into the SQLite database to assist with the efficiency of both the auto- and incremental vacuum processes. The pointer map is not available within the SQLite database if the auto-vacuum or incremental vacuum is not enabled. This can be determined by looking at offset 52: if this is *0*, then auto-vacuum or incremental vacuum has not been set. When the database moves a page, the pointer is updated. The pointer map is also a lookup table that can be used to find the parent of the child leaf—something that is not contained within the actual child leaf. Within the pointer map, each pointer is made up of 5 bytes that represent the type of page and the related parent. The pointers are in order starting at page 3 and progress to the number of pages. Each pointer represents one page within the database file. The size of the pointer map in bytes can be obtained using the following formula:

$$(2 - \text{number of pages}) \times 5 = \text{number of bytes}$$

The pointer map is confined to the page size outlined by the SQLite database, as identified by byte 16 in the database header; any pointers that exceed *page size*/5 will be assigned to another pointer map page.

Tip

If an examiner had a database file with a page size of 1024 bytes per page, the total number of pages that could be referenced within a pointer map page would be 1024/5 = 204 page references maximum. Because the first and second pages are not counted (database header and pointer map), if a new page is needed, a new pointer map page will be created and the new page (206) will be referenced.

To help you with decoding the pointer map, Table 13-9 outlines the significant bytes and defines the meanings. The pointer map can be important to the examiner when attempting to identify the parent of a deleted record when coupling the information obtained from the freelist and associated free pages.

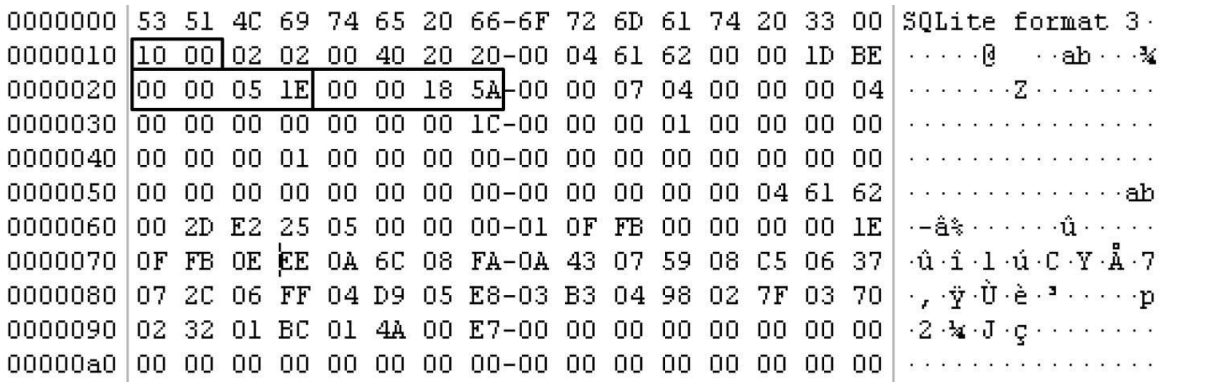
TABLE 13-9 **Pointer Map Records and How to Decode the 5-byte Record to Determine What Type of Page Is Referenced (VV represents a variable HEX character)**

Description	Pointer Map Bytes
The first byte (*x01*) indicates the root page. The following four bytes indicate there is no parent page.	*x01* x00 x00 x00 x00
The first byte (*x02*) indicates a free page. The following four bytes indicate there is no parent page.	*x02* x00 x00 x00 x00
The first byte (*x03*) indicates the first page of an overflow chain. The following four bytes indicate the parent page of the overflow chain.	*x03* xVV xVV xVV xVV
The first byte (*x04*) indicates it is not the first page in an overflow chain. The following four bytes indicate the parent overflow chain.	*x04* xVV xVV xVV xVV
The first byte (*x05*) indicates a page that is part of a table or index but not an overflow page or root page. The following four bytes indicate the parent page of the parent tree node.	*x05* xVV xVV xVV xVV

Uncovering Inactive SQLite Data via Freelists

As mentioned, inactive data within a SQLite database can be recovered using the database header, the freelist, and then the associated pages. In this section, the previously covered material will be put to the test to uncover deleted data from an sms.db file from an iOS device. This technique can be used on any SQLite database the examiner may encounter.

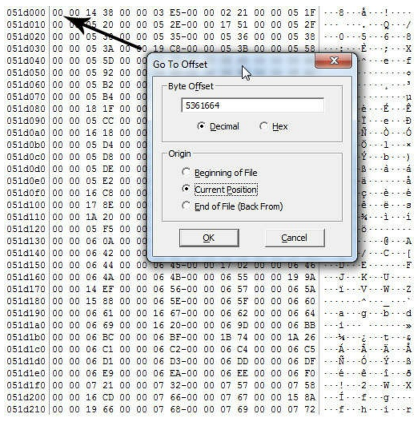
First, we open the SQLite database within FTK Imager and examine it within the HEX viewer. (FTK Imager is a free tool from AccessData.) Figure 13-24 shows the database file, and highlighted is the size of the page in bytes (4096), the freelist page number (1310), and the number of free pages available (6234).



```
0000000  53 51 4C 69 74 65 20 66-6F 72 6D 61 74 20 33 00  SQLite format 3·
0000010  10 00 02 02 00 40 20 20-00 04 61 62 00 00 1D BE  ·····@  ··ab···¾
0000020  00 00 05 1E 00 00 18 5A-00 00 07 04 00 00 00 04  ·······Z········
0000030  00 00 00 00 00 00 00 1C-00 00 00 01 00 00 00 00  ················
0000040  00 00 00 01 00 00 00 00-00 00 00 00 00 00 00 00  ················
0000050  00 00 00 00 00 00 00 00-00 00 00 00 00 04 61 62  ··············ab
0000060  00 2D E2 25 05 00 00 00-01 0F FB 00 00 00 00 1E  ·-â%·····û·····
0000070  0F FB 0E EE 0A 6C 08 FA-0A 43 07 59 08 C5 06 37  ·û·î·l·ú·C·Y·Å·7
0000080  07 2C 06 FF 04 D9 05 E8-03 B3 04 98 02 7F 03 70  ·,·ÿ·Ù·è·³·····p
0000090  02 32 01 BC 01 4A 00 E7-00 00 00 00 00 00 00 00  ·2·¼·J·ç········
00000a0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

FIGURE 13-24 **The page size, the freelist page number, and the number of pages within the sms.db database file from an iOS device**

Next, we navigate to the freelist area. The formula subtracts 1 from the freelist page number from the freelist (to take into account the database header that always holds the first page). The size of the page is then multiplied by the bytes to obtain the offset within the sms.db for the freelist (see Figure 13-25):



```
051d000  00 00 14 38 00 00 03 E5-00 00 02 21 00 00 05 1F   ···8···å···!····
051d010  00 00 05 20 00 00 05 2E-00 00 17 51 00 00 05 2F   ··· ···.···Q···/
051d020  00 00 05 30 00 00 05 35-00 00 05 36 00 00 05 38   ···0···5···6···8
051d030  00 00 05 3A 00 00 19 C8-00 00 05 3B 00 00 05 58   ···:···È···;···X
051d040  00 00 05 5D 00 00                                 ^···e···f
051d050  00 00 05 92 00 00                                 ·········
051d060  00 00 05 B2 00 00                                 ···,···³
051d070  00 00 05 B4 00 00                                 ········µ
051d080  00 00 18 1F 00 00                                 è···É···Ê
051d090  00 00 05 CC 00 00                                 Ï···e···Ð
051d0a0  00 00 16 18 00 00                                 Ñ···Ò···Ó
051d0b0  00 00 05 D4 00 00                                 Ö···1···×
051d0c0  00 00 05 D8 00 00                                 Ý···b···)
051d0d0  00 00 05 DE 00 00                                 ß···à···á
051d0e0  00 00 05 E2 00 00                                 ä·······å
051d0f0  00 00 16 C8 00 00                                 ç···è···é
051d100  00 00 17 8E 00 00                                 ê···ë···з
051d110  00 00 1A 20 00 00                                 ¾···ì···í
051d120  00 00 05 F5 00 00                                 ö········
051d130  00 00 06 0A 00 00                                 ·····@···A
051d140  00 00 06 42 00 00                                 /···C···[
051d150  00 00 06 44 00 00 06 45-00 00 17 02 00 00 06 46   ···D···E·······F
051d160  00 00 06 4A 00 00 06 4B-00 00 06 55 00 00 19 9A   ···J···K···U····
051d170  00 00 14 EF 00 00 06 56-00 00 06 57 00 00 06 5A   ···ï···V···W···Z
051d180  00 00 15 88 00 00 06 5E-00 00 06 5F 00 00 06 60   ·······^···_···`
051d190  00 00 06 61 00 00 16 67-00 00 06 62 00 00 06 64   ···a···g···b···d
051d1a0  00 00 06 69 00 00 16 20-00 00 06 9D 00 00 06 BB   ···i··· ·······»
051d1b0  00 00 06 BC 00 00 06 BF-00 00 1B 74 00 00 1A 26   ···¼···¿···t···&
051d1c0  00 00 06 C1 00 00 06 C2-00 00 06 C4 00 00 06 C5   ···Á···Â···Ä···Å
051d1d0  00 00 06 D1 00 00 06 D3-00 00 06 DD 00 00 06 DF   ···Ñ···Ó···Ý···ß
051d1e0  00 00 06 E9 00 00 06 EA-00 00 06 EE 00 00 06 F0   ···é···ê···î···ð
051d1f0  00 00 07 21 00 00 07 32-00 00 07 57 00 00 07 58   ···!···2···W···X
051d200  00 00 16 CD 00 00 07 66-00 00 07 67 00 00 15 8A   ···Í···f···g····
051d210  00 00 19 66 00 00 07 68-00 00 07 69 00 00 07 72   ···f···h···i···r
```

Go To Offset

Byte Offset: 5361664

[✓ Decimal] Hex

Origin: Beginning of File [✓ Current Position] End of File (Back From)

OK Cancel

FIGURE 13-25 **The location of the freelist is obtained by using the Go To Offset in FTK Imager.**

$$(1310 - 1) \times 4096 = 5361664$$

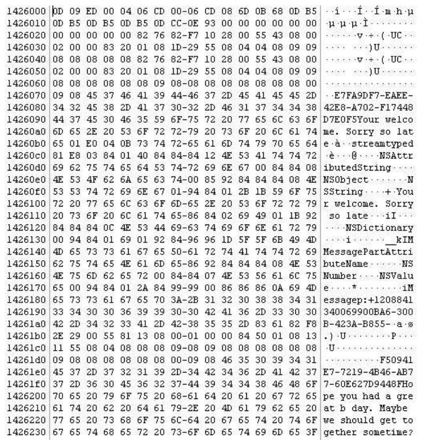
As outlined in Table 13-8, the freelist's first four bytes identify whether there is another freelist trunk. If this value is equal to *0*, then this is the last freelist or the last freelist trunk. In this example, the first four bytes (x00 x00 x14 x38) equal integer 5176, indicating the page size of the next freelist trunk page:

$$(5176 - 1) \times 4096 = 21196800$$

However, the first freelist should be examined.

The second set of bytes after the freelist trunk list pointer, starting at offset 4, indicates the number of pages referenced within this freelist. This value (x00 x00 x00 x83) in the example equals integer 131, indicating there are 131 pages within this freelist. The rest of the 4-byte values starting at offset 8 within the freelist page represent the location of the leaf pages. These are the pages that represent the inactive data within the database file.

In this example, navigating to the first listed leaf (x00 x00 x14 x27) with an integer value of 5159 represents the page within the database. Using the same formula, (5159 – 1) × 4096 = 21127168, and navigating to this location, it is evident that the page contains text messages, particularly iMessages, and is contained in a table leaf node page as indicated by the first byte (x0D) (see Figure 13-26). Some of the data is repeated within the free pages, but this is extremely useful for any investigation. This information can be collected, searched, and extracted now that the examiner can identify the pages from the freelist pages.



```
1426000 0D 09 ED 00 04 06 CD 00-06 CD 08 6D 0B 68 0D B5  ··i···Í··Í·m·h·µ
1426010 0D B5 0D B5 0D B5 0D CC-0E 93 00 00 00 00 00 00  ·µ·µ·µ·Ì········
1426020 00 00 00 00 00 82 76 82-F7 10 28 00 55 43 08 00  ······v·÷·(·UC··
1426030 02 00 00 83 20 01 08 1D-29 55 08 04 04 08 09 09  ····  ···)U······
1426040 08 08 08 08 08 82 76 82-F7 10 28 00 55 43 08 00  ······v·÷·(·UC··
1426050 02 00 00 83 20 01 08 1D-29 55 08 04 04 08 09 09  ····  ···)U······
1426060 08 08 08 08 08 08 09 08-08 08 08 08 08 08 08 00  ················
1426070 09 08 45 37 46 41 39 44-46 37 2D 45 41 45 45 2D  ··E7FA9DF7-EAEE-
1426080 34 32 45 38 2D 41 37 30-32 2D 46 31 37 34 34 38  42E8-A702-F17448
1426090 44 37 45 30 46 35 59 6F-75 72 20 77 65 6C 63 6F  D7E0F5Your welco
14260a0 6D 65 2E 20 53 6F 72 72-79 20 73 6F 20 6C 61 74  me. Sorry so lat
14260b0 65 01 E0 04 0B 73 74 72-65 61 6D 74 79 70 65 64  e·à··streamtyped
14260c0 81 E8 03 84 01 40 84 84-84 12 4E 53 41 74 74 72  ·è··@···NSAttr
14260d0 69 62 75 74 65 64 53 74-72 69 6E 67 00 84 84 08  ibutedString····
14260e0 4E 53 4F 62 6A 65 63 74-00 85 92 84 84 84 08 4E  NSObject·······N
14260f0 53 53 74 72 69 6E 67 01-94 84 01 2B 1B 59 6F 75  SString···+·You
1426100 72 20 77 65 6C 63 6F 6D-65 2E 20 53 6F 72 72 79  r welcome. Sorry
1426110 20 73 6F 20 6C 61 74 65-86 84 02 69 49 01 1B 92   so late··iI···
1426120 84 84 84 0C 4E 53 44 69-63 74 69 6F 6E 61 72 79  ····NSDictionary
1426130 00 94 84 01 69 01 92 84-96 96 1D 5F 5F 6B 49 4D  ····i······__kIM
1426140 4D 65 73 73 61 67 65 50-61 72 74 41 74 74 72 69  MessagePartAttri
1426150 62 75 74 65 4E 61 6D 65-86 92 84 84 84 08 4E 53  buteName······NS
1426160 4E 75 6D 62 65 72 00 84-84 07 4E 53 56 61 6C 75  Number···NSValu
1426170 65 00 94 84 01 2A 84 99-99 00 86 86 86 0A 69 4D  e····*········iM
1426180 65 73 73 61 67 65 70 3A-2B 31 32 30 38 38 34 31  essagep:+1208841
1426190 33 34 30 30 36 39 39 30-30 42 41 36 2D 33 30 30  340069900BA6-300
14261a0 42 2D 34 32 33 41 2D 42-38 35 35 2D 83 61 82 F8  B-423A-B855-·a·ø
14261b0 2E 29 00 55 81 13 08 00-01 00 00 84 50 01 08 13  .)·U········P···
14261c0 11 55 08 04 08 08 08 09-08 09 08 08 08 08 08 08  ·U··············
14261d0 09 08 08 08 08 08 08 00-09 08 46 35 30 39 34 31  ··········F50941
14261e0 45 37 2D 37 32 31 39 2D-34 42 34 36 2D 41 42 37  E7-7219-4B46-AB7
14261f0 37 2D 36 30 45 36 32 37-44 39 34 34 38 46 48 6F  7-60E627D9448FHo
1426200 70 65 20 79 6F 75 20 68-61 64 20 61 20 67 72 65  pe you had a gre
1426210 61 74 20 62 20 64 61 79-2E 20 4D 61 79 62 65 20  at b day. Maybe 
1426220 77 65 20 73 68 6F 75 6C-64 20 67 65 74 20 74 6F  we should get to
1426230 67 65 74 68 65 72 20 73-6F 6D 65 74 69 6D 65 3F  gether sometime?
```

FIGURE 13-26 **First free page leaf list in the freelist of the sms.db. This is a table leaf node as represented in the first byte (x0D), and Table 13-7 can assist in obtaining additional page information.**

Some of the data collected from the free pages could be void of metadata (such as phone numbers, dates, and times). The next section will cover write-ahead logs, which can also hold significant data that often is not accessible within the original database.

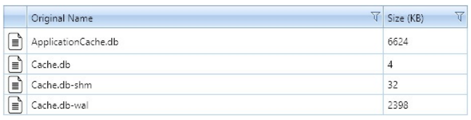
Journal Files and Write-Ahead Logs

SQLite databases use files somewhat like cache files, which operate in a way similar to journal and write-ahead log files, but instead of writing changes to the main database, writing occurs to a separate file. Beginning with SQLite version 3.7.0, the write-ahead log (WAL) file became supported as a new transaction control mechanism. All data that has yet to be committed, both new data and altered data, is saved to the WAL instead of directly to the main SQLite database if the main database is in WAL mode. What is also extremely interesting about WAL files is that the database engine will reference an altered page within the WAL and ignore that page in the main database until a checkpoint operation occurs.

A SQLite database can use either a journal file or a WAL file, but it cannot use both concurrently. A WAL file will be used instead of a journal file if the SQLite database is operating in WAL mode. Both WAL files and journal files must be located in the same folder as the main database file in order to operate properly. If a WAL file is removed, or if a database is removed and subsequent queries are run against the database without an associated WAL or journal file, a file will be created to support the database. The SQLite database transactions will COMMIT when a WAL *frame* is written that contains a COMMIT marker. A WAL will generally contain multiple transactions. When a COMMIT occurs, the information that is unique and has yet to be written to the main database gets committed, and any data that is non-unique or inactive remains within the WAL file. The WAL file can assist with corruption and failure of the main database, and if this occurs, the WAL file can then rebuild the main database with the data contained within the WAL. As mentioned, the WAL does not have to exist within the logical file system, and these files are not recovered with a standard Apple Mobile Backup (such as iTunes). The recovery of these files is available only by using a mobile solution capable of conducting a more advanced collection using the file_relay service or techniques to obtain a physical collection.

The main database the WAL file supports can be easily recognized because it shares the database name.

The WAL file often is much larger than the database itself. In Figure 13-27, the Cache.db WAL file is almost 600 times larger than the main database! Quite often, the WAL file contains duplicate data along with unique data from the main database. This is primarily due to the main database leaving the WAL file even upon issuing a checkpoint. The WAL file just continues to grow.

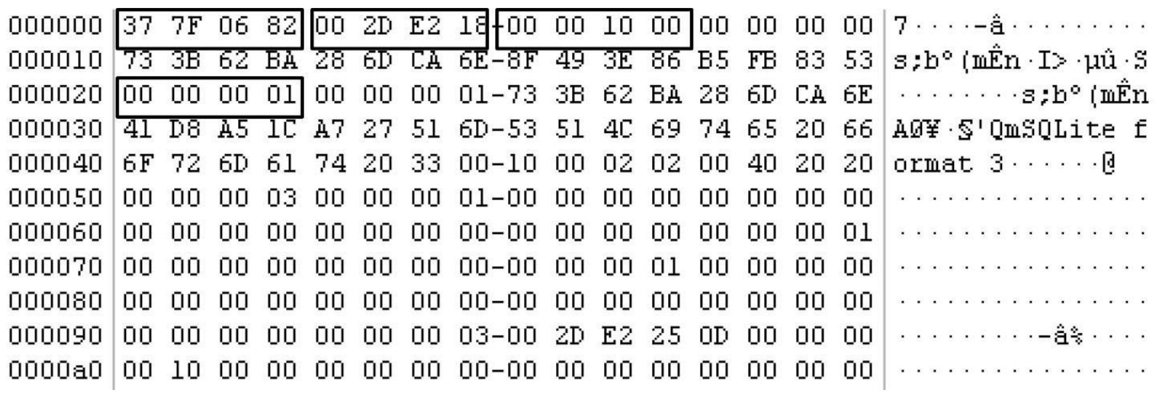


Original Name	Size (KB)
ApplicationCache.db	6624
Cache.db	4
Cache.db-shm	32
Cache.db-wal	2398

FIGURE 13-27 **Often the WAL file is much larger than the main database, because it caches data from many instances of the database runs with the *frames*.**

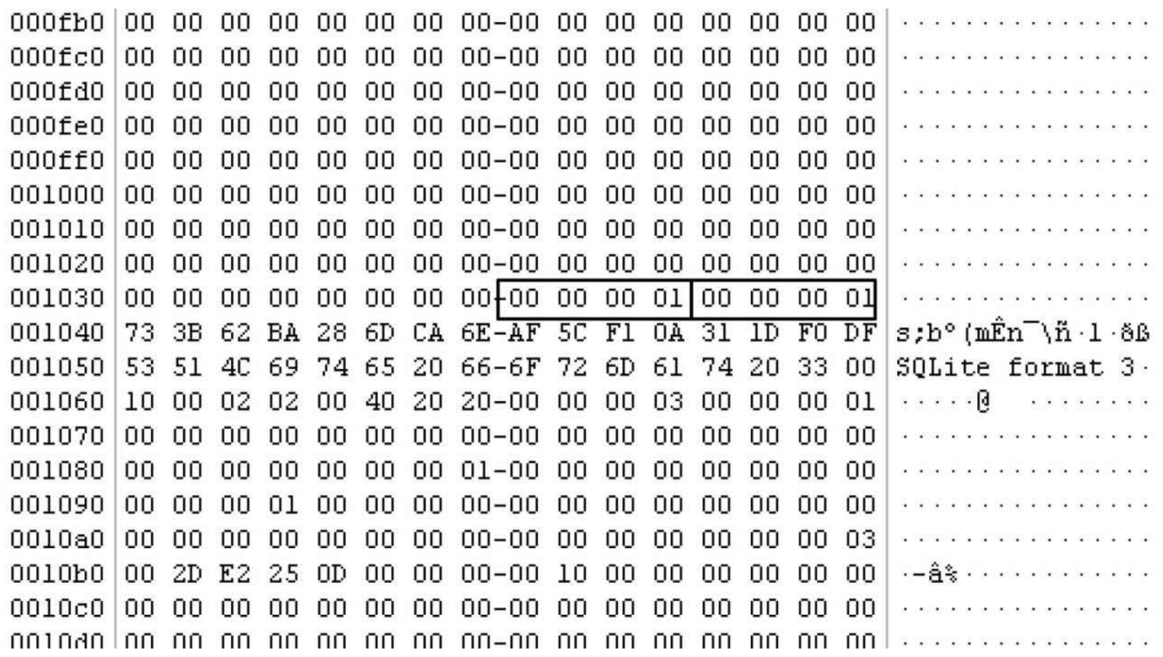
The WAL file is not used just as a backup, but as a storage location for data prior to the data reaching the main database. All changes to the database are recorded to the WAL file by use of *frames*, recordings of the changed content from a single page within the main database file. Data is written back to the main database in a *checkpoint* operation. During this operation, the data within the WAL file is not removed but is marked as inactive and can be overwritten if a new frame or frames are introduced to the WAL.

The WAL file has a unique file header, as shown in Figure 13-28, along with unique frame headers with the file itself. The mapping of each is outlined within Tables 13-10 and 13-11. Using this information can help the examiner identify the size and characteristics of each page within the file. Like the main SQLite database the WAL represents, the WAL header can describe much about the contents of the contained file, and using the header map, the investigator can traverse the file to identify data that may have been committed or that has not been committed to the main database file.



```
000000  37 7F 06 82 00 2D E2 18-00 00 10 00 00 00 00 00  7····-â·········
000010  73 3B 62 BA 28 6D CA 6E-8F 49 3E 86 B5 FB 83 53  s;b°(mÊn·I>·µû·S
000020  00 00 00 01 00 00 00 01-73 3B 62 BA 28 6D CA 6E  ········s;b°(mÊn
000030  41 D8 A5 1C A7 27 51 6D-53 51 4C 69 74 65 20 66  AØ¥·§'QmSQLite f
000040  6F 72 6D 61 74 20 33 00-10 00 02 02 00 40 20 20  ormat 3······@
000050  00 00 00 03 00 00 00 01-00 00 00 00 00 00 00 00  ················
000060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 01  ················
000070  00 00 00 00 00 00 00 00-00 00 00 01 00 00 00 00  ················
000080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000090  00 00 00 00 00 00 00 03-00 2D E2 25 0D 00 00 00  ·········-â%····
0000a0  00 10 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

FIGURE 13-28 **The header of a WAL file from a Cache.db file, with the magic number, file version, and database page size. Also, the first frame is visible.**



```
000fb0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000fc0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000fd0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000fe0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000ff0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
001000 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
001010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
001020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
001030 00 00 00 00 00 00 00 00-00 00 00 01 00 00 00 01  ················
001040 73 3B 62 BA 28 6D CA 6E-AF 5C F1 0A 31 1D F0 DF  s;b°(mÊn¯\ñ·1·ðß
001050 53 51 4C 69 74 65 20 66-6F 72 6D 61 74 20 33 00  SQLite format 3·
001060 10 00 02 02 00 40 20 20-00 00 00 03 00 00 00 01  ·····@  ········
001070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
001080 00 00 00 00 00 00 00 01-00 00 00 00 00 00 00 00  ················
001090 00 00 00 01 00 00 00 00-00 00 00 00 00 00 00 00  ················
0010a0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 03  ················
0010b0 00 2D E2 25 0D 00 00 00-00 10 00 00 00 00 00 00  ·-â%············
0010c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
0010d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

FIGURE 13-29 **Traversing 4096 bytes as indicated in the header for the page size, the next frame is located within the WAL.**

TABLE 13-10 **The WAL Header Format as Outlined on SQLite.org**

Offset	Bytes	Description
0	4	Identity of the WAL file, commonly referred to as the "magic number." Using this can help located deleted WAL files. It will be either x37 x7F x06 x82 or x37 x7F x06 x83.
4	4	File format version.
8	4	Page size of the database.
12	4	Checkpoint sequence number.
16	4	Salt-1: This is an integer that increments with each checkpoint. It can be used to determine the order of pages if there are duplicates within the WAL. Newer pages have larger values, and if the Salt-1 is the same, the page that is positioned closest to end of file is the newest version.
20	4	Salt-2: This will be a different random number for each checkpoint.
24	4	Checksum-1: First part of a checksum on the first 24 bytes of the header.
28	4	Checksum-1: Second part of the checksum on the first 24 bytes of the header.

TABLE 13-11 **The Frame Header Format as Outlined by SQLite.org**

Offset	Bytes	Description
0	4	Page number of the record.
4	4	For committed records, the size of the database file in pages after the commit. All other records will be 0.
8	4	Salt-1 that is copied from the WAL header.
12	4	Salt-2 that is copied from the WAL header.
16	4	Checksum-1: Cumulative checksum up to and including this page.
20	4	Checksum-1: Second half of the cumulative checksum.

To traverse the WAL, the examiner must first obtain the page size from the four bytes staring at offset 8. Once the page size is obtained, the examiner should navigate to offset 32, which is the start of the first frame. The examiner can then navigate using the following formula,

$$(P + 24 = F)$$

where *P* equals the page size and *F* is the resulting frame offset. The *24* is added to the solution to take into account that the examiner will be starting from the beginning of the frame header. If the examiner starts at the actual start of the page, the offset would be equal to the page size. Each page, with the exception of a root table, will contain the page identifiers listed in Table 13-6. The root table is easily identified with the database header SQLite format 3.

Committed data that still resides within the frames, as shown in Figure 13-28, can also be historical data that was once removed by the user and is no longer listed in the main database. This is another place the examiner should traverse when looking for information that may have not been added to the main database at the time of the collection. In addition, data that has been removed from the main database but still resides in the WAL file can be located. Using the information in the file header, additional pages can be located and the contents of the entire WAL file can be mapped, as shown in Figure 13-29.

The WAL file cannot be viewed in a standard database viewer for various reasons, including the fact that the WAL file does not have a SQLite file header at the correct offset, there are multiple SQLite headers, and the file contains incomplete and sometimes unintelligible bytes for a database viewer. Also, if the WAL file exists in a folder outside of a forensic solution's reach—say, within a logical folder on a desktop computer with its main database file—problems could immediately be recognized. The WAL file will vanish if any SQLite statement is conducted on the main SQLite database, and the main database will *grow*! The growth results from the addition of data from the WAL to the main database for records that are marked to be committed. Because the WAL can support concurrent-write transactions, new log records and appended log records are occurring. Because of this, the marked frame within the WAL is referenced only for a unique transaction so that any newly appended data to the WAL that may also be occurring on another thread is ignored. The problem arises when running a query on a main database with the WAL file available: A single transaction referencing a single marked frame can omit other data within the WAL file, leaving the examiner believing that all data within the WAL is now moved to the main database. This is not always the case, however, because frames that are not marked to be committed could be discarded. Also, the WAL file can grow exponentially, and because of this growth will be checkpointed. SQLite by default sets this threshold at 1000 pages, but this can be changed by the app designer using SQLite commands. Also, the WAL file uses a Shared Memory file that is located within the same directory as the WAL file, with the same name as the main database and the .shm extension. This file is used by the WAL file as an index. Because multiple processes are accessing the database concurrently, the SHM file is used to keep track of these transactions. The SHM file is joined to the WAL file, and when the WAL is created, the SHM is created; when the WAL is deleted, the SHM is also deleted. The SHM file contains no persistent data and is not generally of value to an investigation.

An examiner should make sure the solution that will be used to examine SQLite databases does not taint the database with the contents from the WAL file so a pristine snapshot-in-time image of the main database is maintained. However, the examiner must also conduct a secondary examination of the contents of the WAL file and, if available, commit the contents to the database and then find the delta information from the main database and subsequent query. The secondary examinations should always be on copies of the evidentiary files obtained from the original evidence.

A perfect example of the changes that can occur with a database file can be observed in the following figures. A Cache.db file was exported from the com.google.GooglePlus app within an iPhone 5 file system. The Cache.db file was 4KB in size, and a table schema had not been created for the SQLite database. The SQLite header was in place, but a table had not been built. Figure 13-30 shows what the database revealed in a SQLite database view—nothing. Table Schema is empty, so there are no records.

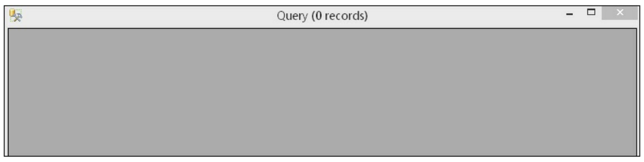
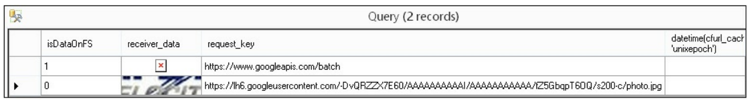

Query (0 records)

FIGURE 13-30 **A query of the Cache.db file from com.google.GooglePlus did not reveal any records.**

The main database, Cache.db, as well as the Cache.wal file, were exported into a directory, and tables became available (Figure 13-31). A query of several of the tables and their rows revealed two records, both of which had BLOB data and user data, important to the examiner.


Query (2 records)

	isDataOnFS	receiver_data	request_key	datetime(cfurl_cach 'unixepoch')
	1	☒	https://www.googleapis.com/batch	
▸	0		https://lh6.googleusercontent.com/-DvQRZZX7E60/AAAAAAAAAAI/AAAAAAAAAAA/fZ5GbqpT6OQ/s200-c/photo.jpg	

FIGURE 13-31 **A query of the Cache.db file with the associated Cache.wal file revealed records that had not been committed and are now visible.**

Note

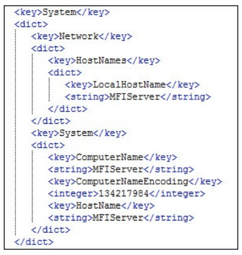
The Cache.db file grew from 4KB to 52KB, but the original Cache.wal file within the forensic solution indicated its size was 1248KB. Further investigation of the Cache.wal file using the methods outlined in this section revealed that a single page had been created 11 times within the WAL file, contributing to the size of the original.

The investigation of write-ahead logs can be just as important as examining the freelist and free pages within the main database. These files can reveal information that is not readily available within any mobile forensic solution. Some pages within the WAL file had not been added to the main database file, and some pages that had been marked inactive (deleted), upon checkpoint, could be resolved to the main database and also removed. The examination of the delta between these two files could be eye opening if done correctly.

Property Lists

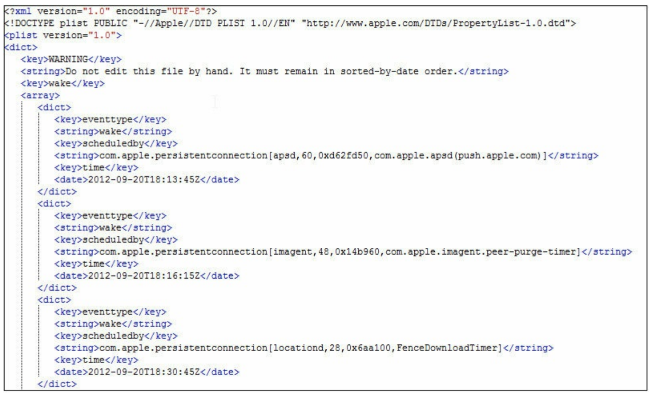
Property lists, or plists, are used in macOS and iOS devices; the filenames have the .plist extension, although the extension is not always used with files in the iOS system. Plist files store Apple configuration files and settings, application configuration files and settings, storage of history events for apps, and information that is to be stored temporarily. They are sometimes compared to the Registry within a Windows operating system. Apple describes a plist as a structured data representation to store, organize, and access standard types of data. Furthermore, data can be expressed within a plist, but only certain types of data can be within a plist: *containers* and *single values* (often called standard types). Both containers and single values are assigned to a key, which contains a value or values depending on whether the data is a container or a single value. A plist container and single value are represented by the name of the key value in brackets, <KeyOrValueName>, and ending with a closing bracket, </KeyOrValueName>. The only exception to this is the Boolean values that contain only the closing bracket at the end of the value: <true/> or <false/>.

Collections within a plist are either a dictionary or an array. A *dictionary* (Figure 13-32) contains key-value pairs. An entire single plist in actuality is a single dictionary and is often represented as the root node. The value can be obtained outside of the plist by a reference to the dictionary and key that is assigned to the value. An *array* (Figure 13-33) is a collection of values that are indexed and can be referenced to by a single key outside of the plist with an associated index or ordinal value (that is, name [0], name [1], name [2]). Array objects are contained within the single container and can be single values and dictionaries.



```
<key>System</key>
<dict>
    <key>Network</key>
    <dict>
        <key>HostNames</key>
        <dict>
            <key>LocalHostName</key>
            <string>MFIServer</string>
        </dict>
    </dict>
    <key>System</key>
    <dict>
        <key>ComputerName</key>
        <string>MFIServer</string>
        <key>ComputerNameEncoding</key>
        <integer>134217984</integer>
        <key>HostName</key>
        <string>MFIServer</string>
    </dict>
</dict>
```

FIGURE 13-32 **A dictionary contains values and keys and can also contain other dictionaries, arrays, or single independent objects.**



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>WARNING</key>
   <string>Do not edit this file by hand. It must remain in sorted-by-date order.</string>
   <key>wake</key>
   <array>
      <dict>
         <key>eventtype</key>
         <string>wake</string>
         <key>scheduledby</key>
         <string>com.apple.persistentconnection[apsd,60,0xd62fd50,com.apple.apsd(push.apple.com)]</string>
         <key>time</key>
         <date>2012-09-20T18:13:45Z</date>
      </dict>
      <dict>
         <key>eventtype</key>
         <string>wake</string>
         <key>scheduledby</key>
         <string>com.apple.persistentconnection[imagent,48,0x14b960,com.apple.imagent.peer-purge-timer]</string>
         <key>time</key>
         <date>2012-09-20T18:16:15Z</date>
      </dict>
      <dict>
         <key>eventtype</key>
         <string>wake</string>
         <key>scheduledby</key>
         <string>com.apple.persistentconnection[locationd,28,0x6aa100,FenceDownloadTimer]</string>
         <key>time</key>
         <date>2012-09-20T18:30:45Z</date>
      </dict>
```

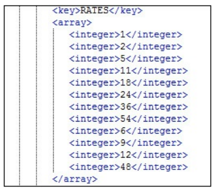
FIGURE 13-33 **An array is identified by a key and can contain multiple dictionaries or other single independent objects.**

Both arrays and dictionaries can be contained within each other. An array can be a part of a dictionary and dictionaries can be a part of an array. The examiner should understand that a collection and an array are not independent storage methods, but methods that store a list of common data in different ways to be easily referenced.

Note

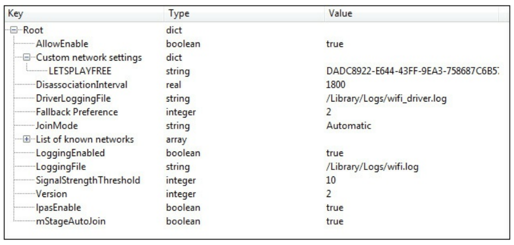
If order index matters to the values and a change to the order index will alter the results, an array should be used. A dictionary can represent a list of items, where the addition or removal of data does not change the overall value since ordering is not a concern.

Plist data also contains single values, or standard values. Single values can be independent or contained within an array or dictionary. These values must be of the following types as defined by Apple: string `<string>`, data `<data>`, date `<date>`, number `<integer>`, number `<real>`, or Boolean `<true/>` or `<false/>`. This data can be found within arrays, dictionaries, and the root node. These values are always referenced by a key-value pair unless they are contained within an array. If they are within an array, as shown in Figure 13-34, the key is defined by the array, which then will have one or more values, each indexed. Tools such as plist Editor Pro for Windows can make it easy to view a plist that contains many different types and data (see Figure 13-35).



```xml
<key>RATES</key>
<array>
   <integer>1</integer>
   <integer>2</integer>
   <integer>5</integer>
   <integer>11</integer>
   <integer>18</integer>
   <integer>24</integer>
   <integer>36</integer>
   <integer>54</integer>
   <integer>6</integer>
   <integer>9</integer>
   <integer>12</integer>
   <integer>48</integer>
</array>
```

FIGURE 13-34 **An example of an array holding several values all related and indexed to the RATES key**



Key	Type	Value
Root	dict	
AllowEnable	boolean	true
Custom network settings	dict	
LETSPLAYFREE	string	DADC8922-E644-43FF-9EA3-758687C6B5
DisassociationInterval	real	1800
DriverLoggingFile	string	/Library/Logs/wifi_driver.log
Fallback Preference	integer	2
JoinMode	string	Automatic
List of known networks	array	
LoggingEnabled	boolean	true
LoggingFile	string	/Library/Logs/wifi.log
SignalStrengthThreshold	integer	10
Version	integer	2
IpasEnable	boolean	true
mStageAutoJoin	boolean	true

FIGURE 13-35 **A list view showing the many standard values both within the root and in a dictionary using the plist Editor Pro for Windows**

Note

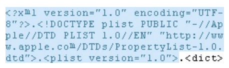
Apple recommends the use of property lists for the storage of a small amount of persistent information. If large objects are required to persist, however, property lists should not be used, and other alternatives, such as SQLite databases or other archiving files, should be used instead.

Obtaining information from the plist files within an iOS system is often automated by mobile forensic software, such as Cellebrite and Oxygen Forensic Detective, but the examiner should be extremely cautious with the displayed results. Some tools do not display the binary data within the rendered natural view. The examiner must view this data outside and independent of most forensic tools. To do that, he or she can use the free Property List Editor for Macs, available from the Apple Developer site, and for Windows, the plist Editor Pro is available from www.icopybot.com/. Both tools enable the examiner to navigate XML and binary plist files obtained from an iOS device. Using these tools can help the examiner recover data more easily than using some of the mobile forensic solutions currently available.

Property lists can be stored in three different formats: text XML, binary, or ASCII. ASCII format is seldom seen because it was used in the OpenStep framework, which Apple's Cocoa is derived from. With the release of OS X, ASCII property lists became obsolete, likely because of their read-only format. The ASCII property list storage method will not be covered in this chapter. Both text and binary property lists are covered in the following sections.

Text XML

A text XML property list is readable within an application such as Notepad or TextEdit natively, without conversion. As an examiner moves through an iOS file system, a text XML file is easily recognized by the header viewable in a HEX editor, as shown in Figure 13-36. The term "XML property list" is often used to describe the plist file, when in actuality XML is a storage format, not a type of plist. Note that a binary plist does not use XML but can be easily converted to XML to show the converted binary representation of the data in XML format!



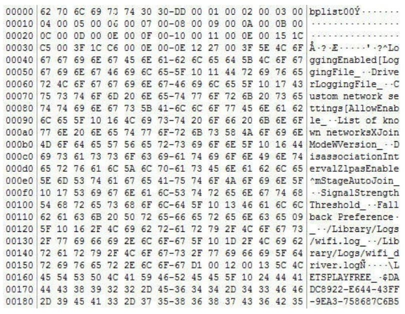
```
<?xml version="1.0" encoding="UTF-
8"?>.<!DOCTYPE plist PUBLIC "-//Ap
ple//DTD PLIST 1.0//EN" "http://ww
w.apple.com/DTDs/PropertyList-1.0.
dtd">.<plist version="1.0">.<dict>
```

FIGURE 13-36 **The header of a text XML plist is easily recognized within a HEX editor.**

Text XML property lists can be viewed and edited, and they are far more portable than a binary plist. However, when only a small amount of data is stored, a text XML property list can be much better for reads and writes. The inverse would be true: if the persistent data is required to be large, the text XML format would not be the best choice. To the examiner, this means a text XML plist will contain generally `<string>`, `<integer>`, `<real>`, `<date>`, and Boolean `<true/>` or `<false/>`. Data `<data>` values are generally not used within a text XML property list. The examiner should look into the text XML file for app settings and, most importantly, the device settings as listed in the "Additional File System Locations" section earlier in this chapter.

Binary

Binary formatted property lists are the most prevalent type used within the iOS file system, and binary is now the default format for preference files. This was not always the case, as earlier versions of iOS used the standard text XML format. This change occurred primarily because of the need to store more `<data>` within the Plist. Apple recommends using a binary property list format when data size is important. The size will negatively affect performance and speed of reads and writes if this information is stored within a text-based property list. The binary format of the property list has a common header, bplist00, which represents a binary plist. An example of the header is shown in Figure 13-37 within the com.apple.wifi.plist.



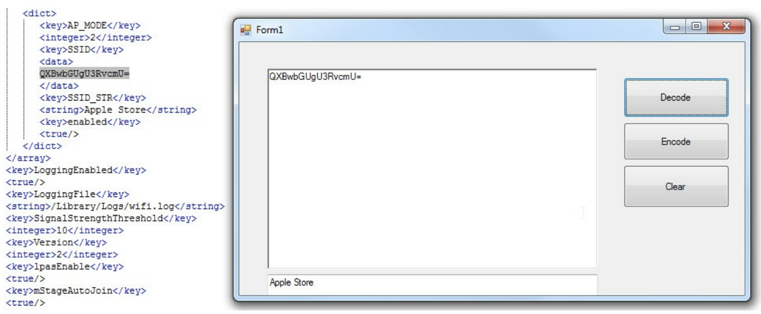
```
00000 62 70 6C 69 73 74 30 30-DD 00 01 00 02 00 03 00 bplist00Ý·······
00010 04 00 05 00 06 00 07 00-08 00 09 00 0A 00 0B 00 ················
00020 0C 00 0D 00 0E 00 0F 00-10 00 11 00 0E 00 15 1C ················
00030 C5 00 3F 1C C6 00 0E 00-0E 12 27 00 3F 5E 4C 6F Å·?·Æ·····'·?^Lo
00040 67 67 69 6E 67 45 6E 61-62 6C 65 64 5B 4C 6F 67 ggingEnabled[Log
00050 67 69 6E 67 46 69 6C 65-5F 10 11 44 72 69 76 65 gingFile_··Drive
00060 72 4C 6F 67 67 69 6E 67-46 69 6C 65 5F 10 17 43 rLoggingFile_··C
00070 75 73 74 6F 6D 20 6E 65-74 77 6F 72 6B 20 73 65 ustom network se
00080 74 74 69 6E 67 73 5B 41-6C 6C 6F 77 45 6E 61 62 ttings[AllowEnab
00090 6C 65 5F 10 16 4C 69 73-74 20 6F 66 20 6B 6E 6F le_··List of kno
000a0 77 6E 20 6E 65 74 77 6F-72 6B 73 58 4A 6F 69 6E wn networksXJoin
000b0 4D 6F 64 65 57 56 65 72-73 69 6F 6E 5F 10 16 44 ModeWVersion_··D
000c0 69 73 61 73 73 6F 63 69-61 74 69 6F 6E 49 6E 74 isassociationInt
000d0 65 72 76 61 6C 5A 6C 70-61 73 45 6E 61 62 6C 65 ervalZlpasEnable
000e0 5E 6D 53 74 61 67 65 41-75 74 6F 4A 6F 69 6E 5F ^mStageAutoJoin_
000f0 10 17 53 69 67 6E 61 6C-53 74 72 65 6E 67 74 68 ··SignalStrength
00100 54 68 72 65 73 68 6F 6C-64 5F 10 13 46 61 6C 6C Threshold_··Fall
00110 62 61 63 6B 20 50 72 65-66 65 72 65 6E 63 65 09 back Preference·
00120 5F 10 16 2F 4C 69 62 72-61 72 79 2F 4C 6F 67 73 _··/Library/Logs
00130 2F 77 69 66 69 2E 6C 6F-67 5F 10 1D 2F 4C 69 62 /wifi.log_··/Lib
00140 72 61 72 79 2F 4C 6F 67-73 2F 77 69 66 69 5F 64 rary/Logs/wifi_d
00150 72 69 76 65 72 2E 6C 6F-67 D1 00 12 00 13 5C 4C river.logÑ····\L
00160 45 54 53 50 4C 41 59 46-52 45 45 5F 10 24 44 41 ETSPLAYFREE_·$DA
00170 44 43 38 39 32 32 2D 45-36 34 34 2D 34 33 46 46 DC8922-E644-43FF
00180 2D 39 45 41 33 2D 37 35-38 36 38 37 43 36 42 35 -9EA3-758687C6B5
```

FIGURE 13-37 **Binary plist file header represented starting at offset 0**

Although binary plist files are better suited to hold `<data>` for faster read and write speeds, data values for binary plists are generally stored as Base64 values because most developers, including Apple developers, seem to prefer using Base64 for its compactness and lack of "plain textedness," which is better for security (through obscurity). Some plists using Base64 encode small files, strings, and even XML and binary plists within a `<data>` key of a property list!

Base64 is a function that enables binary data to be transferred over media, allowing only printable characters. Base64's need was generated when binary content was required to be attached to e-mails. Items such as media and files are converted to Base64 and attached per the Internet Engineering Task Force RFC 4648 Standards track from October 2006. Base64 uses values from the US-ASCII characters table (26 [a–z], 26 [A–Z], 10 [0–9]), and to round out the extra two characters for an even 64, both + and / are included. Often, the examiner will see an equal sign (=) also used within the Base64 value. This character is used for padding purposes if there are fewer than 24 characters; if the number of bits in the input stream is not a multiple of 24, padding is needed to make up the remaining bits. The extra character is not part of the encoded data, per se, but to decode the Base64 value, the padding value (=) will be needed.

Some forensic tools such as FTK and EnCase will decode the Base64 values when they are located within a plist file, but if the tool is unable to decode them, other online services and portable tools can assist with the decoding (Figure 13-38).



```
    <dict>
        <key>AP_MODE</key>
        <integer>2</integer>
        <key>SSID</key>
        <data>
        QXBwbGUgU3RvcmU=
        </data>
        <key>SSID_STR</key>
        <string>Apple Store</string>
        <key>enabled</key>
        <true/>
    </dict>
</array>
<key>LoggingEnabled</key>
<true/>
<key>LoggingFile</key>
<string>/Library/Logs/wifi.log</string>
<key>SignalStrengthThreshold</key>
<integer>10</integer>
<key>Version</key>
<integer>2</integer>
<key>lpasEnable</key>
<true/>
<key>mStageAutoJoin</key>
<true/>
```

```
QXBwbGUgU3RvcmU=
```

```
Apple Store
```

FIGURE 13-38 **Binary plist com.apple.wifi.plist encodes the SSID values as Base64, which are easily decoded to show the string value.**

Investigation and understanding of property lists and their role throughout the iOS device can be profitable for the examiner, most importantly during the documentation and testimony phase.

Miscellaneous iOS Files

iOS supports not only SQLite databases and property lists but also uses other standard files types such as text, XML, JSON, and some other proprietary files. Also, encryption has become available for individual files that pose a problem to the investigation at the file level. Nevertheless, additional files should also be investigated within an iOS file system.

Log Files

Log files within an iOS file system do not hold much user data, but they can lend an overview of the device operation. Some of the files and their definitions are listed here:

- **/diagnostics/device_info.txt** This file is located just off the root of the device file system and contains many items of interest about the device, including the phone number, serial number, phone type, the mobile equipment identifier (MEID) or International Mobile Equipment Identity (IMEI), and many more user-specific and phone-specific details. This folder is obtained only with a diagnostic collection using advanced services. This file is not generally available on an iOS 8 device collection, and the folder diagnostics are no longer observed for current devices in a raw file system or otherwise.
- **private/Library/Logs/lockdown.log** This text file shows all of the connections attempted and made to pair with the iOS device. The examiner can use this to observe connections and pairing of the device along with the dates. This file is located at private/var/logs in iOS 8 and current devices only in a raw file system (jailbroken, or otherwise) at private/var/logs as well.
- **private/Library/Logs/lockdown_service.log** This interesting text file shows the services that are accessed when the device is paired. The file shows the services that the mobile forensic solution would access on the device. This information could be important to help an examiner understand how certain files where obtained, while others were not. This file is located at private/var/logs in iOS 8 and available with a jailbroken device and current devices only in a raw file system (jailbroken, or otherwise) at private/var/logs

as well.

JSON

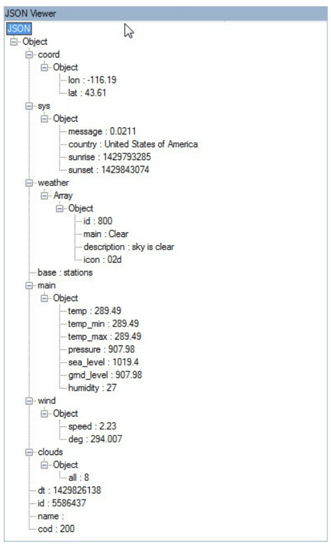
The JSON (JavaScript Object Notation) format is an open standard and uses readable text to transmit data. Much like a dictionary, JSON uses a key-value pair to store data. JSON can be used on any platform, which makes it suitable for use within an iOS device. JSON is based on standards from RFC 7159 and is used in apps within the iOS file system as a substitute for a SQLite database. Both Instagram and Facebook apps use JSON: Instagram uses a JSON file to identify media and associated text, and Facebook embeds JSON strings within fields in the main database file. Since JSON was developed for server-to-browser communication without plug-ins, it makes sense that JSON is used in these applications. Note, too, that web servers such as Google, Yahoo!, and Twitter also return JSON-formatted strings. Typically JSON will not hold media streams, but it will include the string to a local or remote URL that can be called to display the media.

JSON files, like the property list, can contain numbers, strings, Boolean values, arrays, objects (such as a dictionary of an unordered set of name-value pairs), and nulls. The object value, much like the dictionary within a plist, is the most widely used type of variable within the JSON file. The object can hold any other type of variable available in the JSON open source format and is being used more widely within apps in the iOS file system.

To recognize the JSON format, the examiner should look for the use of the curly brackets, `{` and `}`, starting and ending the file; brackets, `[` and `]`, enclosing data strings; and colons, `:`, separating names from values. The following example is a return from a server when the weather for the state of Idaho in the United States was requested:

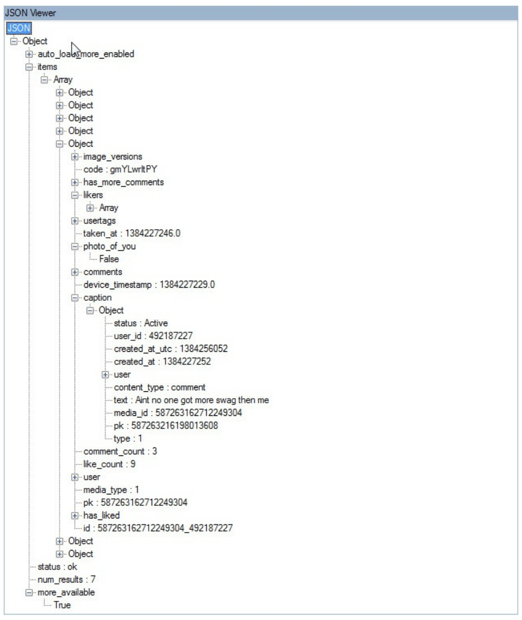
```
{
        "coord": {
                "lon": -116.19,
                "lat": 43.61
        },
        "sys": {
                "message": 0.0211,
                "country": "United States of America",
                "sunrise": 1429793285,
                "sunset": 1429843074
        },
        "weather": [{
                "id": 800,
                "main": "Clear",
                "description": "sky is clear",
                "icon": "02d"
        }],
        "base": "stations",
        "main": {
                "temp": 289.49,
                "temp_min": 289.49,
                "temp_max": 289.49,
                "pressure": 907.98,
                "sea_level": 1019.4,
                "grnd_level": 907.98,
                "humidity": 27
        },
        "wind": {
                "speed": 2.23,
                "deg": 294.007
        },
        "clouds": {
                "all": 8
        },
        "dt": 1429826138,
        "id": 5586437,
        "name": "",
        "cod": 200
}
```

If the data in the JSON is analyzed, the examiner will see several objects defined: cord, sys, weather, base, main, wind, clouds, dt, id, name, and cod. Values are associated with each object, which can be strings, arrays, numbers, other objects, and, of course, Boolean values. The file can be more easily read using a free viewer called Notepad++ and a free JSON Viewer plug-in, as shown in Figures 13-39 and 13-40.



```
JSON Viewer
JSON
  Object
    coord
      Object
        lon : -116.19
        lat : 43.61
    sys
      Object
        message : 0.0211
        country : United States of America
        sunrise : 1429793285
        sunset : 1429843074
    weather
      Array
        Object
          id : 800
          main : Clear
          description : sky is clear
          icon : 02d
    base : stations
    main
      Object
        temp : 289.49
        temp_min : 289.49
        temp_max : 289.49
        pressure : 907.98
        sea_level : 1019.4
        grnd_level : 907.98
        humidity : 27
    wind
      Object
        speed : 2.23
        deg : 294.007
    clouds
      Object
        all : 8
    dt : 1429826138
    id : 5586437
    name :
    cod : 200
```

FIGURE 13-39 **JSON file from the sample weather from the Idaho example when viewed in Notepad++**



```
JSON Viewer
JSON
  Object
    auto_load_more_enabled
    items
      Array
        Object
        Object
        Object
        Object
        Object
          image_versions
          code : gmYLwrltPY
          has_more_comments
          likers
            Array
          usertags
          taken_at : 1384227246.0
          photo_of_you
            False
          comments
          device_timestamp : 1384227229.0
          caption
            Object
              status : Active
              user_id : 492187227
              created_at_utc : 1384256052
              created_at : 1384227252
              user
              content_type : comment
              text : Aint no one got more swag then me
              media_id : 587263162712249304
              pk : 587263216198013608
              type : 1
          comment_count : 3
          like_count : 9
          user
          media_type : 1
          pk : 587263162712249304
          has_liked
          id : 587263162712249304_492187227
        Object
        Object
    status : ok
    num_results : 7
    more_available
      True
```

FIGURE 13-40 **JSON file from Instagram's fsCachedData folder decoded into a readable format in Notepad++ with the JSON Viewer plug-in**

JSON files can hold valuable information within iOS devices. An examiner should be familiar with the location and subsequent analysis of these files.

Chapter Summary

The iOS file system contains thousands of files that are examined by automated tools. With the examiner's ability to obtain the file system of an iOS device using iTunes backups, many files are already available. Special services such as file_relay and house_arrest enable some automated tools to obtain files and folders that simple iTunes backups fail to provide. Apple has disabled the file_relay service in iOS 8 devices, so files that could have been recovered in previous iOS versions are no longer available. With new methods of extraction, such as during jailbreak or using tools such as GrayKey, a complete raw file system can be available. If these choices are unavailable to the examiner, however, it doesn't mean that the examiner should abandon navigating and searching the file system.

Files such as SQLite databases contain valuable data, and free pages can be recovered with little effort. Free page data includes data no longer referenced, often referred to as inactive pages, within an active database. Within the freelist, an examiner can navigate the page and identify these inactive pages—often recovering valuable deleted data. Often the default in SQLite version 3.7x database files, the write-ahead log holds valuable data that enters the WAL file prior to entering the main database. Automated tools often miss the most recent data in the main database, but this data is still within the WAL file. Using the knowledge and suggestions in this chapter, the examiner can recover and display this information, leading to results far superior to those most automated forensic tools provide. Because a WAL file can also contain frames that are marked inactive, deleted data can also be recovered, often identified by automated tools as undeleted. Having the skills to determine where the data actually resides within the main database or WAL file is critical to the examiner.

iOS devices contain many other files, including property lists, or plist files, which are generally either text XML or binary files. The plist files within an iOS device include great information detailing settings, configurations, user data, and passwords. Many forensic tools support these files natively, but some do not. The examiner should be sure to identify the support of plist files by the mobile forensic solution and whether or not the viewer within the solution displays all of the information. Some tools will show that there is binary data within a key, but they will not display this data, such as FTK. The examiner should use tools such as the Property List Editor for Macs and plist Editor Pro for Windows to dive into the property lists from iOS devices. Both of these tools enable the examiner to rummage through both text- and binary-based property lists, showing binary data within keys and more. Many other text files and settings files can be found within the file system of an iOS device that can show not only device information and diagnostics but also user information that can be important for any investigation.

The examiner can use the tables within this chapter to help develop a course of action. Using the information on file location and file types can help the examiner peel the many layers of the iOS onion, exposing the many artifacts often overlooked by automated solutions.

14 Querying SQLite and Taming the Forensic Snake

Chapter 13 covered the various files within the iOS file system, and Chapter 15 will deal with files within the Android file system. This chapter introduces the methods for scraping, carving, infiltrating, and exposing the maximum amount of data from iOS and Android mobile devices—or any device with which the examiner has access to files and folders. With repetition, practice, and desire, an examiner can uncover, dissect, and parse critical data that an automated tool can miss.

In this chapter, SQLite database parsing will be demystified to offer the examiner a better understanding of how automated tools pull data from these files. It includes instructions on creating queries within apps that may (or may not) be supported by an automated tool that may be missing critical data or data types. SQLite database files are the primary source for today's smart devices, including iOS, Android, Windows Phone, and BlackBerry 10 devices. These files, along with their cache files, contain an enormous amount of evidence. Because of the many devices supporting this file type, the treasure trove of information about SQLite provided in this chapter can be helpful for the examiner involved in any smart device collections that involve these files.

Thousands of files are contained within a smart device file system, and the examiner should by now recognize that automated tools cannot support every file type, dissecting each file and displaying its contents. Because of this impossibility, the examiner should be versed in how to obtain that information without having to document the contents of the file manually by photographing the mobile device or printing the screen containing the data. Instead, the examiner can automate the process by creating custom-built SQL queries or custom-built code using Python, a free, open source scripting language that is making its way into many forensic tools.

Python enables an examiner to build file parsers, file identifiers, file searchers, and much more, removing the bottleneck that can be created by the process of software design, from implementation to release. An examiner can now create, in as few as eight lines of code, support for a file type within an app that would have taken a software company two months to release to the public.

Technological power must shift back to the examiner because of tech growth volatility. A software company releases fixes and updates to apps that have been updated three times prior to the solution's support release. Essentially, a forensic solution's release is built on a previous version of an app and as such can be completely useless to the examiner. Given the tools and information, the examiner can supplement the automated tools to fill the gap between solution release and app proliferation. This is how a mobile forensic expert is born.

Querying the SQLite Database

Probably the most important skill for an examiner is his or her ability to analyze a SQLite database. Of course, there are automated tools that will query and recover information from these file types, but what if the data is not retrieved, is in an improper format, or the particular app database is not directly supported? With the approximately 6.2 million available apps for iOS, Android, Windows Phone, and BlackBerry 10 devices, an examiner needs to have advanced skills to open the associated SQLite databases, examine the contents, and create queries to recover the needed data.

Many tools are available for both Windows and Mac devices that can enable an examiner to view the contents of SQLite files and extract targeted information to produce data for any investigation. An examiner's reliance on an automated tool to recover app data consistently will be short lived, however—not because the automated tools are not up to par, but because the forensic software companies are aiming their products at moving targets. Apps can be updated weekly or monthly, so keeping up with the database schema changes, data value formats, and addition of functionality is an almost impossible task. To combat that, the examiner must be able to traverse the SQLite database and build queries to stay current with even the "zero-day" apps.

What Is a SQL Query?

SQLite, as mentioned earlier in the chapter, is a structured language and, as such, SQLite understands most of the standard SQL queries and uses a few of its own. Communicating with a SQLite database is completed by formulating a structured query using a properly formatted statement. Several types of queries can be used to create, delete, and alter tables (these will not be covered in this book because they will not be needed for forensic investigations). An examiner querying SQLite databases should be aware of several basic statements shown in Table 14-1 that are often used "under the hood" of forensic software. Other statements not covered in this book can be found at SQLite.org.

TABLE 14-1 **SQL Syntax for the Investigation of App Data Within SQLite Databases**

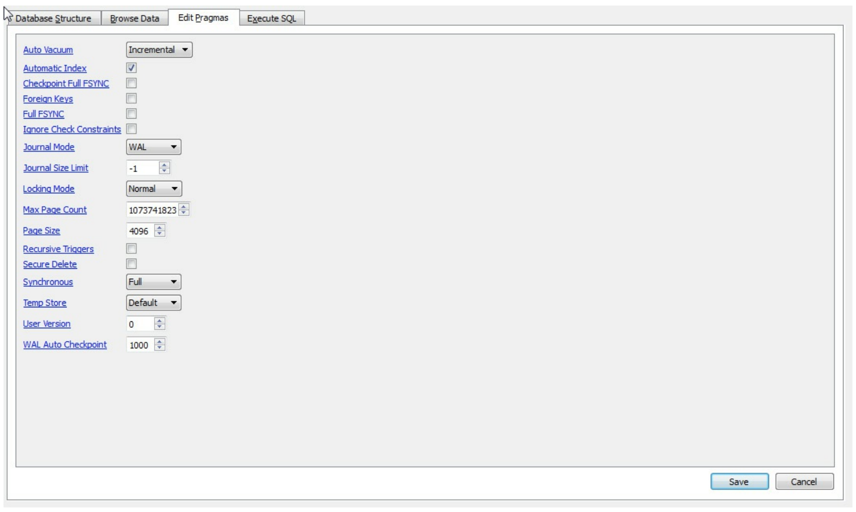
Statement	Description
`SELECT`	Used to query the database. A `SELECT` statement does not make any changes to the database, and the result is 0 or more rows of data.
`FROM`	Used to direct the `SELECT` statement to the location the data should be queried `FROM`.
`INNER JOIN`	The default `JOIN` used in SQL, which will select records only from the database tables that have matching values.
`LEFT OUTER JOIN`	The only `OUTER JOIN` that can be used in SQLite, which returns `ALL` records from the table on the left of the `JOIN` statement, plus the matched records from the table on the right of the `JOIN` statement. `OUTER` can be omitted from the statement. However, SQLite will ignore the `LEFT JOIN` statement if `ON` or `USING` is used instead of `WHERE` and treat it like an `INNER JOIN`.
`WHERE`	This condition is used to specify whether an expression or column in a table has a relationship with another column or expression. The standard operands can be used (=, >, >=, <, <=). `WHERE` shows relationships between tables and columns, particularly an index, and is also used to filter data.
`ON`	This condition executes `ON` the `JOIN` statement and indicates how the `JOIN` should be performed, unlike `WHERE`, which is a filter of the collected data from the table.

If an investigation of the SQLite database will occur outside of the forensic environment, the examiner should use a compiled SQL browser because of communication requirements with the SQLite database. Windows does not natively communicate with a SQLite database unless the SQLite engine is installed, available, and used for communication; using a compiled application, either portable or installed, will be much easier. (Alas, if the examiner wants to communicate at the command prompt with the database, that is also easily done—but is not covered in this book—and the sqlite3 library is needed.)

Note

Of course, prior to communicating with the SQLite database, the sqlite3 library (www.sqlite.org/download.html) must be available or used by any software the examiner will use to complete the navigation of the database. The most current and up-to-date open source browser for SQLite databases is available without cost on the DB Browser for SQLite site (http://sqlitebrowser.org). The examiner should become familiar with the DB Browser for SQLite and the exposed SQLite database using this tool, which is supported by Windows, Mac, and Linux. This robust software will allow for navigating, viewing, query building and saving, as well as exporting the executed query data. The following section outlines the examination of a database file within DB Browser for SQLite in an effort to convey the usefulness of the tool.

Looking into a database after being exposed to the interworking and file metadata, an examiner should be able to understand the information displayed by the automated tool. As you examine the figures in this chapter, you will see that the information identified within the DB Browser GUI includes the kind of journaling used and the size of the page in bytes, as shown in the Edit Pragmas tab in Figure 14-1. If necessary, the examiner could find this information manually, now armed with the knowledge of its location in the database file.



Auto Vacuum	Incremental
Automatic Index	✓
Checkpoint Full FSYNC	☐
Foreign Keys	☐
Full FSYNC	☐
Ignore Check Constraints	☐
Journal Mode	WAL
Journal Size Limit	-1
Locking Mode	Normal
Max Page Count	1073741823
Page Size	4096
Recursive Triggers	☐
Secure Delete	☐
Synchronous	Full
Temp Store	Default
User Version	0
WAL Auto Checkpoint	1000

FIGURE 14-1 **The Edit Pragmas tab, where you can see and edit information about the database**

While using a database viewer, the examiner can view tables and rows as a hierarchy and concentrate on identifying the relationships between the tables as defined in the schema. SQLite databases within apps will include some information in one table, more in another, and sometimes even more in a third. Being able to locate these relationships is important to the examiner, and using a GUI to identify *keys* within a database can assist when these tables need to be joined in an effort to merge the tables using their relationships. Keys define the relationships within a database, and using a key within a query will enable the JOIN to be successful and show all the related data. There are two types of keys: *primary keys* and *foreign keys*.

Caution

DB Browser for SQLite can alter, create, delete, add, and modify the SQLite database with features included within the product. Before using it on an actual evidence file, examiners should work only on a copy of the evidence file and learn about the feature set by reading the application documentation.

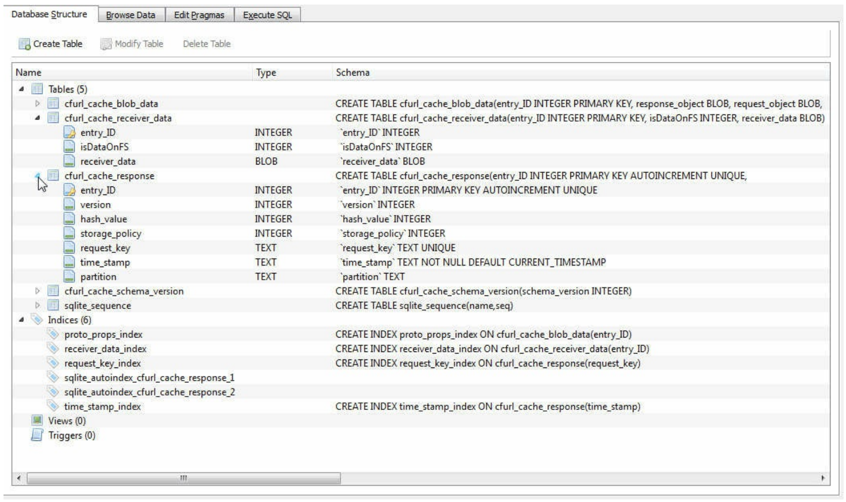
Primary keys are unique and singular, and only one can be assigned within a single table, although the table is not required to have a primary key. The primary key is generally an integer primary key and can often contain an empty column (NULL) if the table does not contain a ROWID (rowid). Most SQLite databases will use ROWID (rowid) or a derivative of ROWID to indicate the primary key. Within a database viewer, the primary key is often shown graphically as a key icon next to the table. The primary key is used by the database to create tables and to make updates and other changes to the database to ensure that information is not overwritten.

Tip

To maintain uniqueness, an examiner should look at the ROWID in iOS databases. A gap in the ROWID indicates that a row has been deleted, because the ROWID is never duplicated, to maintain uniqueness within the table.

Foreign keys, on the other hand, are references that show the relationship *between* tables. You can observe this relationship when a value in the parent table is equal to a value in the associated child table. You can also identify a foreign key within a SQLite database table if a column contains one or more NULL values, which indicates that the column is a foreign key per SQLite standards (this will also be reported by database viewers). The examiner will use the related values between existing parent and child tables to build JOINs within a SQL query.

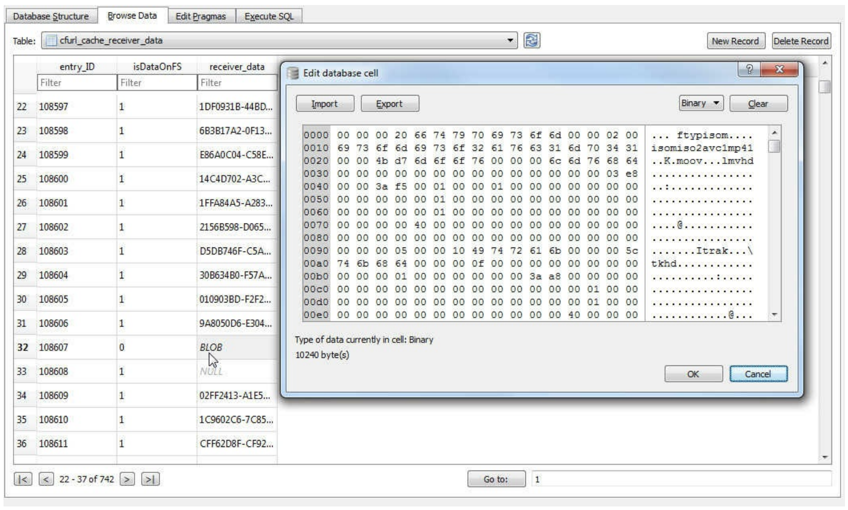
When constructing and viewing app databases, the examiner should pay special attention to foreign keys by first checking the tables and verifying the relationships prior to creating a SQL query. This information is available in the Database Structure tab, shown in Figure 14-2. Some keys may be related to other tables, and on many occasions, a field (column) is not identified as a key but can be used as such to create stellar results.



Name	Type	Schema
Tables (5)		
cfurl_cache_blob_data		CREATE TABLE cfurl_cache_blob_data(entry_ID INTEGER PRIMARY KEY, response_object BLOB, request_object BLOB,
cfurl_cache_receiver_data		CREATE TABLE cfurl_cache_receiver_data(entry_ID INTEGER PRIMARY KEY, isDataOnFS INTEGER, receiver_data BLOB)
entry_ID	INTEGER	\`entry_ID\` INTEGER
isDataOnFS	INTEGER	\`isDataOnFS\` INTEGER
receiver_data	BLOB	\`receiver_data\` BLOB
cfurl_cache_response		CREATE TABLE cfurl_cache_response(entry_ID INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
entry_ID	INTEGER	\`entry_ID\` INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE
version	INTEGER	\`version\` INTEGER
hash_value	INTEGER	\`hash_value\` INTEGER
storage_policy	INTEGER	\`storage_policy\` INTEGER
request_key	TEXT	\`request_key\` TEXT UNIQUE
time_stamp	TEXT	\`time_stamp\` TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
partition	TEXT	\`partition\` TEXT
cfurl_cache_schema_version		CREATE TABLE cfurl_cache_schema_version(schema_version INTEGER)
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (6)		
proto_props_index		CREATE INDEX proto_props_index ON cfurl_cache_blob_data(entry_ID)
receiver_data_index		CREATE INDEX receiver_data_index ON cfurl_cache_receiver_data(entry_ID)
request_key_index		CREATE INDEX request_key_index ON cfurl_cache_response(request_key)
sqlite_autoindex_cfurl_cache_response_1		
sqlite_autoindex_cfurl_cache_response_2		
time_stamp_index		CREATE INDEX time_stamp_index ON cfurl_cache_response(time_stamp)
Views (0)		
Triggers (0)		

FIGURE 14-2 **Once the database is imported into the application, the Database Structure tab shows the tables and keys associated with a column; these are important for joining tables.**

BLOB (Binary Large Object) data is used in many SQLite databases to store binary data representing large objects (such as images and files) not to exceed 65,535 bytes. Typically, the BLOB data will be displayed as thumbnails of images, small video clips, or text. Having a tool to look at the BLOB data (Figure 14-3) can be extremely important to the examiner, since most mobile forensic solutions do not resolve the BLOB data. If binary files are too large to be stored efficiently within the database, they will be stored to disk and then referenced within the BLOB data. The examiner should carefully examine BLOB data, because this might be the only location with important images, videos, or text from the mobile device.



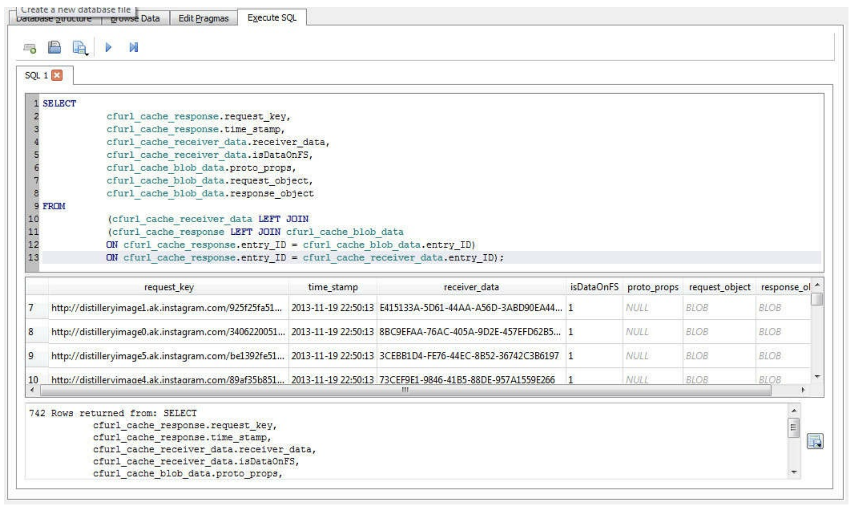
	entry_ID	isDataOnFS	receiver_data
	Filter	Filter	Filter
22	108597	1	1DF0931B-44BD...
23	108598	1	6B3B17A2-0F13...
24	108599	1	E86A0C04-C58E...
25	108600	1	14C4D702-A3C...
26	108601	1	1FFA84A5-A283...
27	108602	1	2156B598-D065...
28	108603	1	D5DB746F-C5A...
29	108604	1	30B634B0-F57A...
30	108605	1	010903BD-F2F2...
31	108606	1	9A8050D6-E304...
32	108607	0	BLOB
33	108608	1	NULL
34	108609	1	02FF2413-A1E5...
35	108610	1	1C9602C6-7C85...
36	108611	1	CFF62D8F-CF92...

```
0000 00 00 00 20 66 74 79 70 69 73 6f 6d 00 00 02 00  ... ftypisom....
0010 69 73 6f 6d 69 73 6f 32 61 76 63 31 6d 70 34 31  isomiso2avc1mp41
0020 00 00 4b d7 6d 6f 6f 76 00 00 00 6c 6d 76 68 64  ..K.moov...lmvhd
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 e8  ................
0040 00 00 3a f5 00 01 00 00 01 00 00 00 00 00 00 00  ..:.............
0050 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ................
0060 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ................
0070 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00  ....@...........
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090 00 00 00 05 00 00 10 49 74 72 61 6b 00 00 00 5c  .......Itrak...\
00a0 74 6b 68 64 00 00 00 0f 00 00 00 00 00 00 00 00  tkhd............
00b0 00 00 00 01 00 00 00 00 00 00 3a a8 00 00 00 00  ..........:.....
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00  ................
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00  ................
00e0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 00  ............@...
```

Type of data currently in cell: Binary
10240 byte(s)

FIGURE 14-3 **Looking into the table cfurl_cache_receiver_data, the examiner can click any record labeled BLOB and the data will be exposed in a separate window, as shown with a video file that can be exported and viewed within a native viewer.**

Once an examiner has located the tables and significant columns within the table, any associated foreign keys between tables and child tables should be ascertained. A SQL query should then be created outlining the significant tables and their rows, the related table columns (foreign key), and either a reference or filter. The SQL statement can then be executed on the database. The SQLite engine will parse the database based upon the built statement and produce the results. As shown in Figure 14-4, the examiner can manually input, paste, or load a saved SQL statement into the Execute SQL tab to execute the script.



```
SELECT
            cfurl_cache_response.request_key,
            cfurl_cache_response.time_stamp,
            cfurl_cache_receiver_data.receiver_data,
            cfurl_cache_receiver_data.isDataOnFS,
            cfurl_cache_blob_data.proto_props,
            cfurl_cache_blob_data.request_object,
            cfurl_cache_blob_data.response_object
FROM
            (cfurl_cache_receiver_data LEFT JOIN
            (cfurl_cache_response LEFT JOIN cfurl_cache_blob_data
            ON cfurl_cache_response.entry_ID = cfurl_cache_blob_data.entry_ID)
            ON cfurl_cache_response.entry_ID = cfurl_cache_receiver_data.entry_ID);
```

	request_key	time_stamp	receiver_data	isDataOnFS	proto_props	request_object	response_ol
7	http://distilleryimage1.ak.instagram.com/925f25fa51...	2013-11-19 22:50:13	E415133A-5D61-44AA-A56D-3ABD90EA44...	1	NULL	BLOB	BLOB
8	http://distilleryimage0.ak.instagram.com/3406220051...	2013-11-19 22:50:13	8BC9EFAA-76AC-405A-9D2E-457EFD62B5...	1	NULL	BLOB	BLOB
9	http://distilleryimage5.ak.instagram.com/be1392fe51...	2013-11-19 22:50:13	3CEBB1D4-FE76-44EC-8B52-36742C3B6197	1	NULL	BLOB	BLOB
10	http://distilleryimage4.ak.instagram.com/89af35b851...	2013-11-19 22:50:13	73CEF9E1-9846-41B5-88DE-957A1559E266	1	NULL	BLOB	BLOB

```
742 Rows returned from: SELECT
            cfurl_cache_response.request_key,
            cfurl_cache_response.time_stamp,
            cfurl_cache_receiver_data.receiver_data,
            cfurl_cache_receiver_data.isDataOnFS,
            cfurl_cache_blob_data.proto_props,
```

FIGURE 14-4 **Results of the executed SQL statement**

Using a GUI to navigate a SQLite database, an examiner can uncover much more information than is available by relying on a software engineer to decide what is important in a SQLite database. No automated tool supports the many cache.db files within an iOS device, and as shown in Figure 14-4, a lot of additional information gathered using a SQL query can assist any mobile forensic examination.

Building a Simple SQL Query

Building a query is not the most difficult thing to do, but some examiners believe it may not be the best use of their time. For those examiners, there are automated tools that build SQL queries with just a few clicks of the mouse. Others will appreciate some basic queries covered within this section to help them understand the compilation and execution of a statement on a database.

The easiest queries can be formulated using some of the following examples. Note that data shown within brackets (< >) are variables the examiner would fill in with a unique name from the database being examined.

The first query starts with a `SELECT` and asks the database to retrieve ALL (`*`) of the rows from a table the examiner specifies after the `FROM` statement:

```
SELECT * FROM <TABLENAME>;
```

The next query asks for four columns within the table. The data from the four columns will be retrieved and displayed:

```
SELECT <column1>,<column2>,<column3>,<column4> FROM <TABLENAME>;
```

The next query will obtain all the records from a single table, but will show the matching records only if column 4 contains the string `Lee`. This query can help the examiner search for records with a column and return only data that contains that record.

```
SELECT * FROM <TABLENAME> WHERE <column4> = 'Lee';
```

The next query will obtain records from two different tables within the same database and combine the results. The statement requests the data from columns 1 and 2 in both table 1 and table 2 if the information in column 3 of table 1 matches the information in column 4 in table 2. This statement is extremely powerful since most apps will be using multiple tables.

```
SELECT <column1.TABLE1>,<column2.TABLE1>,<column1.TABLE2>,<column2.TABLE2>
FROM <TABLENAME1> INNER JOIN <TABLENAME2> ON <column3.TABLE1> = <column4
.TABLE2> ;
```

Knowing how to build SQL queries will help the examiner not only in creating his or her own queries but also in understanding how an automated tool uses queries to obtain information from a database. Furthermore, an examiner can save the template SQL queries and simply change table names, column names, and functions and reuse the queries on many apps within the iOS device and any other media where a SQLite database is located.

There are some limitations, of course, to these examples. What if the data is stored in UNIX, MICROSECONDS, WEBKIT, MILLISECONDS, MAC DATE, or other date formats? In such cases, the SQL statement will probably be more complex, especially if the examiner is assigning formulas to rows—but writing such a query is not impossible, and more about this is covered in the next section.

Once the examiner gets more comfortable building queries, he or she can build queries that extract selected data from any SQLite database. Following are some examples of queries that can work with iOS databases for two popular apps: WhatsApp and Kik Messenger.

The WhatsApp query will extract the messages, any attachment data information, and associated metadata and match it with the username of the sender. This first query starts with the `SELECT` command and then the `<TABLENAME>.<COLUMNNAME>` from the database `FROM` the `<TABLENAME>` and `<TABLENAME>` based upon the primary keys of the `SELECTED TABLES`.

```
SELECT
      ZWAMESSAGE.ZFROMJID,
      ZWAMESSAGE.ZPUSHNAME,
      ZWAMESSAGE.ZTEXT,
      ZWAMESSAGE.ZISFROMME,
      ZWAMESSAGE.ZTOJID,
      ZWAMEDIAITEM.ZMEDIAURL,
      ZWAMEDIAITEM.ZMEDIALOCALPATH
FROM
      ( ZWAMESSAGE LEFT JOIN ZWAMEDIAITEM
      ON ZWAMEDIAITEM.Z_PK = ZWAMESSAGE.Z_PK);
```

The next query for Kik Messenger will extract valuable message data, convert date and time information, and match the associated data with the username of the sender. This query is similar to the preceding query, but the `datetime` function converts the `unixepoch` date to a normalized date:

```
SELECT
     datetime(ZKIKMESSAGE.ZTIMESTAMP + 978307200, 'unixepoch'),
     datetime(ZKIKMESSAGE.ZRECEIVEDTIMESTAMP + 978307200, 'unixepoch'),
     ZKIKMESSAGE.ZBODY,
      ZKIKUSER.ZDISPLAYNAME,
      ZKIKUSER.ZEMAIL,
      ZKIKUSER.ZFIRSTNAME,
      ZKIKUSER.ZLASTNAME
FROM
      (ZKIKUSER INNER JOIN ZKIKMESSAGE
      ON ZKIKMESSAGE.ZUSER = ZKIKUSER.Z_PK);
```

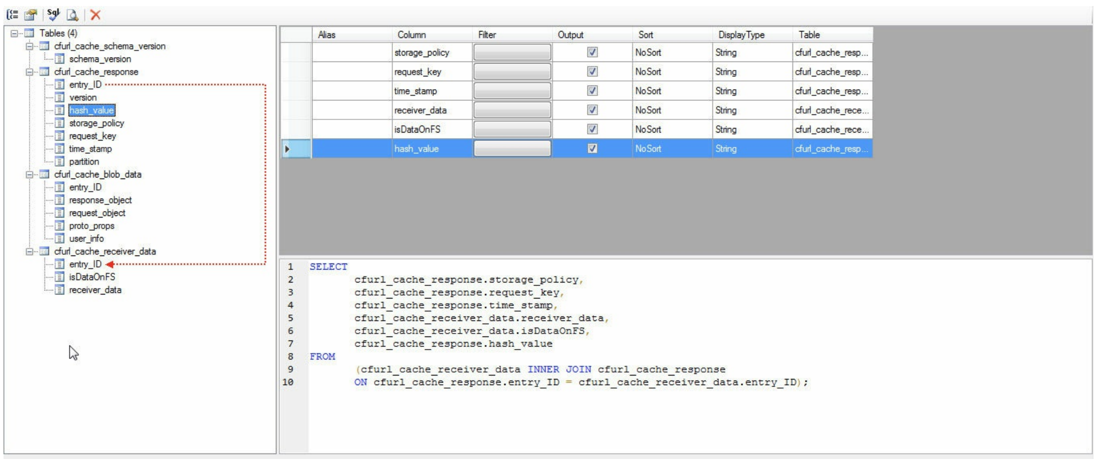
Automating Query Building

An examiner can manually build queries within many SQL database browsers. As mentioned, time is sometimes against many examiners as they perform forensic examinations; they might not have the time to learn, dive into, or formulate their own queries. These examiners tend to rely on automated tools to extract data from SQLite databases or apps that come their way. There are just too many apps in the world today—with no end in sight—to rely on commercial tools. For these examiners, and even those who feel comfortable writing queries, there are automated query building tools available within many mobile forensic tools and in standalone products.

For example, Oxygen Forensic Detective and Cellebrite Physical Analyzer include a SQL builder that enables an examiner to build queries based upon the selection of the table and then the rows. Using a SQL builder, the examiner can assign a data format (such as a String, Date/Time, Duration) and create a SQL query without having to write a single line of SQL code.

Paul Sanderson, from Sanderson Forensics, created a tool called Forensic ToolKit for SQLite, which is similar to both the Oxygen Forensic and Cellebrite tools. It also enables browsing, write-ahead-log reviewing, and SQLite navigation. This tool can be used to verify information discussed within this chapter, such as exploration of pages, frames, and byte-level information with a built-in HEX view.

A third tool that automates query building is SQLQuery Builder (Figure 14-5). Using this tool, the examiner drags and drops, double-clicks, and joins tables by dragging the parent table to the child table to form the relationship. Columns can be added to the query or entire tables by simply moving them into the query build frame. There is no limit on the foreign keys that can be joined between multiple tables within a single database, and, when completed, the SQL query can be executed and results reported.

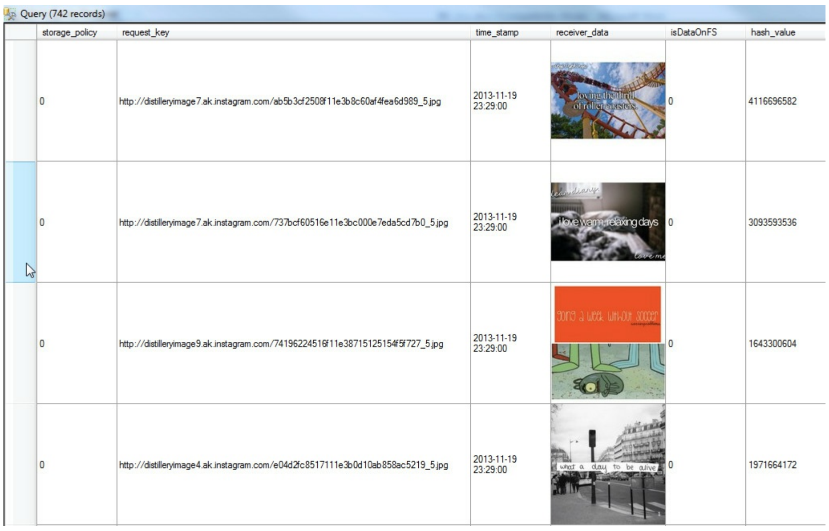


Alias	Column	Filter	Output	Sort	DisplayType	Table
	storage_policy		✓	NoSort	String	cfurl_cache_resp...
	request_key		✓	NoSort	String	cfurl_cache_resp...
	time_stamp		✓	NoSort	String	cfurl_cache_resp...
	receiver_data		✓	NoSort	String	cfurl_cache_rece...
	isDataOnFS		✓	NoSort	String	cfurl_cache_rece...
	hash_value		✓	NoSort	String	cfurl_cache_resp...

```
SELECT
        cfurl_cache_response.storage_policy,
        cfurl_cache_response.request_key,
        cfurl_cache_response.time_stamp,
        cfurl_cache_receiver_data.receiver_data,
        cfurl_cache_receiver_data.isDataOnFS,
        cfurl_cache_response.hash_value
FROM
        (cfurl_cache_receiver_data INNER JOIN cfurl_cache_response
        ON cfurl_cache_response.entry_ID = cfurl_cache_receiver_data.entry_ID);
```

FIGURE 14-5 **A complex query between two tables with multiple rows using a foreign key can be completed in a short time using automated query builders such as SQLQuery Builder.**

These tools will assist the examiner who does not have the time to create his or her own queries, but they also help the examiner gain a better understanding of what it takes to create a SQL query. These automated query builders also help the examiner not to miss or misidentify data—or give up for lack of support by mobile forensic solutions. In Figure 14-6, the results of the query of a database show what current mobile forensic solutions do not offer: by using SQL queries, an examiner can recover any type of data from most SQLite databases.


Query (742 records)

	storage_policy	request_key	time_stamp	receiver_data	isDataOnFS	hash_value
	0	http://distilleryimage7.ak.instagram.com/ab5b3cf2508f11e3b8c60af4fea6d989_5.jpg	2013-11-19 23:29:00	loving the thrill of roller coasters.	0	4116696582
	0	http://distilleryimage7.ak.instagram.com/737bcf60516e11e3bc000e7eda5cd7b0_5.jpg	2013-11-19 23:29:00	i love warm, relaxing days	0	3093593536
	0	http://distilleryimage9.ak.instagram.com/74196224516f11e38715125154f5f727_5.jpg	2013-11-19 23:29:00	going a week without soccer	0	1643300604
	0	http://distilleryimage4.ak.instagram.com/e04d2fc8517111e3b0d10ab858ac5219_5.jpg	2013-11-19 23:29:00	what a day to be alive	0	1971664172

FIGURE 14-6 **Results of a SQL query**

The analysis of SQLite databases is critical during any mobile device examination. An examiner who understands SQLite structures, the data they can hold, and the various types of data formats and hidden data within free pages is a necessity for examining today's smart devices. With so many apps available from the iTunes Store and Google Play, having this knowledge and the ability to harvest these artifacts is essential for any good examiner.

Analysis with Python

Mobile devices contain massive amounts of data, and as you know, mobile forensic solutions cannot extract every app's data or parse every file for its data. For an examiner to be successful and eliminate such obstacles, he or she will have to think outside of the box, forget about the limitations imposed by a software solution, and start thinking of ways to parse the information from these valuable files. A mobile forensic examiner today must believe that *in order to be prepared for tomorrow, you have to think about tomorrow today*.

In the early days of mobile forensic examination, data on mobile device file systems was usually parsed manually. Most early forensic examiner training focused on extracting data, manually locating the data, converting it into a readable format, and finally making the data presentable for court. As device storage grew, these tasks became extremely time consuming, often taking weeks for a first-generation iPhone. Forensic solution providers struggled to release updates to their software to deal with the growing number of smart devices. To make the best use of their limited time, examiners needed a way to automate the repetitive task of manually parsing data that was not supported by mobile forensic solutions.

Enter Python, the most widely used scripting language in mobile forensics today. Using Python, examiners can create individual "programs" to act upon a file, folder, image, or really anything stored on a mobile device. Python scripts act as parsers to complete the task the examiner previously undertook manually.

Python was created in 1989 by Guido von Rossum, who also helped create the ABC language, a precursor to Python. Python also has roots in the C programming language. It was targeted at nonprofessional programmers with its ease of readability, many supported functions, modules, and cost. Python is 100 percent free, and because of its many contributors, it has a large repository of modules for building powerful scripts with little effort. Python is currently at version 3.4.x and 2.7.x, which are available on the www.python.org web site. The most popular version of Python is still version 2.7, and most available scripts in the community are written for that version. Version 2.7 does not contain many of the new modules introduced in Python 3, however, which makes Python 2.7 and 3.x not cross-compatible; modules are version specific, and running a 2.7 script against a 3.x interpreter will fail. In the following section and all included examples, Python 3.x is used.

Note

If the examiner is looking to keep current with modules and support, using Python 3.x is recommended. If a software solution has built-in Python support, such as in Cellebrite Physical Analyzer, it is important to read the documentation to determine what version of Python the solution supports. Because Python 2.7 and 3.x are not 100 percent compatible, understanding the version of Python supported by the tool is important.

Python can be used in forensics to recover volatile memory from mobile devices, recover files, decrypt files, parse strings and dates from a file, recover passcodes from a binary image, pull exchangeable image file format (EXIF) data from photos taken with the device camera, and so much more. The things a Python script can do are limited only by the examiner's imagination. If an examiner becomes familiar with the creation and execution of a Python script, the limitation of forensic tools will be negligible and the script will be invaluable.

This section is not intended to teach every aspect of Python, but to introduce the concept of using scripts to automate processes and supplement the mobile forensic solutions available. If you are looking for additional information on learning Python, the www.python.org web site offers guides for both nonprogrammers and programmers and is a wealth of information.

Tip

Python binaries can be downloaded from www.python.org/downloads. To use Python, the interpreter must be installed prior to running a script. However, if scripts are being created to be used within a forensic tool such as Physical Analyzer, the interpreter does not need to be installed. Both forensic solutions have built-in interpreters and have installed their respective Python bundles.

Python Terminology

An examiner will hear many terms when getting involved with a programming language. These terms describe actions and data types and are also important to know while creating statements within the code. Of course, all terms used in Python cannot be included within this section, but terms that are commonly seen, heard, and used are included here. These terms will be used throughout the section to create scripts that complete tasks the examiner can immediately use in a mobile forensic examination.

Functionality Terms

Python's syntax can explain the code in comments, load common libraries, and commence and end the executed code. Common syntax that anyone conducting mobile forensic exams using Python should understand are listed next.

Comments Python and any other programming language can become difficult to read, especially when the code contains many lines. To help with readability, an author can annotate the code with comments to assist with the interpretation. A statement is not processed by the interpreter as long as the comment annotation (#) precedes the comment.

```
# This is to instruct the reader the code below prints "Mobile Forensics
Rocks!"
print('Mobile Forensics Rocks')
```

Expressions When values, variables, and operators are combined, an expression is formed.

```
x = 11
x + 6
lastname = 'Reiber'
```

Operators Symbols are used to represent some type of computation in an expression. Examples of operators are addition (+) subtraction (−), multiplication (×), and division (/) operators.

Statements In Python or any other programming language, a statement is a unit of code that can be executed. A statement can comprise a single word, multiple words, a single line, or many lines.

Variables All programming languages, including Python, use a name to represent a value. This value can be assigned a name, and the name can be used to refer to the value throughout the script. In this example, `x` and `lastname` are both variables: `x` holds the integer `11` and `lastname` holds the string `'Reiber'`.

```
x = 11
x + 6
lastname = 'Reiber'
```

Values Values are the represented data within a variable and the data with which a script is working. Data is represented by values, and values are represented as a type.

Data Types

The data type or value will tell Python how the data is to be handled and how expressions should be evaluated. Python is different from some other popular programming languages, because the type of value or variable does not have to be declared; Python just knows. In some languages, such as C#, a variable must be declared and a type assigned prior to use, as shown in this C# example:

```
string lastname = 'Reiber';
```

The Python equivalent does not have to declare the type; the interpreter knows the value is a string because of the quotation marks:

```
lastname = 'Reiber'
```

Python's data types are listed here with some examples.

Boolean A Boolean value is either a true or false value. A Boolean is used to make a decision. The Python interpreter evaluates a statement, determines the Boolean value of the statement, and then acts as instructed in the code depending on whether the statement is true or false.

```
11==11
```

This is true (the == tests for equality).

```
11==1
```

This is false.

```
11!=11
```

This is false (the `!=` tests for inequality).

```
11!=1
```

This is true.

String A string is text that contains characters—numbers, letters, spaces, and anything else as long as it is surrounded by double (`"string"`) or single (`'string'`) quotation marks.

```
print ('Python can be good for forensics')
```

Integer Numbers without a decimal point are integers. Integers are recognized by Python in a statement because they are not surrounded by quotation marks.

Float Numbers with a decimal point are integers. Floats are also recognized by Python in a statement because they are not surrounded by quotation marks.

List Lists are represented to Python enclosed within brackets: `[ ]`. A list is used in Python when ordering is important and an index is needed to represent a store of data that can include any Python type. The index within a list always starts at 0, so the first value in a list would be referenced by [0].

```
phone_list = ['android', 'iOS', 'WindowsPhone', 11, 9518, 'BlackBerry']
print (phone_list[2])
```

The result would be

```
'WindowsPhone'
```

The result would be

```
'WindowsPhone'
```

Dictionary Much like lists, dictionaries store multiple types of data within a single listing, but instead of using ordering by an index like a list, a dictionary uses a key-value pairing system. Dictionaries are represented in Python using braces: `{ }`. Each key can have only one value, but multiple keys can store the same value. The key and value within a dictionary are separated using a colon (`:`), and values can be accessed with the key.

```
addressbook = {'firstname': 'Bob', 'lastname':'Smith','street': '104 S
Overland']
print (addressbook.get('street'))
```

The result would be

```
'104 S Overland'
```

The result would be

```
'104 S Overland'
```

Additional Terms

Additionally, Python provides ways to create arguments, import common or created libraries, assign parameters, and create functions to run methods within a single script, as listed next.

Argument An argument is the value (or values) that is passed to a function. The two types of arguments are a keyword (kwargs) and positional argument (args). A keyword argument contains an identifier, whereas a positional argument does not. In this example, `firstname` is an args and `lastname` is the kwargs:

```
def my_name(firstname, lastname='reiber'):
      print ('My name is {} {}').format(firstname,lastname)
```

Another argument commonly used within Python is a system-specific function/parameter, which allows command-line arguments to be used with a Python script called sys.argv. To use sys.argv, the argv module must be imported from the Python sys module into the script. Once this module is imported, the script can assign a variable that will be used as the command-line argument. The first argv will always be the script name and can be referenced as `argv[0]`; subsequent command line arguments then follow. Here's an example:

```
from sys import argv

script, directory = argv
```

The first line imports the module argv from the module sys, and the second line holds two variables, script and directory; both are assigned the argv argument. The directory value can then be used in the script after the value is obtained at the command line when the script is executed (that is, my_script.py C:\Path\To\Directory).

Parameter A parameter value is contained within the parentheses of a function and defines which types of arguments are accepted by the function.

Modules Modules are separate, precomposed scripts that can be imported by another script to use the module's function. Using modules expands the functions of a script without having to recode many lines since they are already composed within the imported module.

import An import statement is generally accompanied by a value telling the Python interpreter what module should be loaded into memory. This import statement imports the standard Python system module and OS module, making that code available to the script that is being composed:

```
import sys
import os
```

def These three letters represent *define* and are often included within Python code to define a function. A *function* is a block of reusable code that performs a single action and is often requested multiple times within a Python script, where writing the code block over and over would be inefficient. A statement within the code can call the defined function using the function name followed by parentheses. The parentheses often contain a variable that represents an input parameter or argument. Using functions is highly recommended when repetitive tasks are required. Here's an example:

```
def create_and_check_path(tmpDir):

#Check first to see if there is this path

    if not os.path.exists(tmpDir):

#If not will create the directory

        os.makedirs(tmpDir)
```

Here's how to call the function and perform the creation of a directory if one does not exist:

```
tmpDir = "C:\Temp\Test"

 #run function to check to see if path is there
 create_and_check_path(tmpDir)
```

The function `create_and_check_path( )` is called, and the variable `tmpDir` is passed and used within the function to see if this path exists and, if not, to create the path.

Python Rules

When creating a Python script, you must follow some rules, which enable the interpreter to make sense of the script, understand what blocks of code work with others, and know how the script should progress. These rules are based upon the Python Enhancement Proposals (PEPs) within the Python Developer's Guide (www.python.org/dev/). The information in this section does not cover every rule that is suggested in the PEPs, but it represents the most frequent issues examiners might encounter when creating or running Python scripts.

Indentation Python defines blocks of code using indentation—exactly four spaces should be used per indention level. Indenting a statement line four spaces tells the interpreter that it belongs with the statement immediately preceding it. Using indentation levels enables blocks of code to work either together or independently, depending on the function the block of code is providing. Mixing tabs and spaces within a script will often create interpreter errors; spaces are the preferred indentation method. Colons are also used to declare that the following line will be the start of an indented block. Colons are not entirely needed within the code to declare an indentation is coming, but their use is widely accepted to help with the readability of the code, and if an editor uses syntax highlighting, it keys off the colon to identify syntax problems. Here's an example:

```
if foo:
    if bar:
        x = 518
        y = 911
else:
    x = 11
```

The preceding code shows the colon in the first line, which indicates the next line will be indented four spaces. The second line is indented, which indicates that it is a part of the block started in the first line. This line also has a colon, which indicates that the following line will be indented. The next line, `x = 518`, is indented to indicate it is a part of the block immediately preceding `if bar:` and tells the interpreter the statements at this level, which also includes `y = 911`, will be evaluated as part of the `if bar:` statement block. The code continues with a *dedent* (the opposite of indent) and the `else:` statement. The dedent indicates to the interpreter that this statement is at the same level of the `if foo:` statement and will be evaluated along with the `if foo:` statement. If the `else:` statement is true, the following statement will be evaluated because the interpreter recognizes the indent and associates `x = 11` with the `else:` statement.

Line Length Python documentation recommends that a single line be limited to 79 characters.

Imports Python rules state that import statements will always be placed at the top of the file and on separate lines, as shown earlier in the section "Additional Terms."

String Quotes Python allows either double or single quotation marks, which should be mixed only when a literal quote must be added to the string.

String Formatting Python allows two methods of formatting strings using placeholders instead of using the variable within the string. This method is much like creating a template and filling the placeholders with variable data assigned at runtime. This is often referred to as *string interpolation*. The variables are located at the end of the statement, and the placement of the variables into the statement string is dictated by the placement in the list. One method uses braces (`{}`), whereas the other method uses the percent sign (`%`). The `%` is currently deprecated in Python 3.x, but because it is frequently used in Python 2.7x, it is important that the examiner understand its use within a Python statement. The difference between the two methods is that if the examiner uses the `{}` string formatting technique, he or she must use the `.format()` method within the statement; if the examiner uses the `%` string formatting, the `.format()` method is not used.

The following would be used and observed in Python 2.7x and older versions:

```
#This line prints to console the files written to the directory

 print 'Done! %s was copied to %s successfully!'%(filename,dirname)
```

The following example would be used and observed in Python in version 3.x:

```
 #This line prints to console the files written to the directory

  print('Done! {} was copied to {} successfully!'.
format(filename,dirname))
```

When using this technique, if the placeholder contains an integer (such as {0} or{1}), the integer within the braces identifies the variable that will be used at that position. If there is no integer, as in the example, the placement in the string indicates the ordinal position within the list.

print A big change from 2.7x and 3.x versions of Python is the use of parentheses for print statements. To print in Python 3.x, the examiner must enclose the information that is to be printed in parentheses. When converting any 2.7 scripts to 3.x scripts, this change will be the most frequently encountered. An example of the print statement is in the preceding string formatting example.

Using Python Scripts

As indicated earlier in the chapter, the examiner can use Python within every forensic examination to complete various tasks that automated tools cannot do. Much like using SQL queries, Python scripts enable an examiner to dive deeper within mobile device file systems, uncovering data and formatting that data into readable output. Python scripts can be built to search for files; to search for strings; to carve data from binary files and file systems; to pull out laser-focused data from files such as EXIF, IP addresses, or names; and even to convert data into a readable format. As mentioned, Python scripting can enable an examiner to do just about anything with the data; the only limitation will be imposed by the examiner's imagination.

The following sections contain code examples that the examiner can use in forensic situations. These scripts can be copied, modified, or used as is. Included within the code snippets are comments that explain the code to aid in understanding or to enable the examiner to make adjustments when needed.

EXIF

Exchangeable image file format (EXIF) can be important for any investigation. Several tools, including Physical Analyzer, XRY, and Oxygen Forensic Detective, will parse EXIF data from photographs and display it and enable the reporting of such—but what if the examiner simply wants to extract the location information from all the images within a directory without having to look at each one within a directory? A customized script is the answer. Using the script located here and the directory to be searched as the argument, the examiner can run this code as a Python file at the command line or within an interpreter against a directory containing photos. The output data will contain the latitude and longitude along with the name of the file from which it was obtained.

```
# import all needed modules

import os
import sys
from sys import argv
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

#Returns a dictionary from the exif data of an PIL Image item. Also
converts the GPS Tags
def get_exif(image):

    exif_data = {}
    info = image._getexif()
    if info:
        for tag, value in info.items():
            decoded = TAGS.get(tag, tag)
            if decoded == "GPSInfo":
                gps_data = {}
                for gps_tag in value:
                    sub_decoded = GPSTAGS.get(gps_tag, gps_tag)
                    gps_data[sub_decoded] = value[gps_tag]

                exif_data[decoded] = gps_data
            else:
                exif_data[decoded] = value

    return exif_data
#function to convert the GPS coordinates stored in the EXIF to degree

def convert_to_degrees(value):
    deg_num, deg_denom = value[0]
    d = float(deg_num) / float(deg_denom)

    min_num, min_denom = value[1]
    m = float(min_num) / float(min_denom)

    sec_num, sec_denom = value[1]
    s = float(sec_num) / float(sec_denom)

    return d + (m / 60.0) + (s / 3600.0)
#Returns the latitude and longitude, if available, from the get_exif_data
function
def get_lat_lon(exif_data):

    lat = None
    lon = None

    if "GPSInfo" in exif_data:  #Check for tag
        gps_info = exif_data["GPSInfo"]

        gps_latitude = gps_info.get("GPSLatitude")   #Check for tag
        gps_latitude_ref = gps_info.get('GPSLatitudeRef')  #Check for tag
        gps_longitude = gps_info.get('GPSLongitude')  #Check for tag
        gps_longitude_ref = gps_info.get('GPSLongitudeRef')  #Check for
tag
        if gps_latitude and gps_latitude_ref and gps_longitude andgps_
longitude_ref:
            lat = convert_to_degrees(gps_latitude)
            if gps_latitude_ref != "N":
                lat *= -1

            lon = convert_to_degrees(gps_longitude)
            if gps_longitude_ref != "E":
                lon *= -1

    return lat, lon

def processDirectories(directory):
    for root, dirs, files in os.walk(directory):

        for file in files:
            try:  #This tests to see if valid image file
                fname = Image.open(os.path.join(root, file))
                exif_data = get_exif(fname)



                items = get_lat_lon(exif_data)
                e=items
                #','.join(str(i) for i in items)
                print (','.join(str(i) for i in e)+ "," + file)



            except IOError:  #This is raised if NOT image file

                print ('+++++ This is not an image file: ',file)

if __name__ == '__main__':   #Entry point for script

    script, directory = argv  #Assign the variable dictionary the argument
    processDirectories(directory)  #Send argument to the directory
function
```

Searching for Files

Examiners may want to search through a file system within a mobile device and identify and output certain files, or at least obtain their names and locations. This would be much like filtering the files by extension in an effort to find important files quickly. Why not write a Python script?

Tip

The file extension is important; if the file extension is changed to hide the real file type, this script will not locate the file. However, a simple MIME type script could easily be created with this script to identify the file by its header, which would identify the file even if the user changed the extension.

```
# import all needed modules
import os
from sys import argv
import shutil

#function to walk directory and copy files to tmp location
```

```
def processDirectories(directory):

    for root, subdirs, files in os.walk(directory):
            for file in files:

                #Here you can specify ANY file extensions
                if os.path.splitext(file)[1].lower() in ('.db', '.png',
'.html'):

     #This will copy out all files with their metadata using copy2
                     shutil.copy2(os.path.join(root, file), os.path.
join(tmpDir,file))

   #This line prints to console the files written to the directory
    print('Task Done!, All files were copied to ' +tmpDir +'
successfully!')

def create_and_check_path(tmpDir):

#Check first to see if there is this path
    if not os.path.exists(tmpDir):

#If not will create the directory
        os.makedirs(tmpDir)

if __name__ == '__main__':   #Entry point for script
    script,directory = argv

#Use this path as a place you are going to move found files to: You can
add any path here

    tmpDir = "C:\Temp\Test"

    #run function to check to see if path is there
    create_and_check_path(tmpDir)

    #Send argument to the directory function
    processDirectories(directory)
```

Hashing a Directory of Files

As you'll recall, obtaining hashes for files is important for verification, validation, and overall case success. Many mobile forensic solutions provide hashes for media only and another for created reports. The examiner can create a hashing script that will enable him or her to hash an entire file system and output that list of data to a CSV. When access to a file system is available, examiners can process and list all files using this script:

```
# import all needed modules
import os
import hashlib
from sys import argv
import datetime


def processDirectories(directory):

    global ProcessCount
    global ErrorCount

    ProcessCount = 0
    ErrorCount = 0

    for root, dirs, files in os.walk(directory):

        for file in files:
            fname = os.path.join(root, file)
            result = hashFile(fname, file)
            if result is True:
             ProcessCount += 1
        else:
            ErrorCount += 1

def hashFile(theFile, simpleName):

    # Is the path is valid
        if os.path.exists(theFile):

               #Verify that the file is real

                if os.path.isfile(theFile):


                    try:


                    # any errors will be processed in the exception clause

                        fp = open(theFile, 'rb')
                        fData = fp.read()
                        fp.close()
```

```
                         # stats will get the file information
                         # we did not include dates
                         # and times will be when copied to local disk.

                            theFileStats =  os.stat(theFile)
                            (mode, ino, dev, nlink, uid, gid, size, atime,
                            mtime, ctime) = os.stat(theFile)

                            hashType = 'MD5'
                            SHAtype = 'SHA256'

                        # Create the hash objects
                            hash = hashlib.md5()
                            hash256 = hashlib.sha256()

                        #  hash the contents obtain from the file
                            hash.update(fData)
                            hash256.update(fData)

                        # extract the hexidecimal version of the hash
                            hexMD5 = hash.hexdigest()
                            hexSHA = hash256.hexdigest()

                        # convert the hex string to upper case
                            hexMD5 = hexMD5.upper()
                            hexSHA = hexSHA.upper()

                        # Print the results
                            print ('Filename: {} , Hashtype: {} , {} ,
                                    Hashtype: {}, {},size: {}.format(
                                    os.path.basename(theFile), hashType,
                                    hexMD5, SHAtype, hexSHA, size))


                        except IOError:
                        # An exception occurred when processing the file
                            print (theFile + ' File Processing Error')


if __name__ == '__main__':

    script, directory = argv
    processDirectories(directory)
```

Using Regular Expressions

Searching across a data set to locate strings and other values can often provide the “smoking gun”; however, most mobile forensic tools do not enable searching across data sets, and if a tool does support searching, it can be limited by what can be searched and how a search can be performed. By creating a Python script, the examiner can create regular expressions for things like credit card numbers, Social Security numbers, name formats, IP addresses, and more. These regular expressions can be controlled within the script to turn on and off what regular expressions will be used. An examiner can also add to the list in the following example when needed. Python is very versatile.

```
# import all needed modules
import os
import glob
from sys import argv
import shutil
import re

#function to walk directory and copy files to tmp location

def processDirectories(directory):

    for root, subdirs, files in os.walk(directory):
            for file in files:

                f = os.path.join(root, file)
                f = open(f, 'rb' )
                file_contents = str(f.read())

                #You can place any string here to look for

                #SocialSecurity ("^\d{3}-\d{2}-\d{4}$")
                #Website
                #("(http://|)(www\.)?([^\.]+)\.(\w{2}|(com|net|¬
                            org|edu|int|mil|gov|arpa|biz|aero|name|coop|in
fo|pro|¬
                            museum))$")
                #Latitude/Longitude
                #("^[-+]?([1-8]?\d(\.\d+)?|90(\.0+)?)
,\s*[-+]?(180(\.0+)?|((1[0-
                             7]\d)|([1-9]?\d))(\.\d+)?)$")
#IP Addresses
                #("\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b")

                # ********** Paste or Enter your regular expression below
                rxo = re.compile("(?:\d{1,3}\.){3}\d{1,3}")

                #If the string is located it will print out to the console
the
                # offset, hit and the filename

                pos = 0
                while 1:
                    evid_hits = rxo.search(file_contents, pos)
                    if not evid_hits: break
                    #for found in evid_hits:
                    print ('Offset: %4s: 4%s filename: 2%s' %
                             (evid_hits.start(), evid_hits.group(),file))
                    pos = evid_hits.end()
                f.close()

if __name__ == '__main__':   #Entry point for script
    script,directory = argv

    #Send argument to the directory function
    processDirectories(directory)
```

Chapter Summary

Building SQLite queries to supplement an automated tool's poor support, or even lack of support, for extracting information from an app is a necessity for today's mobile device investigations. Today's tools such as Oxygen Forensic Detective SQL Viewer and Cellebrite's SQL tool can build visual queries without your knowing a line of SQL! However, queries can be built using many free tools as well, such as the SQLQuery Builder and SQLite Toolkit, with little knowledge of SQLite structures and statement building as well. The examiner must understand the `SELECT` command and various others, such as `JOIN`, `ON`, and `WHERE`, even when using an automated query builder. This knowledge will help the examiner explain what the commercial mobile forensic tools are doing when extracting information. Understanding what is "under the hood" will help an examiner build superior queries that extract much more information, surpassing even the most expensive mobile forensic solutions. The information in this chapter showed that data is available and within the examiner's grasp—just build a query and extract it!

SQLite queries cannot be built for files that are not SQLite files, but there are thousands of these types of files within a mobile device. Armed with the power of Python, no file is safe from an examiner's custom Python scripts. Building Python scripts from scratch may seem intimidating, but the examiner can consult many online resources to help with writing scripts. In this chapter, you were exposed to the surface-level scripting needs and rules, along with terminology frequently used to build simple, quick, and effective scripts for mobile device file interrogation. By using this information, an examiner gains the basic skills to complete several tasks using Python. An examiner should be able to take an already built Python script and modify it as needed to complete a specific task. In addition, an examiner should be able to create small scripts using the various statements, variables, expressions, and operands in the Python language. The take-away is the realization that these tools and methods enable an examiner to parse information from any file, any file system, and any mobile device image using custom-built Python scripts. The sky is the limit, and the options are unlimited for an examiner who has the desire and aptitude to learn.

15 Advanced Android Analysis

Android devices are the most widely used smart devices globally. In fact, on May 17, 2017, the CEO of Google, Sudar Pichai, announced that more than 2 billion active Android devices were in use globally. Some research firms have placed the market share as of August 2017 at almost 90 percent (87.7 percent stated by DMR, a statistical company), though Google VP Dave Burke reported it was around 66 percent in all actuality, after statistical companies put market share considerably higher. Consumers can purchase any of the approximately 18,000 to 25,000 Android devices available, including smart phones and tablets. Luckily for the mobile examiner, these devices all have similar file system structures.

This chapter is dedicated to the deconstruction and analysis of the many files stored within the Android file system. As with a logical analysis of Android devices, in order for an examiner to perform an advanced analysis, a file system must be available. Such an analysis can be accomplished using a backup of the device, a noninvasive rooting technique, or acquisition via JTAG (Joint Test Action Group), ISP (In Service Programming), or a chip-off procedure. An examiner can also obtain some app and file system files using the Media Transfer Protocol (MTP) method discussed in Chapter 10, which will obtain the internal storage area along with external media.

As with iOS devices, there is, of course, the "tip of the iceberg data," but that information pales in comparison to the app data, file system cache files, and user storage files that can be found intertwined within the Android file system. Using SQLite queries and Python scripting, as discussed in Chapter 14, the examiner can analyze artifacts that an automated tool is incapable of parsing. Deep analysis of the Android file system should be attempted with each examination. This chapter lists common storage areas, such as Short Message Service (SMS), contacts, Multimedia Messaging Service (MMS), and call logs, along with files that are found within most Android file systems. Tables and lists of additional file locations are also revealed throughout the chapter.

As discussed in Chapter 13 with regard to iOS advanced analysis, freefile lists and the write-ahead log (WAL) are also present within an Android device, and this data will be extremely helpful during many examinations. Android malware can also pose a significant problem for the Android operating system, and it can infiltrate many of the apps from Google or any other Android app repository. An examiner must investigate the possibility of malware within the apps on an Android device, which can add complexity to any Android investigation.

The examination of an Android device should never be limited simply to obtaining a logical collection using a mobile forensic solution's application package (APK) file. Using the correct processes, procedures, and methodologies can yield a monumental amount of investigatory data in any Android mobile forensic examination.

Android Device Information

Android device manufacturers include LG, Samsung, Motorola, ZTE, and others, but it is the underlying file system that is of importance to an examiner, not the manufacturer. The file system is formulated by the

Android OS and the version used by the actual device. One Android device might look like an apple, while another looks like a banana—but the examiner should understand that the data under the hood, including the file system look and feel, is almost identical in all of them—depending, of course, on the OS version. Granted, an Android device's firmware can make a difference—so Samsung might include some files that Motorola does not, and LG might include some files not found in a ZTE device. The point is, one Android device might look different from another on the outside, but the underlying file system on both is remarkably similar. To be quite honest, the storage areas for actionable examiner data are relatively consistent over OS versions, with security and low-level development updates being the main changes.

Examiners should refer to the device examination by operating system type instead of by simply stating they are examining a Samsung Galaxy device. With more than 60 different Samsung Galaxy devices on the market and available, referring to the operating system version, security patch, and model first will help solidify the file system variations. It is possible the same Samsung Galaxy devices can be using two different versions of the OS, depending on the carrier. Documenting the OS version can also assist the examiner when analyzing another similar device, because often file system location and other data are the same.

Note

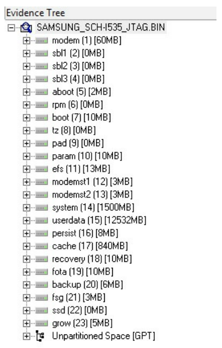
To obtain many of the details (such as security patch and model number) described in this chapter, the examiner or device owner will usually need to unlock the device to examine the settings. Most tools report most of the data mentioned here, with the exception of the security patch version.

The particular methods used to obtain the information should not matter so long as a file system is available. The location of data within the file system will be the same, whether an image is obtained from a binary dump, backup, JTAG/ISP, or chip-off. The information in this chapter should be used as a map to point the examiner toward the various pots of gold within the Android file system.

Partitions

When an examiner is completing a non-invasive or an invasive physical collection of an Android device, he or she can find many partitions within the file system. Some practitioners state that there are six partitions, but from personal experience, I know that there is too much variation among devices to estimate the number of partitions an examiner may encounter—I have encountered four, eight, and even twelve partitions. The issue with today's devices, and encryption, will be the primary partition (userdata), and the mount point /data folder will be fully encrypted.

Partitioning in Windows creates independent areas on disk, assigns a drive letter, decides where an operating system can be loaded if active, and determines where files can be stored or designated as a program directory. Android devices, however, do not use drive letters, but instead use *mount points*, directories under the root directory (/) that contain the data from each partition. All partitions are within the internal storage of the Android device but are logically separated (see Figure 15-1). The Android system sees partitions as separate, independent media, even though they are mounted in a single logical location under root. This root location is in fact a virtual pointer to the actual location of the data in flash media, and this is evident when you look at a JTAG binary dump or chip-off, where many folders are located under root and contain various file system files and folders.



```
Evidence Tree
⊟ SAMSUNG_SCH-I535_JTAG.BIN
  ⊞ modem (1) [60MB]
  ⊞ sbl1 (2) [0MB]
  ⊞ sbl2 (3) [0MB]
  ⊞ sbl3 (4) [0MB]
  ⊞ aboot (5) [2MB]
  ⊞ rpm (6) [0MB]
  ⊞ boot (7) [10MB]
  ⊞ tz (8) [0MB]
  ⊞ pad (9) [0MB]
  ⊞ param (10) [10MB]
  ⊞ efs (11) [13MB]
  ⊞ modemst1 (12) [3MB]
  ⊞ modemst2 (13) [3MB]
  ⊞ system (14) [1500MB]
  ⊞ userdata (15) [12532MB]
  ⊞ persist (16) [8MB]
  ⊞ cache (17) [840MB]
  ⊞ recovery (18) [10MB]
  ⊞ fota (19) [10MB]
  ⊞ backup (20) [6MB]
  ⊞ fsg (21) [3MB]
  ⊞ ssd (22) [0MB]
  ⊞ grow (23) [5MB]
  ⊞ Unpartitioned Space [GPT]
```

FIGURE 15-1 **A JTAG binary file, when imported into FTK Imager, will show as a logical file system under the root folder.**

Depending upon the Android's operating system version, an examiner can encounter various options. For example, prior to Honeycomb (version 3.0), the sdcard0 partition would be accessible within the partition table and was often included in a collection by use of a *symlink* (symbolic link) to /mnt/sdcard, the mount point for the media card. However, on device versions later than Gingerbread (version 2.3), the examiner may see another "external media card" that may actually be the internal media card. Because some devices can contain both internal and external media, an examiner should always examine both the sdcard and ext_sd (sdcard2, sd, external_sd) partitions. In addition, some other partitions reside only within certain device families. Table 15-1 shows the Android OS version numbers, release dates, and common names.

TABLE 15-1 **Android Operating Systems**

OS Version	Release Date	Common Name
1.0	November 2007	Beta
1.5	April 2009	Cupcake
1.6	September 2009	Donut
2.0	October 2009	Eclair
2.2	May 2010	Froyo
2.3.x	December 2010	Gingerbread
3.0	February 2011	Honeycomb (tablet only)
4.0–4.2.x	October 2011	Ice Cream Sandwich
4.3	July 2013	Jellybean
4.4	October 2013	KitKat
5.0	November 2014	Lollipop (Android L)
6.0	October 2015	Marshmallow (M)
7.0	August 2016	Nougat (N)
8.0	August 2017	Oreo (O)
8.1	December 2017	Oreo (O), but adds Android GoEdition (slim for devices with less than 1GB of RAM)
9.0	March 2018 (beta)	Android (P)

For example, Samsung Galaxy S Android devices have a data partition within their file systems called dbdata. With the Samsung Galaxy series, the devices not only kept the legacy dbdata partition but also conformed to the other Android operating systems and included a userdata partition. Samsung devices are the only devices in which the user data is split between the dbdata and userdata partitions, which is observed primarily in Samsung S Series Galaxy devices running Froyo and Gingerbread (2.2 and 2.3) versions of Android. An examiner should know whether both of the partitions are available or the singular dbdata partition is available. When data is split between these two partitions, the personal information manager (PIM) data and "tip of the iceberg data," along with their respective databases, are generally stored in the dbdata partition, and app data, phone settings, and configurations are stored on the userdata partition. If the examiner's solution does not allow for simultaneous importing of both partitions, these locations will have to be individually analyzed to recover valuable user data. The predominant version of Android remains 7.0 and 7.1 for devices, with the exception being the Pixel and Nexus series devices that are currently utilizing Android 8. However, current Android devices (as of this writing, Android 9x) consistently store third-party app data and PIM data within the /data folder (AKA userdata). The legacy dbdata partition has not been observed since the Series S Samsung devices.

When examining an Android device, the examiner usually will be looking through the userdata partition, found at /data/data for physical and /data for critical artifacts, but the cache partition often contains valuable data as well. The various file system types, along with paths to various files, will be covered in the following section. Table 15-2 shows various memory partitions and descriptions that might be encountered. Note that this list is not all inclusive but represents partitions commonly encountered.

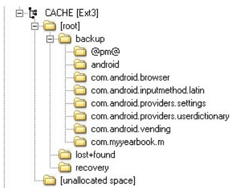
TABLE 15-2 **Partitions Encountered Within an Android Mobile Device**

Partition Name	Description
boot	RAMDisk, kernel
cache	App and OS cache area; may include artifacts
data	User data, settings, applications, and third-party apps for some Samsung devices; can be represented as userdata as well
dbdata	User data, settings, and some stock applications and settings (Samsung S)
emmc	Internal media card
misc	System feature settings; used by the device for configuration and hardware settings
modem	Firmware for modem; hardware dependent
radio	Firmware for radio, cellular, GPS, data connection, and Bluetooth; hardware dependent
recovery	Device stock recovery image often used as location to install alternative boot partitions (Custom Recovery) by mobile forensic tools; no user data here unless hidden by user
sdcard	Internal or external media card; for some phones, other partitions could be sd, emmc, and so on, depending on device type and whether internal or external to the device
system	Operating system and settings, built-in application settings
userdata	User data, settings, applications, and third-party apps for some Samsung devices; can also be represented as data
wimax	Firmware for WiMAX; hardware dependent

The File System

The various Android file system types were referenced in Chapter 8 and range from the first Android devices using YAFFS (Yet Another Flash File System) or YAFFS2, to Robust File System (RFS), to the EXT (Extended File System) variants, and F2FS (Flash-Friendly File System) developed by Samsung. Predominantly, EXT 4 is used as the underlying file system format for today's Android devices. If the device utilizes an internal or external media card, various File Allocation Table (FAT) file system formats could be used. The newest SD card format was created by Samsung but released in Android O, called SDCardFS, which is replacing FUSE (Filesystem in Userspace). This is something an examiner will not notice, but it fixed a glaring forensic implication that few could explain: the change of a date on media if transferred to a computer from the device, typically via MTP. The date and time was often altered, which could have serious forensic implications. These various file systems are currently interpreted by most mobile forensic tools, with the differences in file systems generally being the size of the storage blocks on the flash, journaling, read/write speed, and minimum/maximum file sizes. The newer the device OS version, the larger the block size typically. Today's mobile forensic tools can distinguish and mount these file systems without a problem, displaying a file system to the examiner that can be navigated and parsed. It was typical in the earlier years of Android analysis for the examiner to use Linux tools and command line tools within Windows to format these file systems and view them properly. That, fortunately, is no longer the case. What this means to the examiner is that the examination and analysis can occur immediately, requiring little work to maintain or rebuild the Android file system's integrity—so the examiner can concentrate on the analysis.

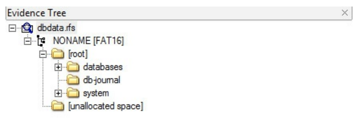
The userdata partition contains not only the user data storage areas but also app information, app configuration files, uncompiled app code, and virtual sandboxed–compiled code. With the exceptions mentioned with the dbdata partition, on some Samsung S Android devices, the examiner will find a gold mine of information within this area. Not only are user files and app data available, but the actual app code is within the examiner's reach. With the app code, an examiner can also look for malware instances—something a standard logical collection and analysis cannot offer. The cache partition (Figure 15-2) can contain /backup, /lost+found, and /recovery folders. The /backup folder often contains Android app packages with stored temporary data and can yield good artifacts. The /lost+found folder will contain files that had been recovered as a result of file-system corruption and should also be examined. Finally, the /recovery folder holds log files listing the recovery operations of the mobile device.



```
CACHE [Ext3]
  [root]
    backup
      @pm@
      android
      com.android.browser
      com.android.inputmethod.latin
      com.android.providers.settings
      com.android.providers.userdictionary
      com.android.vending
      com.myyearbook.m
    lost+found
    recovery
  [unallocated space]
```

FIGURE 15-2 **The cache partition should be examined for possible evidence when this partition exists within the Android device.**

Android dbdata and userdata partitions extracted from a Samsung SCH-I500 running version 2.2.3 are shown in Figures 15-3 and 15-4. There are obvious differences between the two partitions in the number of folders and files stored. Within the userdata partition is the majority of user data within an Android file system, including valuable app data and system configuration files; the dbdata partition contains valuable user data in the lone /databases directory. The separation of user data between partitions is not typical, and generally the examiner will concentrate on artifact hunting within the userdata partition and media storage (such as sdcard, emmc, sd) partitions within the Android device output. Outlined in Table 15-3 are good locations for an examiner to investigate when conducting an advanced analysis of an Android's userdata file system. The file system layouts for today's devices up to Android 8.0 are similar, as discussed later in the chapter.



```
Evidence Tree
dbdata.rfs
  NONAME [FAT16]
    [root]
      databases
      db-journal
      system
    [unallocated space]
```

FIGURE 15-3 **The dbdata partition in some Samsung Android devices can contain valuable data within the**

/databases folder.

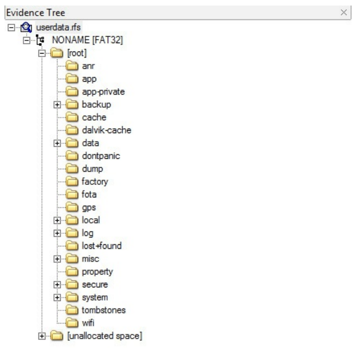


FIGURE 15-4 The userdata (data) partition is much more robust and is found with the dbdata in predominantly Samsung Galaxy S Androids, but in /data through the current OS (9x).

TABLE 15-3 Android Partitions Compiled to Show the Major Landmarks Within the userdata, dbdata, sdcard, and emmc Partitions

Folder (data partition)	Data Description
anr	(App Not Responding) Debug and threading information, including dates and times for app execution. This is not located in devices from Android 6 to current (Android 9x).
app	Android application package files (APK). Consistent in current Android (9x), but only available in physical collection.
app-private	Apps marked as private by the developer for copy protection. Beginning with Android 4.4, an app can have an app-private folder within its own file system, deprecating this folder from use for an app storage or APK file. This is not observable in 6x to current Android devices (9x).
backup	File metadata, settings, and other user data marked to be backed up using enterprise, Google, or other backup services. Consistent in current Android (9x), but available only in physical collection.
clipboard	Any data copied and pasted into an app; Samsung devices generally. Still available in Samsung 8x devices with root access and decryption of the userdata partition.
dalvik-cache	DEX (Dalvik Executable) files for APK files installed on the system. Third-party apps are prefixed with *data@*, system files are prefixed with *system@*. The @ prefix is not observable in current devices, but simply the .dex file. This is consistent in current Android devices (9x), but available only in physical collection.
data	All user data, stored app data. Different from the app folder because this holds the actual "user" data.
local/tmp	Temporary files that are removed upon restart. An examiner can often see files used to exploit and obtain root access (such as BusyBox, superuser, forensic agent for logical files).
log	System-critical errors and built-in app errors (such as CallDropInfoLog.txt, CallNosvcInfoLog.txt). This is not observable in 6x to current devices of Android.

media	Internal storage location for apps, media files (images, videos, documents), phone media files (DCIM folder), and phone settings.
Misc	Various folders pertaining to operation of things like Wi-Fi, Android Debug Bridge (ADB), and Bluetooth.
property	Persistent mobile device settings (language, country, time zone).
system	Password key, account information, SIM information, and other system-level files
tombstones	Used for diagnostic work for app crashes. Contains log files that can be used for debugging purposes. This is not observable in 6x to current devices of Android.
[unallocated space]	Not a folder, but space unused by the Android system. Also contains valuable deleted files no longer used by the system and user apps. Available only on a non-invasive physical or an invasive physical extraction.
Folder (sdcard/emmc partition) and data/0/ media	
Android/data	App file storage area for third-party apps storing data to the removable media.
Android/media	App media storage.
DCIM and 0/DCIM	Thumbnail and camera data.
download	Device storage for downloads from apps, if specified, on the device.
Pictures	Apps storage for editors, apps.
ScreenCapture	Screen shots produced on the Android.
Various app folders	Apps can store data within the root of the media and produce folders to hold media files.
Folder (dbdata partition)	Only Samsung Galaxy S.
databases	Storage for user system-level apps.
databases/system	Storage for all installed apps with permissions along with installation paths.

Unallocated space can be observed within each Android partition and is accessible with physical collection.

This area typically is unstructured because software is unable to build a file view. Generally, an examiner must use data carving tools to parse through the many blocks of data within this area. Many mobile forensic tools contain some type of data carving tool, but any data carving conducted by a forensic tool should allow for custom file type searching along with typical file types (such as jpeg, .html, .pdf, and .doc). FTK, EnCase, Scalpel, Oxygen Forensic Detective, and Cellebrite Ultimate are recommended. The carving of unallocated space, when available, should occur, because many images, videos, database files, text strings, and other evidentiary files can be recovered from this space. (This luxury of unallocated space is not available for iOS, as discussed in Chapter 13.) Of course, if the file system of an Android device has been encrypted, recovery is much more difficult and sometimes impossible. In devices running Android 6x to Android 9x, encryption is the default.

Devices up to Android 7 using full disk encryption (FDE) are based on dm-crypt, which works at the block device layer and can encrypt the device and removable media. This is not a file-level encryption but a full disk encryption, which differs from iOS encryption. FDE on an Android applies only to the user partition (/data), not the entire system. This means that the system partition containing the OS is unencrypted, and often the way in which decryption techniques exploit the OS is via the system partition. The encryption is only as good as the password in place, since the algorithm is based on the complexity of the password or phrase. At Android 5.0, new techniques for encryption were added that allowed only the used portions of the data partition to be encrypted, helping with speed. Furthermore, added to Android 5.0 was the encrypt on first boot feature, and an off-the-shelf device running Android 5.0 would encrypt by default on the first bootup; these devices cannot be returned to an unencrypted state. This is different for devices that had been upgraded to Android 5.0; these devices could, in fact, be returned to an unencrypted state if a factory reset was performed. Some investigative methods exploit devices with unlocked bootloaders, replace the system OS partition, and attempt to capture a user entering a password with a keylogger. Others, such as Passware, tested versions up to Android 4.0, to brute-force the password or phrase and then use the information to decrypt the user data.

Note

The caveat to all of these tools is an unlocked bootloader, because using these methods on a locked bootloader can lead to all data on the user partition being wiped.

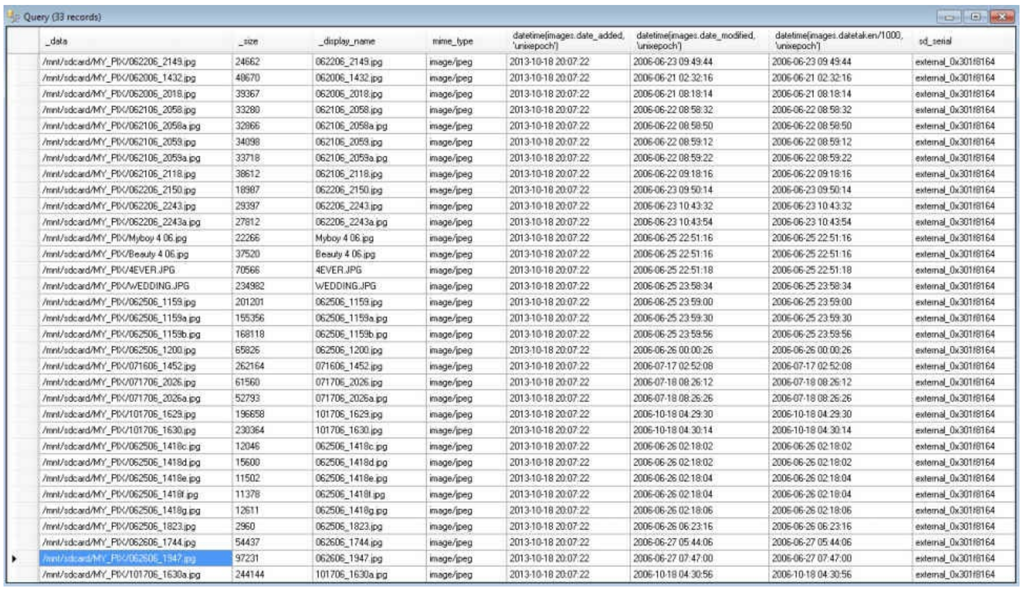
Also, open source tools such as hashkill (now deprecated) can break the passcode or phrase in versions up to Android 4.4, even when another encryption password is used along with the lock PIN or password. Android Lollipop (5) changed the encryption to rely not only on the PIN or passcode but also a secondary salt; thus, the currently available methods will not work. Android 5 devices came out of the box with encryption turned on, but without a password, and using the randomly generated keys shipped with the device, the user data via regular extraction such as ADB backup and logical agent is available when the device is turned on. If the user does not enter a password, the device data will be still be accessible with forensic tools, but physical extraction of the data can be hampered because of the encryption. At Android 7, file-level encryption was implemented and made available, which obviously made decryption more difficult, because at the file level a different decryption key could be implemented, which is much different from block-level or entire device encryption. This is true for both Android 8 and 9. The decryption of these devices upon obtaining a full physical extraction is where many commercial tools are devoting their research and development.

A basic agent-type logical examination of the Android device does not allow access to the device file system because the methods employed by software solutions simply use an installed APK file to run code to query known databases. If the database has a permission level appropriate for communication with another app, the data can be extracted. As mentioned, access to the internal and external media partitions is accessible with MTP tools and will allow for the collection of this data and give the examiner a file system–like experience—often containing many app media storage areas—but access to the /data area of the device at critical databases is not available unless root access is granted. If an examiner must conduct an advanced analysis of an Android device, the tool or method must either gain root (such as custom ROM, exploit, and so on) access via JTAG/ISP or chip-off. Some access to app data can be obtained using the built-in backup function (ADB Backup) within Android versions 4.x and later, but again this is only a portion of the data from the Android user data partition, and as the operating systems have progressed, the amount of app data has dwindled.

Predominant Android File Types

Android devices contain a multitude of file types, ranging from APK, DEX, LOG, SO, ASEC, DB, and JSON, to XML and TXT files. Also included are WAL files and journal files for SQLite databases—significantly more journal files than are found within the iOS file system when looking to older Android OS devices. The types of files an examiner will use in artifact investigations will be SQLite and XML files. APK and DEX files can be used for malware investigations, as discussed later in the chapter in the section "Android App Files and Malware."

SQLite databases contain significant information, and an investigator would be remiss if he or she did not conduct a comprehensive analysis of these files. Android SQLite databases are no different from those of iOS and can be examined in much the same way. SQLite databases contain tables and fields with user preferences, data, and any persistent data the user and app needs stored for later use. Hundreds of thousands of apps are available for installation on any Android device, and mobile forensic solutions cover only 0.014 percent of the available applications currently available on the Google Play store. Of those supported apps, automatic parsing of data is often flawed. Because apps can change upon each update, a mobile solution is often outdated at release, so these solutions are always at a technological disadvantage. An examiner who can parse apps manually will always have the upper hand on these advances. Figure 15-5 shows an example of parsed data from a SQLite database. In this example, many tools parse the media store but fail to recognize the associated thumbnails.


Query (33 records)

_data	_size	_display_name	mime_type	datetime(images.date_added, 'unixepoch')	datetime(images.date_modified, 'unixepoch')	datetime(images.datetaken/1000, 'unixepoch')	sd_serial
/mnt/sdcard/MY_PIX/062206_2149.jpg	24662	062206_2149.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-23 09:49:44	2006-06-23 09:49:44	external_0x301f8164
/mnt/sdcard/MY_PIX/062006_1432.jpg	40670	062006_1432.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-21 02:32:16	2006-06-21 02:32:16	external_0x301f8164
/mnt/sdcard/MY_PIX/062006_2018.jpg	39367	062006_2018.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-21 08:18:14	2006-06-21 08:18:14	external_0x301f8164
/mnt/sdcard/MY_PIX/062106_2058.jpg	33280	062106_2058.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-22 08:58:32	2006-06-22 08:58:32	external_0x301f8164
/mnt/sdcard/MY_PIX/062106_2058a.jpg	32866	062106_2058a.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-22 08:58:50	2006-06-22 08:58:50	external_0x301f8164
/mnt/sdcard/MY_PIX/062106_2059.jpg	34098	062106_2059.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-22 08:59:12	2006-06-22 08:59:12	external_0x301f8164
/mnt/sdcard/MY_PIX/062106_2059a.jpg	33718	062106_2059a.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-22 08:59:22	2006-06-22 08:59:22	external_0x301f8164
/mnt/sdcard/MY_PIX/062106_2118.jpg	38612	062106_2118.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-22 09:18:16	2006-06-22 09:18:16	external_0x301f8164
/mnt/sdcard/MY_PIX/062206_2150.jpg	18987	062206_2150.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-23 09:50:14	2006-06-23 09:50:14	external_0x301f8164
/mnt/sdcard/MY_PIX/062206_2243.jpg	29397	062206_2243.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-23 10:43:32	2006-06-23 10:43:32	external_0x301f8164
/mnt/sdcard/MY_PIX/062206_2243a.jpg	27812	062206_2243a.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-23 10:43:54	2006-06-23 10:43:54	external_0x301f8164
/mnt/sdcard/MY_PIX/Myboy 4 06.jpg	22266	Myboy 4 06.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-25 22:51:16	2006-06-25 22:51:16	external_0x301f8164
/mnt/sdcard/MY_PIX/Beauty 4 06.jpg	37520	Beauty 4 06.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-25 22:51:16	2006-06-25 22:51:16	external_0x301f8164
/mnt/sdcard/MY_PIX/4EVER.JPG	70566	4EVER.JPG	image/jpeg	2013-10-18 20:07:22	2006-06-25 22:51:18	2006-06-25 22:51:18	external_0x301f8164
/mnt/sdcard/MY_PIX/WEDDING.JPG	234982	WEDDING.JPG	image/jpeg	2013-10-18 20:07:22	2006-06-25 23:58:34	2006-06-25 23:58:34	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1159.jpg	201201	062506_1159.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-25 23:59:00	2006-06-25 23:59:00	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1159a.jpg	155356	062506_1159a.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-25 23:59:30	2006-06-25 23:59:30	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1159b.jpg	168118	062506_1159b.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-25 23:59:56	2006-06-25 23:59:56	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1200.jpg	65826	062506_1200.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 00:00:26	2006-06-26 00:00:26	external_0x301f8164
/mnt/sdcard/MY_PIX/071606_1452.jpg	262164	071606_1452.jpg	image/jpeg	2013-10-18 20:07:22	2006-07-17 02:52:08	2006-07-17 02:52:08	external_0x301f8164
/mnt/sdcard/MY_PIX/071706_2026.jpg	61560	071706_2026.jpg	image/jpeg	2013-10-18 20:07:22	2006-07-18 08:26:12	2006-07-18 08:26:12	external_0x301f8164
/mnt/sdcard/MY_PIX/071706_2026a.jpg	52793	071706_2026a.jpg	image/jpeg	2013-10-18 20:07:22	2006-07-18 08:26:26	2006-07-18 08:26:26	external_0x301f8164
/mnt/sdcard/MY_PIX/101706_1629.jpg	196658	101706_1629.jpg	image/jpeg	2013-10-18 20:07:22	2006-10-18 04:29:30	2006-10-18 04:29:30	external_0x301f8164
/mnt/sdcard/MY_PIX/101706_1630.jpg	230364	101706_1630.jpg	image/jpeg	2013-10-18 20:07:22	2006-10-18 04:30:14	2006-10-18 04:30:14	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1418c.jpg	12046	062506_1418c.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 02:18:02	2006-06-26 02:18:02	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1418d.jpg	15600	062506_1418d.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 02:18:02	2006-06-26 02:18:02	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1418e.jpg	11502	062506_1418e.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 02:18:04	2006-06-26 02:18:04	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1418f.jpg	11378	062506_1418f.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 02:18:04	2006-06-26 02:18:04	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1418g.jpg	12611	062506_1418g.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 02:18:06	2006-06-26 02:18:06	external_0x301f8164
/mnt/sdcard/MY_PIX/062506_1823.jpg	2960	062506_1823.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-26 06:23:16	2006-06-26 06:23:16	external_0x301f8164
/mnt/sdcard/MY_PIX/062606_1744.jpg	54437	062606_1744.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-27 05:44:06	2006-06-27 05:44:06	external_0x301f8164
/mnt/sdcard/MY_PIX/062606_1947.jpg	97231	062606_1947.jpg	image/jpeg	2013-10-18 20:07:22	2006-06-27 07:47:00	2006-06-27 07:47:00	external_0x301f8164
/mnt/sdcard/MY_PIX/101706_1630a.jpg	244144	101706_1630a.jpg	image/jpeg	2013-10-18 20:07:22	2006-10-18 04:30:56	2006-10-18 04:30:56	external_0x301f8164

FIGURE 15-5 **Collecting SQLite information manually parsed from apps that are not supported by any automated tool is a necessity with Android devices, such as the MediaStore database from com.android.providers.media.**

XML (Extensible Markup Language) is used extensively within the Android file system for settings, passwords, application protocols, network data sharing, and other preferences. The most significant XML files are located in the shared_pref folder within the app folders (see Figure 15-6). XML is a file type that uses `<?xml version="1.0" encoding="utf-8"?>` as a header, with a set of rules for encoding documents in machine-readable form. As discussed in Chapter 14, XML can be cleanly parsed using Python scripts to decode and display the key and values listed within the file.

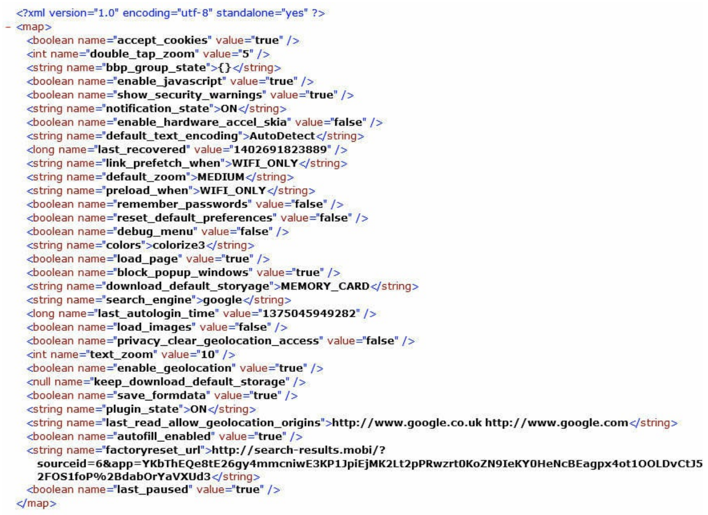

```
  <?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <map>
    <boolean name="accept_cookies" value="true" />
    <int name="double_tap_zoom" value="5" />
    <string name="bbp_group_state">{}</string>
    <boolean name="enable_javascript" value="true" />
    <boolean name="show_security_warnings" value="true" />
    <string name="notification_state">ON</string>
    <boolean name="enable_hardware_accel_skia" value="false" />
    <string name="default_text_encoding">AutoDetect</string>
    <long name="last_recovered" value="1402691823889" />
    <string name="link_prefetch_when">WIFI_ONLY</string>
    <string name="default_zoom">MEDIUM</string>
    <string name="preload_when">WIFI_ONLY</string>
    <boolean name="remember_passwords" value="false" />
    <boolean name="reset_default_preferences" value="false" />
    <boolean name="debug_menu" value="false" />
    <string name="colors">colorize3</string>
    <boolean name="load_page" value="true" />
    <boolean name="block_popup_windows" value="true" />
    <string name="download_default_storyage">MEMORY_CARD</string>
    <string name="search_engine">google</string>
    <long name="last_autologin_time" value="1375045949282" />
    <boolean name="load_images" value="false" />
    <boolean name="privacy_clear_geolocation_access" value="false" />
    <int name="text_zoom" value="10" />
    <boolean name="enable_geolocation" value="true" />
    <null name="keep_download_default_storage" />
    <boolean name="save_formdata" value="true" />
    <string name="plugin_state">ON</string>
    <string name="last_read_allow_geolocation_origins">http://www.google.co.uk http://www.google.com</string>
    <boolean name="autofill_enabled" value="true" />
    <string name="factoryreset_url">http://search-results.mobi/?
      sourceid=6&app=YKbThEQe8tE26gy4mmcniwE3KP1JpiEjMK2Lt2pPRwzrt0KoZN9IeKY0HeNcBEagpx4ot1OOLDvCtJ5
      2FOS1foP%2BdabOrYaVXUd3</string>
    <boolean name="last_paused" value="true" />
  </map>
```

FIGURE 15-6 **XML files found within an app's shared_pref folder can be used by an examiner to obtain critical investigation data.**

Other file types within the Android file system include text files, log files, APKs, and DEX files, as discussed in the following sections.

Artifacts

Much like iOS, an Android device is full of digital gold if the file system is obtainable. Hundreds of SQLite databases, XML files, text files, media files, and more are stored within the user data area of the device. A logical extraction of an Android device still yields some great data, but the information has no "meat"—in other words, the information displayed is simply a query of the appropriate database, and the data is rendered to the examiner within the user interface. What about the files this data came from? These files contain valuable fields that might not have been queried by the mobile forensic solution's APK file but can be accessed if the file is obtained. Also, deleted data within the database free pages can be recovered, as discussed in Chapter 13 with iOS, along with the WAL files that are used. This "tip of the iceberg data" and its locations are listed in Table 15-4 to assist the examiner in recovering the data that cannot be recovered with a logical collection.

TABLE 15-4 **Standard "Tip of the Iceberg" Android Data, with Database Locations for Additional Analysis when Needed and Validation on Automated Parsing by Forensic Tools**

Artifact	Path and Description
Bluetooth devices	/data/com.android.bluetooth/btopp.db
Browser bookmarks	/data/com.android.browser/ browser.db Or if dbdata exists, /dbdata/databases/com.android.browser/browser.db/com.sec.android.app.sbrowser/db/SBrowser.db (Samsung Stock) Bookmarks table New Androids also install Chrome as a default browser and root is needed /com.android.chrome/app_chrome/Default/
Browser history	/data/com.android.browser/ browser.db Or if dbdata exists, /dbdata/databases/com.android.browser/browser.db Bookmarks table in browser.db and bookmark = 0 New Androids also install Chrome as a default browser and root is needed /com.android.chrome/app_chrome/Default/History urls table with visits Table for date/times
Browser searches	/data/com.android.browser/browser.db /com.android.browser.provider/browser.db Searches Table Or if dbdata exists, /dbdata/databases/com.android.browser/browser.db Searches table in browser.db New Androids also install Chrome as a default browser and root is needed /com.android.chrome/app_chrome/Default/History

Call history	/data/com.android.providers.contacts/contacts2.db /data/com.android.providers.contacts/databases/contacts2.db (need root in new devices) **Motorola:** /data/com.motorola.blur.providers.contacts/contacts2.db Calls table in contacts2.db **Samsung:** /data/com.sec.android.provider.logsprovider/logs.db In Android 5x to current. /data/com.sec.android.provider.logsprovider/db/logs.db Or if dbdata exists, /dbdata/databases/com.sec.android.provider.logsprovider/logs.db Calls in log table and logtype = 100
Contacts	/data/com.android.providers.contacts/contacts2.db /data/com.android.providers.contacts/databases/contacts2.db (need root in new devices) Or if dbdata exists, /dbdata/databases/com.android.providers.contacts/contacts2.db
Media (images, video, audio)	/data/com.android.providers.media/external.db or if dbdata exists /dbdata/databases/com.android.providers.media/external.db This is the media store that is used to determine the location of media files cached and used by forensic tools to locate media on the device. This path is no longer available in a regular ADB backup. Root is necessary.
MMS	/data/com.android.providers.telephony/mmssms.db Or if dbdata exists, /dbdata/databases/com.android.providers.telephony/mmssms.db MMS in PDU table This path is no longer available in a regular ADB backup. Root is necessary.

SMS	/data/com.android.providers.telephony/mmssms.db /data/com.android.providers.telephony/app_parts (associated media to message) Or if dbdata exists, /dbdata/databases/com.android.providers.telephony/mmssms.db SMS in SMS table This path is no longer available in a regular ADB backup. Root is necessary.
Installed apps	/system/packages.list—text file no permission listings /system/packages.xml—XML file with app permissions
Wi-Fi hotspots	/misc/wifi/wpa_supplicant.conf /wifi/bcm_supp.conf /wifi/networkHistory.txt

Additional files such as system files, log files, and user settings within the Android file system are also available and can reveal many things about the device, including the OS version, mobile device specifics, user information, and other items pertinent to mobile devices use. By relying on a standard logical Android collection, the examiner is limiting his or her examination to a very small percentage of the available data within the mobile device.

"Tip of the Iceberg Data"

The mobile forensic solutions used by the examiner often collect significant data from an Android device. This data is *active* data from the databases within the user data and typically includes data from contacts, call logs, SMS, MMS, media, browser, Wi-Fi, and a few others. This is the "tip of the iceberg data," with the bulk of the information residing "underwater," however—even more so than with iOS. Because an Android logical extraction queries only the active data, the inactive or free page data or any other field not queried will not be collected. At least with iOS, an examiner would have access to the database the mobile solution queried to obtain the data presented in the user interface—but with Android, not so much.

Using advanced collection methods, however, the examiner can collect the entire file system, including database files and others, for advanced analysis. Table 15-4 represents the typical data that is extracted using a mobile solution's APK file. Using the associated paths, the examiner can investigate the various database files for additional information. Also, information logically obtained by the mobile forensic solution can be verified and validated to ensure that dates and times are accurate, along with any missing fields from the reported data. An examiner can pull additional information from these locations that the logical collection missed.

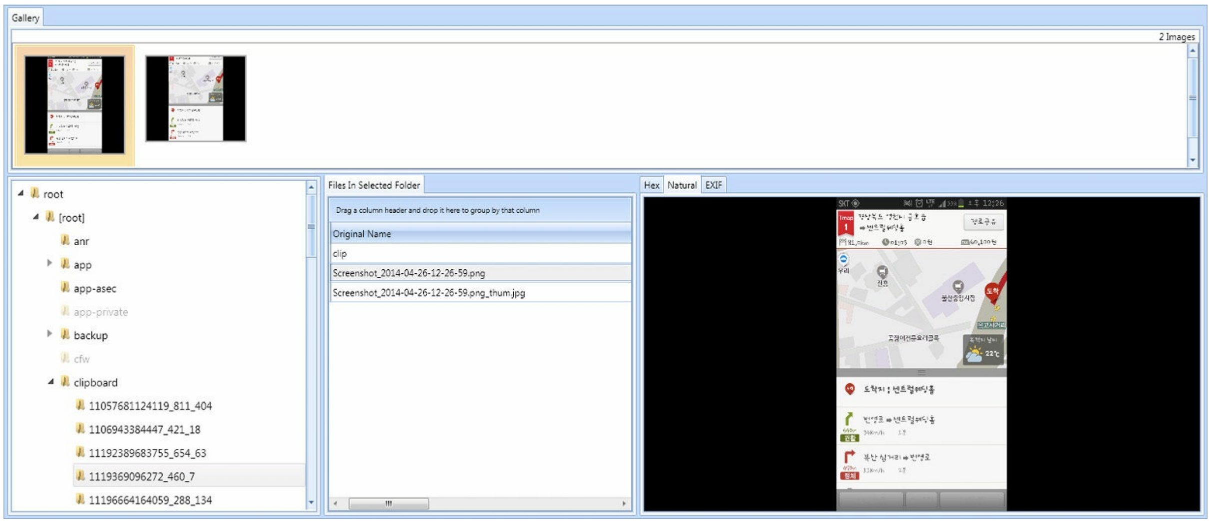
Additional File System Locations

A common mistake made by many examiners is to begin the investigation immediately by going to the app area. Although the app area is a fruitful place and often mesmerizes even the most seasoned examiners, a better idea is to venture into many other areas within the Android file system.

Device configuration files, installed application accounts and device accounts, user settings, device details, and often password information can be gleaned from outside of the app data folder. Needless to say, the actual APK file and the compiled DEX files can be obtained in a file system–level extraction. With the actual files, static malware analysis of the code within the APK and DEX files often helps determine whether malicious code is present. The following section will outline some of the more plentiful areas for gathering system information. All of the root directory information and path data are located in Table 15-3 earlier in this chapter.

Clipboard Information

The path /clipboard/ is generally found only within a Samsung Android device and contains various folders, each identified with a unique number. Pure Android systems store clipboard data in RAM by the Clipboard service. However, each folder holds a file called clip (see Figure 15-7). The clip file header holds a value, `android.sec.clipboard.list.ClipboardData<type of data>`, and the end of the header is `xpt`*X* (*X* = a variable value that changes depending upon the clip file). If the `<type of data>` is `text`, the clipboard content will be a text string; if it's `HTML`, it's a web page; if `Bitmap`, it's an image file. If a bitmap is indicated within the header, the image files will be in the same folder as the clip file. If text or HTML, the data immediately follows the header. In current Samsung devices, in a secondary location, android.samsung.clipboardsaveservice/f/, the examiner can find saved images to the clipboard.


Gallery
2 Images
Hex
Natural
EXIF
Drag a column header and drop it here to group by that column

root
[root]
anr
app
app-asec
app-private
backup
cfw
clipboard
11057681124119_811_404
1106943384447_421_18
11192389683755_654_63
1119369096272_460_7
11196664164059_288_134

Files In Selected Folder

Original Name
clip
Screenshot_2014-04-26-12-26-59.png
Screenshot_2014-04-26-12-26-59.png_thum.jpg

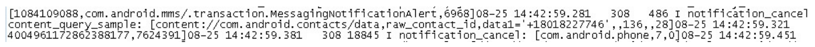
FIGURE 15-7 **A clip file along with the associated image files**

Log Files

In Android devices, particularly Samsung devices, services report errors and then log these errors within the log folder for exceptions within an app or device service. An examiner can find significant information in this folder and associated text files within a physical extraction. However, in a logical ADB extraction, several commercial tools will also include the files Dmesg.log, Dumpsys.log, and Logcat.log. Some text files of interest are listed here:

- **/log/CallDropInfoLog.txt** Listing of failed calls along with the dates and times of the dropped calls. The date is in the format YYYY.MM.DD HH:MM:SS.

- **/log/CallNosvcInfoLog.txt** Listing of calls that were attempted when the device did not have service. The date is in the format YYYY.MM.DD HH:MM:SS.
- **/log/dumpstate_app_error_x.txt.gz** If more than one dumpstate file is within the folder, the *x* will be an ordinal number. This file is a compressed folder created when an app crashes with an uncaught runtime exception. Tools such as 7-Zip can be used to access dumpstate files. Android versions up to 4.4.2 can contain significant user logins, settings, and more. The dumpstate file captures the state of the device at the time of the app failure, often producing data that cannot be found anywhere else on the device. Android 4.4.4 patched some of these vulnerabilities. The following illustration shows a log file produced within an Android 2.3 Samsung SGH-I727 device. It clearly indicates an incoming message alert from a phone number that was critical to an investigation, since the date and time were also logged.



```
[1084109088,com.android.mms/.transaction.MessagingNotificationAlert,6968]08-25 14:42:59.281    308    486 I notification_cancel
content_query_sample: [content://com.android.contacts/data,raw_contact_id,data1='+18018227746',,136,,28]08-25 14:42:59.321
4004961172862388177,7624391]08-25 14:42:59.381    308 18845 I notification_cancel: [com.android.phone,7,0]08-25 14:42:59.451
```

- **/log/dumpstate_app_anr_x.txt.gz** If more than one dumpstate file is within the folder, the *x* will be an ordinal number. This file is a compressed folder that is created when an app receives an App Not Responding (ANR) event. A message to the user of the device is generated, indicating that the app is not responding and the log will be created. This file can be opened with tools such as 7-Zip. By using regular expressions formatted for e-mail, there is a possibility that data can be recovered. Android 4.4.4 patched some of these vulnerabilities.
- **/log/dumpstate_app_native_x.txt.gz** If more than one dumpstate file is within the folder, the *x* will be an ordinal number. This file is a compressed folder that is created when an app error occurs during a process. This error does not cause an exception and subsequent crash, and the user is not notified because this process is moved into the background services. This file can be opened with tools such as 7-Zip. By using search terms such as *contentView*, *contentText*, and *tickerText*, an examiner can find relevant data within some Samsung device log files. Android 4.4.4 patched some of these vulnerabilities.
- **Dmesg.log** This log will report health information about the device to include temperature, battery operation, current, and more. This documentation log is created with the device system reports information on the device.
- **Dumpsys.log** This log will report on application starts, current applications running, reboots, resets, Bluetooth data to include MAC, BSSID, SSID address, and much more. This file can be used for the investigation of a reset device or to get the application last used if there are no other details extracted by the software.
- **Logcat.log** This file can capture application logs that are running on the device. For example, the syncing of contacts to an automobile via Bluetooth can be identified within the logcat.log file, along with many other interactions the system has with apps on the device.

System Configuration

Settings that persist over restarts on an Android device can be stored within the /property folder and can assist an examiner in determining user and device settings. These files are all prefixed with *persist* and then the <data description>.

- **data/property/persist.radio.cdma.mdn** This setting within a CDMA device, if available, will show the device's mobile dialing number (MDN), the device phone number.
- **data/property/persist.service.adb.enable** This setting indicates whether or not Android Debug Bridge (ADB) is enabled on the device. This is a Boolean value; 0 is false and 1 is true. This will generally be 1 if the device was collected using mobile forensic tools, so the state will not determine whether the user had enabled ADB. However, if the examiner uses JTAG or chip-off techniques, this value will be indicative of the user's actions.
- **data/property/persist.sys.country** This setting indicates the country in which the device service is registered, which may or may not indicate where the device was last or is currently located. This will be a two-character representation such as US, KR, or UK.
- **data/property/persist.sys.locale** This setting indicates the language and country in which the device service is registered, which may or may not indicate where the device was last or is currently located. This will be a two-character representation such as US, KR, or UK.
- **/property/persist.sys.language** This setting indicates the language the device is currently set to display and use, represented by two characters; these identify the culture code used by the device for letters, numbers, signs, and so on. The file persist.sys.locale contains the language now.
- **dataproperty/persist.sys.timezone** This setting is the last system time zone update to the device. Since most mobile devices update their time zone, this value will represent the last time zone location of the device. This can be obtained via a cellular or Wi-Fi connection, so an examiner must be sure to isolate the device because this value could update if the device connects to another network.
- **data/property/persist.usb.config** This value indicates what services will be enabled when the device is plugged into a computer. Here are some examples:
  - **mtp** Only MTP will be available.
  - **mtp, adb** Both services will be active when the device is plugged in. This will be seen frequently if the examiner is using mobile forensic tools that root the device, and a true value will generally be available only with a JTAG or chip-off collection.
- **/property/persist.mot.encrypt.mmc** This Boolean value is present only in a Motorola Android and indicates whether or not the external media card will be encrypted.

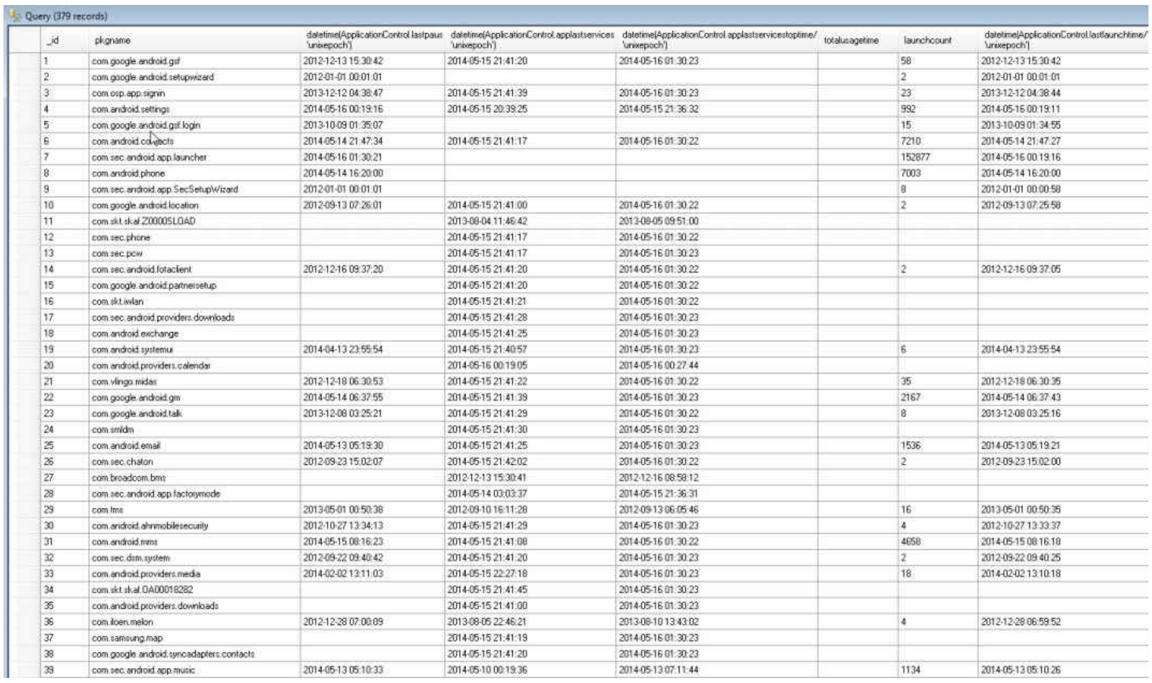
Usage and Logs

The /system folder contains a wealth of information for the examiner, including more settings and storage files for Android system functions, account information, security information, and app listings.

- **data/system/accounts.db** This SQLite database lists all of the accounts that are used on the Android device. The app username and password can be located in this file. As of version 2.3, the password value is no longer plain text but a Base64-encrypted value. For an examiner, this can be valuable information for identifying usernames for specific Android accounts.
- **data/system/user/0/accounts.db** This SQLite database lists all of the accounts that are used on the Android device. The app username and password/token can often be located in this file. For an examiner,

this can be valuable information for identifying usernames for specific Android accounts.

- **/system/dmappmgr.db** This is a fantastic SQLite database (Figure 15-8) for an examiner to determine apps that are frequently used and apps found on Samsung devices that had at one time been installed on the device and are no longer there, including the last time the app was launched. For an examiner, being able to determine app usage and services used can be critical to an investigation.



Query (379 records)

_id	pkgname	datetime(ApplicationControl.lastpaus 'unixepoch')	datetime(ApplicationControl.applastservices 'unixepoch')	datetime(ApplicationControl.applastservicestoptime/ 'unixepoch')	totalusagetime	launchcount	datetime(ApplicationControl.lastlaunchtime/ 'unixepoch')
1	com.google.android.gsf	2012-12-13 15:30:42	2014-05-15 21:41:20	2014-05-16 01:30:23		58	2012-12-13 15:30:42
2	com.google.android.setupwizard	2012-01-01 00:01:01				2	2012-01-01 00:01:01
3	com.osp.app.signin	2013-12-12 04:38:47	2014-05-15 21:41:39	2014-05-16 01:30:23		23	2013-12-12 04:38:44
4	com.android.settings	2014-05-16 00:19:16	2014-05-15 20:39:25	2014-05-15 21:36:32		992	2014-05-16 00:19:11
5	com.google.android.gsf.login	2013-10-09 01:35:07				15	2013-10-09 01:34:55
6	com.android.contacts	2014-05-14 21:47:34	2014-05-15 21:41:17	2014-05-16 01:30:22		7210	2014-05-14 21:47:27
7	com.sec.android.app.launcher	2014-05-16 01:30:21				152877	2014-05-16 00:19:16
8	com.android.phone	2014-05-14 16:20:00				7003	2014-05-14 16:20:00
9	com.sec.android.app.SecSetupWizard	2012-01-01 00:01:01				8	2012-01-01 00:00:58
10	com.google.android.location	2012-09-13 07:26:01	2014-05-15 21:41:00	2014-05-16 01:30:22		2	2012-09-13 07:25:58
11	com.skt.skaf.Z0000SLOAD		2013-08-04 11:46:42	2013-08-05 09:51:00			
12	com.sec.phone		2014-05-15 21:41:17	2014-05-16 01:30:22			
13	com.sec.pcw		2014-05-15 21:41:17	2014-05-16 01:30:23			
14	com.sec.android.fotaclient	2012-12-16 09:37:20	2014-05-15 21:41:20	2014-05-16 01:30:22		2	2012-12-16 09:37:05
15	com.google.android.partnersetup		2014-05-15 21:41:20	2014-05-16 01:30:22			
16	com.skt.iwlan		2014-05-15 21:41:21	2014-05-16 01:30:22			
17	com.sec.android.providers.downloads		2014-05-15 21:41:28	2014-05-16 01:30:23			
18	com.android.exchange		2014-05-15 21:41:25	2014-05-16 01:30:23			
19	com.android.systemui	2014-04-13 23:55:54	2014-05-15 21:40:57	2014-05-16 01:30:23		6	2014-04-13 23:55:54
20	com.android.providers.calendar		2014-05-16 00:19:05	2014-05-16 00:27:44			
21	com.vlingo.midas	2012-12-18 06:30:53	2014-05-15 21:41:22	2014-05-16 01:30:22		35	2012-12-18 06:30:35
22	com.google.android.gm	2014-05-14 06:37:55	2014-05-15 21:41:39	2014-05-16 01:30:23		2167	2014-05-14 06:37:43
23	com.google.android.talk	2013-12-08 03:25:21	2014-05-15 21:41:29	2014-05-16 01:30:22		8	2013-12-08 03:25:16
24	com.smldm		2014-05-15 21:41:30	2014-05-16 01:30:23			
25	com.android.email	2014-05-13 05:19:30	2014-05-15 21:41:25	2014-05-16 01:30:23		1536	2014-05-13 05:19:21
26	com.sec.chaton	2012-09-23 15:02:07	2014-05-15 21:42:02	2014-05-16 01:30:22		2	2012-09-23 15:02:00
27	com.broadcom.bms		2012-12-13 15:30:41	2012-12-16 08:58:12			
28	com.sec.android.app.factorymode		2014-05-14 03:03:37	2014-05-15 21:36:31			
29	com.fms	2013-05-01 00:50:38	2012-09-10 16:11:28	2012-09-13 06:05:46		16	2013-05-01 00:50:35
30	com.android.ahnmobilesecurity	2012-10-27 13:34:13	2014-05-15 21:41:29	2014-05-16 01:30:23		4	2012-10-27 13:33:37
31	com.android.mms	2014-05-15 08:16:23	2014-05-15 21:41:08	2014-05-16 01:30:22		4658	2014-05-15 08:16:18
32	com.sec.dsm.system	2012-09-22 09:40:42	2014-05-15 21:41:20	2014-05-16 01:30:23		2	2012-09-22 09:40:25
33	com.android.providers.media	2014-02-02 13:11:03	2014-05-15 22:27:18	2014-05-16 01:30:23		18	2014-02-02 13:10:18
34	com.skt.skaf.OA00018282		2014-05-15 21:41:45	2014-05-16 01:30:23			
35	com.android.providers.downloads		2014-05-15 21:41:00	2014-05-16 01:30:23			
36	com.iloen.melon	2012-12-28 07:00:09	2013-08-05 22:46:21	2013-08-10 13:43:02		4	2012-12-28 06:59:52
37	com.samsung.map		2014-05-15 21:41:19	2014-05-16 01:30:23			
38	com.google.android.syncadapters.contacts		2014-05-15 21:41:20	2014-05-16 01:30:23			
39	com.sec.android.app.music	2014-05-13 05:10:33	2014-05-10 00:19:36	2014-05-13 07:11:44		1134	2014-05-13 05:10:26

FIGURE 15-8 **Actual dmappmgr.db from a Samsung Android device showing the 379 app records outlining the app name, number of launches, and critical date and time information**

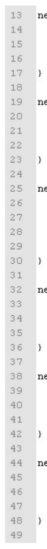
- **data/system/recent_images** This folder contains the screenshots of the corresponding items (app) and actions found within the recent_tasks folder. This could be a gold mine, almost a play-by-play of some actions by the user on the phone, along with images of the actions.
- **data/system/recent_tasks** This folder contains XML files the contain the task ID as well as the activity being utilized by the task, along with date and time information for the first activity time, the last activity time, and the last time moved. This, together with the recent_images, is a great find for the investigation.
- **/system/gesture.key** This file contains the SHA-1 value of the device gesture lock if it was used. The gesture.key file is a hash of the pattern in a byte array. Many online sources can be used to obtain the gesture coordinates from the SHA-1 hash, but this is not really necessary for an analysis of the file system. If the examiner wanted to examine the device powered on and navigate the user interface, he or she could replace the gesture.key with a hash that has been generated in another Android device and use that file instead of the original. The examiner can then enter the created gesture to unlock the device. This is deprecated, and now the gatekeeper.pattern.key is used. The table lockscreen.password_salt is now in /system/locksettings.db, with the added security of being device-specific.
- **/system/packages.list** This text file identifies the app location and the app that is currently on the Android device.

- **/system/packages.xml** This XML file identifies the app location and the app that is currently on the Android device, plus the app permissions.
- **/system/password.key** This file contains the salted SHA-1/MD5 calculation concatenated. The salt is located in the settings.db secure table under the key lockscreen.password_salt within the /data/com.android.providers.settings folder. If the examiner is looking at this file, access has already been made, and accessing the device via the UI is probably unnecessary. If necessary, the examiner can use the salt and the hash with a tool such as hashkill. This is deprecated, and now the gatekeeper.pattern.key is used. The table lockscreen.password_salt is now in /system/locksettings.db, with the added security of being device-specific.
- **/system/SimCard.dat** This text file lists the previous and current SIM operator, SIMSerialNumber, PhoneNumber, CurrentSIMCountry, SIMOperator, SIMOperatorName, CurrentSIMSerialNumber, and SimChangeTime, which is encoded in milliseconds. For an examiner, this can be a location to obtain the SIM information of previous UICC cards inserted and the current UICC, even if the UICC is not in the device at the time of collection. This information can be found at /data/com.android.providers.telephony/databases/telephone.db with current devices.
- **/system/sync/accounts.xml** Android devices enable the user to sync all data across accounts. For an examiner, this can create recoverable evidence, because the account usernames for apps are listed in this file. The examiner can link the username to the associated app using the information found in the accounts.xml file.
- **/system/usagestats/usage - <*date*>** These files can assist the examiner when looking for apps on a particular date. The text file is named usage - <YYYYMMDD> and is a Unicode binary file.
- **data/system/usagestats** This folder has a subfolder, usually 0, which then contains other subfolders: daily, monthly, weekly, and yearly. XML files within the folders include all the apps and details of each. The details are similar to those found in the dmappmgr.db in Samsung devices described earlier.

Wi-Fi Information

Wi-Fi information can be critical for many investigations. The data located within the /misc/wifi folder is recoverable and contains Wi-Fi and hotspot information. This data often shows the SSID and configuration settings.

- **/misc/wifi/networkHistory.txt** This offers much more verbose connection information than what is found in the wpa_supplicant.conf file on most devices, to include connection times (date/time) and failures.
- **/misc/wifi/wpa_supplicant.conf or /wifi/bcm_supp.conf** With logical extractions using an APK, the Wi-Fi information is collected, often without passwords. By navigating to the location within the file system, the examiner can locate each Wi-Fi spot to which the device was connected, along with the SSID and password. This information is stored within a dictionary and is readable in a text editor (see Figure 15-9).



```
network={
    ssid="3800OBD-302"
    psk="3800obd302"
    priority=1
}

network={
    ssid="timeoutboise"
    key_mgmt=NONE
    priority=2
}

network={
    ssid="807Wind"
    psk="807windemere"
    key_mgmt=WPA-PSK
    priority=3
}

network={
    ssid="BWLasVegasWest"
    key_mgmt=NONE
    priority=4
}

network={
    ssid="NewRoyalPlaya"
    key_mgmt=NONE
    priority=12
}

network={
    ssid="NewRoyalPlaya388"
    key_mgmt=NONE
    priority=13
}
```

FIGURE 15-9 **Information from the wpa_supplicant.conf file is clearly readable within a text editor and contains access point names and passwords.**

- **/misc/wifi/hostapd.conf** If the device was set as a hotspot, the information in this file will identify the SSID that was transmitted by the device and the hash of the password.
- **/misc/wifi/softap.conf** This contains the SSID of the hotspot and password configuration.

/data Folder

The primary source for examiner gold is the /data folder. This folder contains not only the stock Android apps such as e-mail, contacts, call logs, SMS, and MMS but also device-specific apps that may be found only on Samsung devices. This folder will also contain third-party apps and content. Unlike iOS, which has a distinct folder for iOS stock apps and another for third-party apps, the Android does not make that distinction. All apps and even some Android settings files are located within the /data folder. As referenced previously in Table 15-4, some Samsung devices also use a dbdata partition to store this same data, and references within this section will indicate this distinction. This section covers Android settings locations, stock Android apps, some Samsung-specific databases, and third-party apps.

Like iOS, Android will prescribe the reverse-DNS naming scheme for the apps located in the /data folder using the top-level domain (such as com, net, kr, org), the domain, and then the app name as the package name (pkg). Some apps use subdomains as well. For example, Samsung uses com.sec to prefix its stock applications (*sec* stands for Samsung Electrical Company).

The app folders have a hierarchy much like that of iOS. Android development API and SDK guidelines direct developers on how an app should store data, but like iOS, an app can create folders and store data in these folders if it must persist. Android does indicate that some folders must be used for certain circumstances, and these folders and the other common folders are outlined in Table 15-5. Note that Android development API information specifically outlines the protection of data during backups. This means that a standard ADB backup of Android devices can be missing many third-party application databases. This is not a limitation of the software or ADB, but is simply set by the app developer in the manifest during development.

TABLE 15-5 **App Folder Common Directories and Associated Content**

Folder	Description
/cache	Temporary data for the Android application. This folder can have many subfolders organized according to the type of files within the folder (such as audio, image, upload).
/databases	One or more SQLite databases to support the main app and significant data for the examiner.
/files	App files that do not belong within the database or are referenced from the database. An app developer can use this area for many different file types.
/lib	Library files for the app. These files are generally .so files, which are library files for Linux and used by the app to perform functions. This folder is now found in the /data/app path.
shared_prefs	Settings that can be accessible by other apps and the mobile device. These files are often XML files that can contain valuable information for an examiner.

Stock Android Folders

Android devices usually contain these folders within the /data folder. Android has two levels of locations that store data—one level uses com.android.providers.<*name of app*> and the other uses com.android.<*name of app*>. The com.android.providers apps use the android.providers class, which allows other apps to access the content providers within the apps that are using this class. These apps are generally those with contacts, images, calendar, and SMS. An examiner should recognize all of these items using this class since most of them are artifacts that can be extracted using a logical collection. Because mobile forensic solutions use an APK file to access the content stores within an Android device, the accessible data types must conform to the classes allowed by Android. The standard com.android apps are stock Android apps that ship with the OS; they do not generally allow com.android.provider type control and are infrequently used. A few stock com.android apps contain data, such as com.android.browser, but typically these apps are void of user data. However, most Android devices also use the /data/com.google.android<*name of app*> path in every device encounter. Some of these locations will also be covered as a semi-default Android stock app with new devices. The following stock Android folders are the places an examiner will likely uncover valuable data on each Android investigation:

- **/data/com.android.browser/databases/browser2.db** This is the SQLite database of the built-in Android browser that contains tables for bookmarks, history, and user searches. This area should be investigated even if the mobile solution parses this data, because some columns are not reported by automated solutions. This has been deprecated in many Android devices. The path /data/com.android.browser.provider is still available in devices such as Samsung, but only with physical collections. However, the Samsung default is Samsung Internet Browser and is collected with both physical and logical collections at com.sec.android.app.sbrowser. On many Android devices the default browser has been replaced by the Chrome browser located at /data/com.android.chrome/app_chrome/Default/

- **/dbdata/databases/com.android.browser/databases/browser.db** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series. This has been deprecated in newer Samsung Androids.
- **/data/com.android.browser/databases/autofill.db** This SQLite database stores user-saved information such as name and address information to be filled into forms stored within the browser history. This has been deprecated in many Android devices. The path /data/com.android.browser.provider/ path is still available in devices such as Samsung, but only with physical collections.
- **/data/com.android.browser/app-databases/Databases.db** This SQLite database stores the app and path information for apps using the Android web browser via WebKit functions. The mobile web app database will be stored within the app-databases folder identified within the Databases.db and can be explored to uncover use of that mobile web app. This has been deprecated in many Android devices. The path /data/com.android.browser.provider/ path is still available in devices such as Samsung, but only with physical collections.
- **/dbdata/databases/com.android.browser/app-databases/Databases.db** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series.
- **/data/com.android.browser/app-geolocation/CachedGeoposition.db** This SQLite database stores the app geolocation information if the user enables this feature when asked within the browser. This is typically seen within WebKit apps such as Yelp, Walmart, Google, and many others. This file will show the date and time, in milliseconds, when the location was requested. This has been deprecated in many Android devices. The path /data/com.android.browser.provider/ path is still available in devices such as Samsung, but only with physical collections.
- **/dbdata/databases/com.android.browser/app-geolocation/CachedGeoposition.db** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series.
- **/data/com.android.chrome/app_chrome/Default/** In many Android devices, the Chrome browser is installed by default. The main databases are located in this area and have been previously described:
  - **Cookies** This SQLite database contains the Chrome cookies that give creation date, host, name, and last access in milliseconds.
  - **History** This SQLite database contains the downloads, URLs, and visits of the user. The urls table contains the URL, number of visits, and last visited time.
  - **Login Data** This SQLite database contains usernames, passwords/token, times used, and date created.
  - **Top Sites** This SQLite database contains the URL and thumbnails of web sites. This thumbnail data can be converted to a JPEG and viewed, oftentimes obtaining user information of the visit.
  - **Web Data** This SQLite database is often contained in many applications that can access web pages within their application. These databases can contain credit card numbers, keyword searches, autofill

data, and server addresses. This file should not be overlooked in a forensic examination.

- **/data/com.android.chrome/cache/Cache/** This location is full of web site page data containing IP routing, URLs, search strings, and more. This is the temporary Internet location that should not be overlooked.
- **/data/com.android.email/databases/EmailProvider.db** This SQLite database stores the built-in Android e-mail client data for the user. The primary tables are Message, Attachment, and EmailAddressCache. The secondary database located in this same folder is called EmailProviderBody.db, and it contains the actual content of the entire e-mail message JOINed using the messageKey in the body table. Of course, if a user uses another app for e-mail, these databases could hold limited information. This location is now /data/com.android.email/db/EmailProvider.db.
- **/data/com.android.email/cache/** This folder can store many files that were embedded into the sent e-mails, including pictures, HTML banners, and much more. An examiner can find many artifacts in the cache area that have been deleted from e-mail or other file locations on the device. This appears to be deprecated in new Android devices.
- **/data/com.android.mms/db/message_glance.db** This database is of interest and is extracted in a regular ADB backup and contains tables events, logs, and priority_sender. The table event lists phone numbers as well as the date of the event. In the logs table the address, which can be a phone number or e-mail address, is available, as are the direction of the contact (incoming/outgoing), how many events were made, and whether an event was deleted. The priority_sender includes the phone numbers that are saved as favorites. The content of the messages is not available.
- **/data/com.android.phone/databases/** This folder can contain several SQLite databases, including autoreject.db, callreminder.db, and rejectmessage.db, if the user has set them up to handle certain phone tasks. These databases will not be created unless the user has set them up. The database names identify the action and subsequent storage of that data. Autoreject stores numbers the user has identified to be rejected when receiving a call, callreminder will notify the user of a call, and rejectmessage enables the user to send a customized message on rejection of calls.
- **/data/com.android.vending/databases/localappstate.db** This database stores installed apps and the installation date along with the username associated with the app. The path has changed to /data/com.android.vending/db/.
- **/data/com.android.vending/databases/suggestions.db** This database stores search terms used by the device user when searching for Android apps to purchase. The path has changed to /data/com.android.vending/ and db or databases (depending on the device).
- **/data/com.android.providers.download/databases/downloads.db** Android devices have always controlled the information that is downloaded from the Internet independently of the app using class DownloadManager. Apps use this class when a file is downloaded to the device, and the information related to the download is stored within this database. This includes downloaded e-mail attachments, downloaded web content, and much more. If the file no longer exists on the mobile device, the metadata can possibly still be recovered from this database to indicate that it was, in fact, downloaded to the device. The path has changed to /data/com.android.providers.download/ and db or databases (depending on

device).

- **dbdata/databases/com.android.providers.download/downloads.db** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series.
- **/data/com.android.providers.media/databases/external<*variable*>.db** Android devices use another class called MediaStore, which caches all media on the internal device and external media. The media can include images, videos, audio, and even files. This database holds any data within the storage areas that does not contain a file named .nomedia. If the .nomedia file is encountered, the MediaScanner will not search that folder, and the information will not be cached in the external.db file. The <*variable*> listed in the path can be seen in some Android devices—it is the actual serial number of the SD card that was inserted. Some Android devices' com.android.providers.media folder can contain multiple external.db files because multiple external SD cards were inserted, and subsequently the MediaScanner scanned and documented the data from each, storing the data in independent files. The external.db SQLite database contains many tables, but the most important is the files table, which contains all the files that have been scanned, including dates/times, paths, geolocation, size, and what app the file is associated with—to name a few. An examiner can gather critical information on the files within the Android mobile device with associated metadata. The path has changed to /data/com.android.providers.media/ and db or databases (depending on device).
- **/data/com.android.providers.media/databases/internal.db** The internal.db file stores media paths to Android built-in system needs. The information found in the internal.db is generally of no use to an examiner because the information is system-based, not user-based, data. The path has changed to /data/com.android.providers.media/ and db or databases (depending on device).
- **/dbdata/databases/com.android.providers.media/** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series.
- **/data/com.android.providers.settings/databases/settings.db** The settings.db file is used by the device to persist all of the user's settings within a SQLite database. The tables of interest within the database are the system and secure tables. The system table identifies the setup of the mobile device, including the location of service prior to collection within the aw_daemin_service_key_city_name key and also whether the device was in hands-free mode using the driving_mode_on key. Both of these keys can provide critical information for an examiner, and the settings of the device at the time of collection can be found within this database.
- **/dbdata/databases/com.android.providers.settings/settings.db** Same as the previous entry, but this location can be found within some Samsung devices, particularly the Samsung Galaxy S series.
- **/data/com.android.providers.telephony/optable.db** This database is used to store the carriers on which the mobile device has operated and lists the carrier name and the public land mobile network (PLMN) identifier within the operator table.
- **/data/com.android.providers.telephony/databases/telephony.db** This database is used to store the carriers on which the mobile device has operated and lists the carrier name and the PLMN identifier within the operator table. This also lists the SIM or SIMs that have been used by the device in the siminfo table. This provides the carrier name for each SIM that had been used.

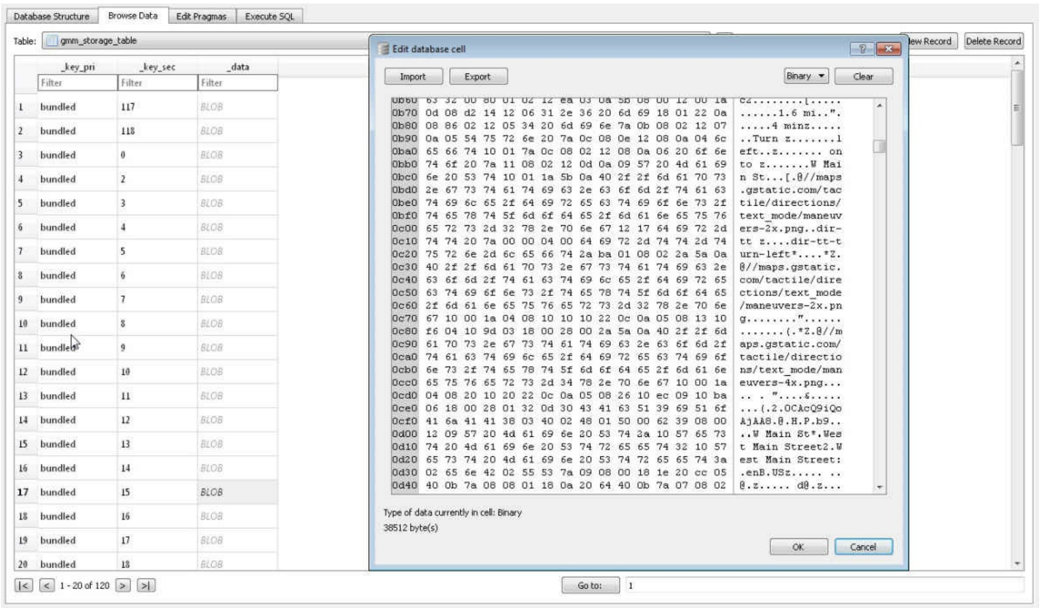
- **/data/com.android.providers.telephony/databases/mmssms.db** This database stores all the MMS and SMS messages. The MMS information is stored within the pdu table, and attachments and content can be associated with the parts table. SMS uses the sms table, and several other significant tables reside within this database, including words and words_content, which both store text-based data that can prove valuable for an examiner. The words table is used by the sms and pdu tables to store searchable, indexed data for a mobile device, and the words_content table is used by the pdu table to store indexed, searchable text-based data for a mobile device. Because the pdu table uses both the words and words_content table, some data can be duplicated, but it can be filtered by using the foreign key in both tables. If the MMS or SMS session does not use a delete function, the information that is within the words and words_content may remain and can be harvested by the examiner even if the original SMS or MMS is deleted. This database is no longer backed up in a regular ADB backup, but can be extracted with a logical extraction using an agent. However, with the logical extraction and agent, deleted records from the database or journal/WAL file will not be recovered.
- **/data/com.android.providers.telephony/app-parts/** This folder contains the attached files to MMS messages and is referenced in the mmssms.db parts table.
- **/dbdata/databases/com.android.providers.telephony/mmssms.db** Same as the previous entry, but this location is within some Samsung family devices, particularly the Samsung Galaxy S series.

Google Stock Apps

Because Google is Android and Android is Google, it should be no surprise to any examiner that most Android devices are populated with stock apps from Google. The following outlines some of these applications and where an examiner may find additional artifacts while examining the Android file system:

- **/data/com.google.android.apps.docs/databases/DocList.db** This database contains the list of files for Google Drive (in table Entry143). In this database, the filename, modified date, owner account, and other details are documented. Also, the table DocumentContent143 is the listing of files that are local and on the device, which can be found using the filepath field. Within the path /data/com.google.android.apps.docs/ are additional folders such as files/fileinternal/<GUID> that holds the files referenced in the filepath field of the DocList.db.
- **/data/com.google.android.apps.docs.editors<docs,sheets,slides>/databases/** This location will house the database depending on the path of docs, sheets, and slides, and will contain information on created documents in the Google Docs platform.
- **/data/com.google.android.apps.maps/databases/gmm_myplaces.db** This database contains saved places or Google Maps favorites. The information is stored within the sync_item table and lists the latitude and longitude, time stamp, and the sync_item as a BLOB file, which is the location name along with the HTTP data to maps.google.com.
- **/data/com.google.android.apps.maps/databases/gmm_strorage.db** This database is the storage location for the turn-by-turn instructions for each route entered in the Google Maps app. The information is stored in the gmm_storage_table in the data field. The data field is BLOB data, and, when viewed, it gives the examiner a view of the directions followed by the user to a destination—the location name along with

the HTTP data to maps.google.com. Figure 15-10 shows BLOB data within the gmm_storage_table that can assist an examiner when using mapping data in a case.



	_key_pri	_key_sec	_data
1	bundled	117	BLOB
2	bundled	118	BLOB
3	bundled	0	BLOB
4	bundled	2	BLOB
5	bundled	3	BLOB
6	bundled	4	BLOB
7	bundled	5	BLOB
8	bundled	6	BLOB
9	bundled	7	BLOB
10	bundled	8	BLOB
11	bundled	9	BLOB
12	bundled	10	BLOB
13	bundled	11	BLOB
14	bundled	12	BLOB
15	bundled	13	BLOB
16	bundled	14	BLOB
17	bundled	15	BLOB
18	bundled	16	BLOB
19	bundled	17	BLOB
20	bundled	18	BLOB

```
0b60 63 32 00 80 01 02 12 ea 03 0a 5b 08 00 12 00 1a  cz........[.....
0b70 0d 08 d2 14 12 06 31 2e 36 20 6d 69 18 01 22 0a  ......1.6 mi..".
0b80 08 86 02 12 05 34 20 6d 69 6e 7a 0b 08 02 12 07  .....4 minz.....
0b90 0a 05 54 75 72 6e 20 7a 0c 08 0e 12 08 0a 04 6c  ..Turn z.......l
0ba0 65 66 74 10 01 7a 0c 08 02 12 08 0a 06 20 6f 6e  eft..z....... on
0bb0 74 6f 20 7a 11 08 02 12 0d 0a 09 57 20 4d 61 69  to z.......W Mai
0bc0 6e 20 53 74 10 01 1a 5b 0a 40 2f 2f 6d 61 70 73  n St...[.@//maps
0bd0 2e 67 73 74 61 74 69 63 2e 63 6f 6d 2f 74 61 63  .gstatic.com/tac
0be0 74 69 6c 65 2f 64 69 72 65 63 74 69 6f 6e 73 2f  tile/directions/
0bf0 74 65 78 74 5f 6d 6f 64 65 2f 6d 61 6e 65 75 76  text_mode/maneuv
0c00 65 72 73 2d 32 78 2e 70 6e 67 12 17 64 69 72 2d  ers-2x.png..dir-
0c10 74 74 20 7a 00 00 04 00 64 69 72 2d 74 74 2d 74  tt z....dir-tt-t
0c20 75 72 6e 2d 6c 65 66 74 2a ba 01 08 02 2a 5a 0a  urn-left*....*Z.
0c30 40 2f 2f 6d 61 70 73 2e 67 73 74 61 74 69 63 2e  @//maps.gstatic.
0c40 63 6f 6d 2f 74 61 63 74 69 6c 65 2f 64 69 72 65  com/tactile/dire
0c50 63 74 69 6f 6e 73 2f 74 65 78 74 5f 6d 6f 64 65  ctions/text_mode
0c60 2f 6d 61 6e 65 75 76 65 72 73 2d 32 78 2e 70 6e  /maneuvers-2x.pn
0c70 67 10 00 1a 04 08 10 10 10 22 0c 0a 05 08 13 10  g........"......
0c80 f6 04 10 9d 03 18 00 28 00 2a 5a 0a 40 2f 2f 6d  .......(.*Z.@//m
0c90 61 70 73 2e 67 73 74 61 74 69 63 2e 63 6f 6d 2f  aps.gstatic.com/
0ca0 74 61 63 74 69 6c 65 2f 64 69 72 65 63 74 69 6f  tactile/directio
0cb0 6e 73 2f 74 65 78 74 5f 6d 6f 64 65 2f 6d 61 6e  ns/text_mode/man
0cc0 65 75 76 65 72 73 2d 34 78 2e 70 6e 67 10 00 1a  euvers-4x.png...
0cd0 04 08 20 10 20 22 0c 0a 05 08 26 10 ec 09 10 ba  .. . ".....&.....
0ce0 06 18 00 28 01 32 0d 30 43 41 63 51 39 69 51 6f  ...(.2.0CAcQ9iQo
0cf0 41 6a 41 41 38 03 40 02 48 01 50 00 62 39 08 00  AjAA8.@.H.P.b9..
0d00 12 09 57 20 4d 61 69 6e 20 53 74 2a 10 57 65 73  ..W Main St*.Wes
0d10 74 20 4d 61 69 6e 20 53 74 72 65 65 74 32 10 57  t Main Street2.W
0d20 65 73 74 20 4d 61 69 6e 20 53 74 72 65 65 74 3a  est Main Street:
0d30 02 65 6e 42 02 55 53 7a 09 08 00 18 1e 20 cc 05  .enB.USz..... ..
0d40 40 0b 7a 08 08 01 18 0a 20 64 40 0b 7a 07 08 02  @.z..... d@.z...
```

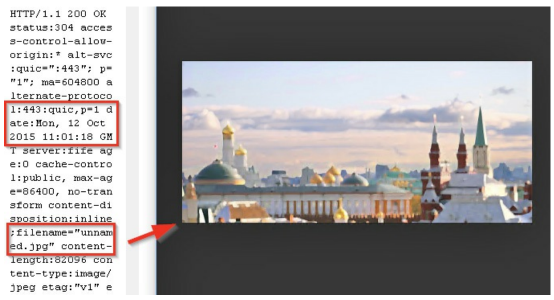
Type of data currently in cell: Binary
38512 byte(s)

FIGURE 15-10 **BLOB data within the gmm_storage_table**

- **/data/com.google.android.apps.maps/databases/search_history.db** This database records the searched locations within the Google Maps app, storing the latitude and longitude along with a time stamp. This data is in the suggestions table in the SQLite database.
- **/data/com.google.android.apps.maps/app_da_speech/** This folder stores the voice WAV files for the directions given.
- **/data/com.google.android.apps.maps/cache/http/** This folder stores cached data in the form of GUID named files. Generally two related files with the same GUID are differentiated by a *0* or *1*; *0*-ending files are the description of the associated file ending in *1*. This can be helpful for an examiner, since some street-view JPGs are stored in this location when a user requests that view of a location.
- **/data/com.google.android.apps.maps/shared_prefs/settings_preferences.xml** This XML file contains values Google Maps is using. One key, current_account_name, lists the registered account name for Google Maps. This information can be used to obtain court-ordered information on geolocation data that's not stored on the mobile device but might be stored on the user's account.
- **/data/com.google.android.apps.maps/databases/ue3.db** This new database for Google Maps contains the table userevent3_table that contains the account information, the field ue3 that contains BLOB data with directional information, and a third field time stamp. This is much like the legacy gmm_storage_table database.
- **/data/com.google.android.maps/databases/mytracks.db** This new database for Google Maps contains all of the tracks, waypoints, and trackpoints. The trackpoints table contains date/time information, latitude and longitude, speed, and elevation. The waypoints contain the pin description, address, name, latitude and longitude, date/time of start and stop, and several other fields of interest.

- **/data/com.google.android.apps.gm/databases/mailstore.<*username*>.db** This Google app store is for Gmail. Each mailbox is prefixed by *mailstore* and then the Gmail e-mail address. The SQLite database contains several tables, including messages, attachments, and conversations. The messages table contains the date and time information for e-mail messages, the subject, and a snippet of the e-mail message. The conversation table is used to present the e-mail on the device visually to the user without showing all the content—allowing for better app performance. Since content is stored on the Gmail server and only a portion of the mail is available to an examiner in the file system, sometimes the only indication of mail, if not in the messages table, is cached in the conversation table. Also found on Android devices with built-in Gmail at /data/com.google.android.gm/databases/mailstore.<*username*>.db.
- **/data/com.google.android.apps.gm/databases/suggestions.db** This SQLite database is found throughout apps within an Android and represents what a user searched for within the app itself using the search bar. Also found on Android devices with built-in Gmail at /data/com.google.android.gm/databases/suggestions.db.
- **/data/com.google.android.apps.gm/cache/** This folder contains several folders representing the various Gmail accounts within the /databases folder. In each folder are files cached from attachments and e-mail. Also, a webViewCacheChromium folder stores temporary cache data to support web data browsing and e-mail–embedded items using HTML pages. Also found on Android devices with built-in Gmail at /data/com.google.android.gm/cache.<*username*>. Also, within the cache folder, the webViewCacheChromium folder name has changed to org.chromium.android_webview.
- **/data/com.google.android.apps.messaging/databases/bugle_db** This SQLite file is the main Android messenger database and contains conversations, messages, participants, and parts, among other tables. Using the data from the messages table, parts table, and participants table, an examiner can easily uncover messages that a commercial tool does not support.
- **/data/com.google.android.apps.plus/databases/es0.db** This SQLite file is the main Google+ database and contains circles, all_photos, activities, activity_comments, and stories, among other tables. Also contained with the /databases folder is the iu.upload.db file, which lists files uploaded to the user's Google+ account. Google+ is deprecated but data is still available here as well as databases within /data/com.google.android.gms/.
- **/data/com.google.android.apps.plus/cache/media/** This folder contains subfolders referenced by both the es0.db and iu.upload.db files. Each folder has numerous media files. Using a data carver in this folder system will yield many media items an examiner may need in an investigation. Google+ is deprecated but data is still available as well as databases within /data/com.google.android.gms/.
- **/data/com.google.android.apps.plus/shared_prefs/accounts.xml** This XML file contains the account information for the Google+ account. Google+ is deprecated but data is still available as well as databases within /data/com.google.android.gms/.
- **/data/com.google.android.gms/** Within this directory for the Google Play services are many databases relating to Google apps that are using the Google Play service function. Databases include android_pay, auto_complete_suggenstions.db, context_<*Gmail account name*>, fitness.db.<email of user>.com, google_account_history, plus.db, pluscontacts.db, reminders.db, and many others.

- **/data/com.google.android.googlequicksearchbox/databases/icingcorpora.db** This SQLite file is used to store indexed applications, contacts, e-mails, phone calls, and postal codes. Within the database, the tables to view are applications, contacts, e-mails, phones, and postal codes. These tables list values that are most prevalent for that type based upon the score field. The value ranges from 1 to *n*. The highest *n*-valued item in the score column for each type will show first when the user begins to type into the Google Search box within each category (such as contacts, call log, e-mail, application). The score is typically based upon frequency of use per each category. An examiner can use this intelligence to determine most frequently contacted and used data within the device.
- **/data/com.google.android.googlequicksearchbox/cache/http/** Like all other HTTP folders within the cache, GUIDs are stored in this location. This folder stores cached data in the form of GUID named files. There are generally two related files with the same GUID differentiated by a *0* or *1*; *0*-ending files are the description of the associated file ending in number *1*. This information can be helpful for an examiner because voice searches, Internet searches, and typed searches using the Google Quick Search box are stored along with the data that was returned to the user (see Figure 15-11). The search, stored in the GUID with an appended *0*, contains date and time information for both the search and return of data along with the content type description. Additionally, a folder under /cache/ could also be cronet. Cronet is a network library and will, like the http folder, hold valuable cached user information.



```
HTTP/1.1 200 OK
status:304 acces
s-control-allow-
origin:* alt-svc
:quic=":443"; p=
"1"; ma=604800 a
lternate-protoco
l:443:quic,p=1 d
ate:Mon, 12 Oct
2015 11:01:18 GM
T server:fife ag
e:0 cache-contro
l:public, max-ag
e=86400, no-tran
sform content-di
sposition:inline
;filename="unnam
ed.jpg" content-
length:82096 con
tent-type:image/
jpeg etag:"v1" e
```

FIGURE 15-11 **The file in the search describes what was searched for with the associated result displayed.**

- **/dbdata/databases/com.google.android.googlequicksearchbox/qsb-history.db** This SQLite file is found in some Samsung Galaxy S series devices and contains the history of the searches used in the Google Quick Search box. This database has been deprecated, but the table history contains the query used and the date and time. Another database in the same directory, qsb-log.db, can be used to log web clicks used in a search. This information is obtainable within the clicklog table.
- **/data/com.google.android.gsf/databases/subscriberfeeds.db** This app is the Google Services Framework, and this SQLite database contains the Google accounts that are subscribed to, along with the user's e-mail address.
- **/data/com.google.android.gsf/databases/talk.db** This SQLite database contains the Google Talk settings such as account and contacts within the app. Google Talk is deprecated and is now Google Hangouts (see com.google.android.talk).
- **/data/com.google.android.street/cache/** This folder contains streetview images from a Google Maps instance. The images are all prefixed by *tile_* and then ordered by a three-digit number. The examiner can

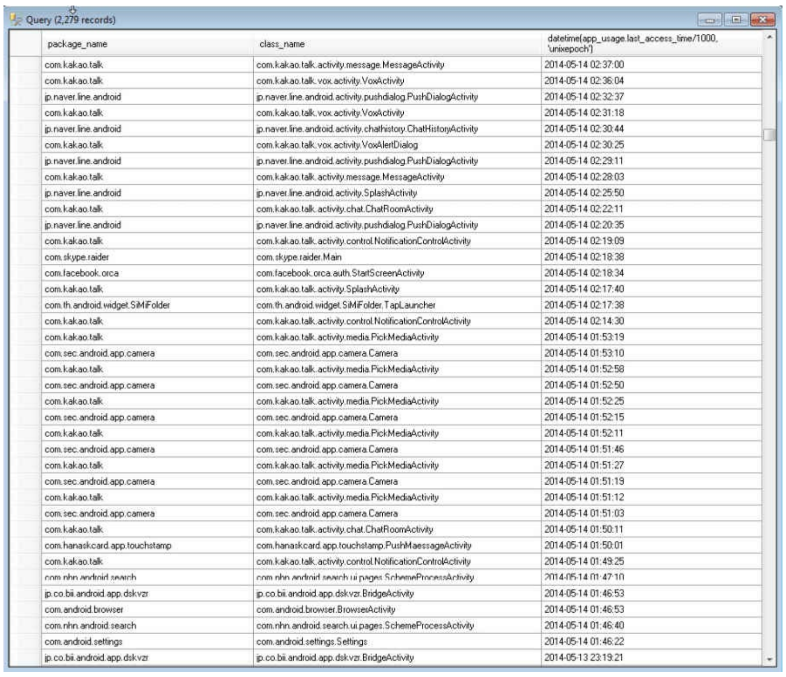
use these tiles to obtain images that the user of the device accessed while using the streeview function in Google Maps.

- **/data/com.google.android.talk/databases/message_store.db** This SQLite database contains the Google Hangouts messages within the messages table.
- **/data/com.google.android.talk/databases/babel.db** This SQLite database is the conversation database for active conversations, participant names, messages, and information about the Google Hangout event. There can be multiple babel.db databases, and each database name will be followed by an integer starting with *0* (such as babel0.db, babel1.db, babel3.db).
- **/data/com.google.android.talk/shared_prefs/accounts.xml** This Google Hangout XML file lists key information to the Hangout owner and preferences for the account.
- **/data/com.google.android.talk/shared_prefs/phone_verification.xml** This Google Hangout XML file lists the phone number associated with the account.
- **/data/com.google.android.youtube/databases/history.db** This SQLite database is used by the Google YouTube app to store searches for videos along with the date and time of each search.
- **/data/com.google.android.youtube/shared_prefs/youtube.xml** This XML file identifies the account of the YouTube user along with a history of downloads, last login, and other significant information of value to an examiner.

Samsung Android Artifacts

With Samsung continuing to be the leader in the global smart device market, as reported by the IDC (www.idc.com/prodserv/smartphone-market-share.jsp), an examiner should be familiar with the file system artifacts that are specific to these Android devices. Following are app locations within the /data folder that an examiner will see only within the Samsung device. Although the many preinstalled applications may annoy some customers, many of these apps are used by many people.

- **/data/com.samsung.android.providers.context/databases/ContextDB.db** This SQLite database can help an examiner determine whether an application was running or had been launched around the time of an incident. The app's internal services are exposed, showing the service that was launched as part of the app, to identify messages, notifications, and conversations. By using the app_usage table and fields package_name, class_name, and app_usage_last_access_time (in milliseconds), an examiner can gain interesting intelligence for a time line. A similar location for the device can also be found at /system/dmappmgr.db and the ApplicationControl table. In Figure 15-12, a parsed ContextDB.db file using the SQLQuery Builder application shows a considerable amount of information that can quickly be placed into a time line.


Query (2,279 records)

package_name	class_name	datetime(app_usage.last_access_time/1000, 'unixepoch')
com.kakao.talk	com.kakao.talk.activity.message.MessageActivity	2014-05-14 02:37:00
com.kakao.talk	com.kakao.talk.vox.activity.VoxActivity	2014-05-14 02:36:04
jp.naver.line.android	jp.naver.line.android.activity.pushdialog.PushDialogActivity	2014-05-14 02:32:37
com.kakao.talk	com.kakao.talk.vox.activity.VoxActivity	2014-05-14 02:31:18
jp.naver.line.android	jp.naver.line.android.activity.chathistory.ChatHistoryActivity	2014-05-14 02:30:44
com.kakao.talk	com.kakao.talk.vox.activity.VoxAlertDialog	2014-05-14 02:30:25
jp.naver.line.android	jp.naver.line.android.activity.pushdialog.PushDialogActivity	2014-05-14 02:29:11
com.kakao.talk	com.kakao.talk.activity.message.MessageActivity	2014-05-14 02:28:03
jp.naver.line.android	jp.naver.line.android.activity.SplashActivity	2014-05-14 02:25:50
com.kakao.talk	com.kakao.talk.activity.chat.ChatRoomActivity	2014-05-14 02:22:11
jp.naver.line.android	jp.naver.line.android.activity.pushdialog.PushDialogActivity	2014-05-14 02:20:35
com.kakao.talk	com.kakao.talk.activity.control.NotificationControlActivity	2014-05-14 02:19:09
com.skype.raider	com.skype.raider.Main	2014-05-14 02:18:38
com.facebook.orca	com.facebook.orca.auth.StartScreenActivity	2014-05-14 02:18:34
com.kakao.talk	com.kakao.talk.activity.SplashActivity	2014-05-14 02:17:40
com.th.android.widget.SiMiFolder	com.th.android.widget.SiMiFolder.TapLauncher	2014-05-14 02:17:38
com.kakao.talk	com.kakao.talk.activity.control.NotificationControlActivity	2014-05-14 02:14:30
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:53:19
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:53:10
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:52:58
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:52:50
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:52:25
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:52:15
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:52:11
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:51:46
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:51:27
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:51:19
com.kakao.talk	com.kakao.talk.activity.media.PickMediaActivity	2014-05-14 01:51:12
com.sec.android.app.camera	com.sec.android.app.camera.Camera	2014-05-14 01:51:03
com.kakao.talk	com.kakao.talk.activity.chat.ChatRoomActivity	2014-05-14 01:50:11
com.hanaskcard.app.touchstamp	com.hanaskcard.app.touchstamp.PushMaessageActivity	2014-05-14 01:50:01
com.kakao.talk	com.kakao.talk.activity.control.NotificationControlActivity	2014-05-14 01:49:25
com.nhn.android.search	com.nhn.android.search.ui.pages.SchemeProcessActivity	2014-05-14 01:47:10
jp.co.bii.android.app.dskvzr	jp.co.bii.android.app.dskvzr.BridgeActivity	2014-05-14 01:46:53
com.android.browser	com.android.browser.BrowserActivity	2014-05-14 01:46:53
com.nhn.android.search	com.nhn.android.search.ui.pages.SchemeProcessActivity	2014-05-14 01:46:40
com.android.settings	com.android.settings.Settings	2014-05-14 01:46:22
jp.co.bii.android.app.dskvzr	jp.co.bii.android.app.dskvzr.BridgeActivity	2014-05-13 23:19:21

FIGURE 15-12 **A parsed ContextDB.db file using the SQLQuery Builder application**

- **/data/com.samsung.android.app.galaxyfinder/SFinderHistory.db** This database is filled with searches across the Samsung device when the user uses the built-in Samsung Finder app from the settings. This file can be a gold mine, especially because the date added is a part of the database table history.
- **/data/com.samsung.clipboardsaveservice/f/** This folder stores saved clips from the clipboard (screenshots).
- **/data/com.sec.android.app.clockpackage/worldclock.db** This SQLite database contains all of the set clocks the user has added to the clock app. The examiner can use the worldclock table within this database to view any city that has been added to the wordclock view. New path and database name: /data/com.sec.android.app.clockpackage/db/worldclock.
- **/data/com.sec.android.app.clockpackage/alarm.db** This SQLite database contains all of the set alarms the user has added to the clock app. This table alarm contains the dates and times of alarm creation, the alert times, and the names of the alarms. New path and database name: /data/com.sec.android.app.clockpackage/db/.
- **/data/com.sec.android.im/databases/taConvDB.db** This SQLite database contains conversations and messages tables that can be used to recover the user's instant messages, as long as the user is using the Samsung instant messaging platform.
- **/data/com.sec.android.app.launcher/db/launcher.db** This SQLite database contains installed applications, and even applications that have been deleted in the appOrder table. Within the favorites table the application name and also icons are stored.
- **/data/com.sec.android.app.sbrowser/db/** This folder contains several databases rich with information. The SBrowser.db contains the BOOKMARKS table, which displays the URL of visited pages. The pintab.db contains the set bookmarks; typically, these are defaults added by Samsung. The weblogin.db

contains the username and web site information that belongs to the user within the fpauth table.

- **/data/com.sec.android.app.sbrowser/db/r/app_sbrowser/Default/databases/Databases.db** This file is the directory listing for WebKit data. Also in the Default directory is the folder Local Storage, which contains the .localstorage database files from web sites visited.
- **/data/com.sec.android.provider.logsprovider/databases/logs.db** This SQLite database is a gold mine of information. This file stores information that has been displayed on the Samsung device, such as notifications from e-mail, MMS, SMS, and calls. To detail the contents, the database file stores call logs, message snippets (SMS/MMS), and e-mail snippets along with their associated metadata. If a message has been deleted from the main SMS, call log, message, or e-mail, the data remains within this log. Experience has shown that even if the user has reset the messages within the device, data can still be recovered from logs.db. Table 15-6 lists some of the information available from this telltale Samsung database with logtype and type coming from the logs table in logs.db.

TABLE 15-6 **Logs.db from the Samsung Logging Function and Database Located at com.sec.android.provider.logsprovider**

Data	logtype	type	Description
Call history	100	1 = Incoming 2 = Outgoing	The call history in the database is verbose, containing the number, contact name, date, geolocation, and other valuable artifacts.
MMS	200	1 = Incoming 2 = Outgoing	The MMS in the database contains information such as telephone numbers, subject, geolocation data, and other valuable artifacts. The examiner can use the messageid to identify whether the message still exists in the mmssms.db.
SMS	300	1 = Incoming 2 = Outgoing	The SMS in the database contains information such as telephone numbers, subject, body, geolocation data, and other valuable artifacts. The examiner can use the messageid to identify whether the message still exists in the mmssms.db.
E-mail	400	1 = Incoming 2 = Outgoing	The e-mail notifications are dependent upon user settings for the built-in Android e-mail logged by this feature. The number of e-mails cached in this database can be much smaller than other artifacts. Information such as e-mail addresses, subject, body, geolocation data, and other valuable artifacts may be located in this table. The examiner can use the messageid to identify whether the message still exists in the EmailProvider.db.

- **/data/com.sec.android.provider.smemo/databases/pen_memo.db** This SQLite database stores data from the Memo app, including voice, pictures, and text. The information is located in the table PenMemo within several fields, including CreateDate, Date, Content, Text, Text_sub, Drawing, and Thumb.

- **/data/com.sec.android.provider.downloads/databases/sisodownloads.db** This SQLite database is comparable to the com.android.providers.downloads database. This is a good database for an examiner to investigate because media downloaded from web sites can be located here.
- **/data/com.sec.android.socialhub/databases/UniboxProvider.db** This SQLite database contains accounts and messages tables that can be used to match accounts for social media, instant messaging, text messaging, and e-mail notifications with the messages themselves. If the user has enabled this feature, this database can contain current and historical data.
- **/data/com.seven.Z7/databases/email.db** This SQLite database contains more data from the social hub addition to the Samsung line. It can store e-mail addresses, passwords, and e-mail content. Also, associated accounts and their instant message data can be stored here.

Third-Party Apps

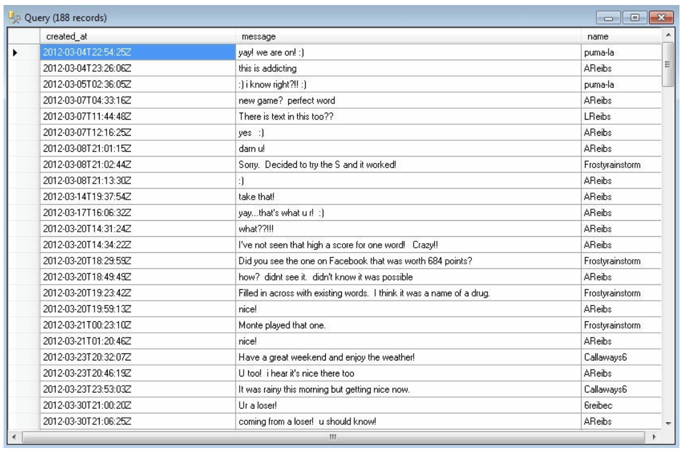
Unlike iOS devices, third-party applications are stored in the same /data folder along with all of the other stock, phone, and often carrier apps. The third-party apps use the domain name format previously outlined.

Note

There are more than 1.5 million Android apps available on the Google Play site, and documenting each and every app naming convention would take hundreds of pages. An examiner should understand that the layout of the folders within each app and the databases used to store the artifacts will be generally similar to what is outlined in Table 15-5. App designers can build their own app folder structures, and one of the difficult things for the automatic parsing of the apps within an Android, or any smart device, is that folder structure and database naming can change with an app update.

Third-party apps generally store the temporary data within the /cache folder, which can also include additional folders, such as webViewCacheChromium, which is now org.chromium.android_webview or cronet_cache on current devices (cronet is the chromium network stack). This folder contains data that was displayed to the user using the app. Files of interest will be prefixed with an *f_* and then a number. The *f* stands for files, and the files used by the web view in the app are listed. This naming convention is not observed in newer Androids, with files being stored in another directory under /cache typically named files or media. The /databases folder contains the databases for the app, along with another database called webview.db that stores form information and user information to be used within a web view in the app. This database can hold passwords, usernames, and other data important to an examiner. It should be noted that the webview.db has become Web Data and stored in the app_webview folder under the root of the app folder. Although Android development clearly encourages developers to use the correct controls of this class, WebViewDatabase, this database still should be queried. Apps also use the shared_prefs folder within the app structure, which contains XML files that outline the preferences for the app, including usernames and passwords. Because some apps do not store data on the mobile device for security concerns, the app simply uses a built-in browser to interface with a secure server, but often automated login information is located within the shared_prefs folder. This location should also be examined when conducting an examination of third-party apps.

Most Android apps use SQLite databases, and as outlined in Chapter 14, building a SQL query to support an application, along with recovering free pages and WAL data, should be conducted on any app important to the examination. Even in today's most popular games, users can chat with one another, often adding to an examiner's investigation, as shown in Figure 15-13.



Query (188 records)

created_at	message	name
2012-03-04T22:54:25Z	yay! we are on! :)	puma-la
2012-03-04T23:26:06Z	this is addicting	AReibs
2012-03-05T02:36:05Z	:) i know right?!! :)	puma-la
2012-03-07T04:33:16Z	new game? perfect word	AReibs
2012-03-07T11:44:48Z	There is text in this too??	LReibs
2012-03-07T12:16:25Z	yes :)	AReibs
2012-03-08T21:01:15Z	darn u!	AReibs
2012-03-08T21:02:44Z	Sorry. Decided to try the S and it worked!	Frostyrainstorm
2012-03-08T21:13:30Z	:)	AReibs
2012-03-14T19:37:54Z	take that!	AReibs
2012-03-17T16:06:32Z	yay...that's what u r! :)	AReibs
2012-03-20T14:31:24Z	what??!!!	AReibs
2012-03-20T14:34:22Z	I've not seen that high a score for one word! Crazy!!	AReibs
2012-03-20T18:29:59Z	Did you see the one on Facebook that was worth 684 points?	Frostyrainstorm
2012-03-20T18:49:49Z	how? didnt see it. didn't know it was possible	AReibs
2012-03-20T19:23:42Z	Filled in across with existing words. I think it was a name of a drug.	Frostyrainstorm
2012-03-20T19:59:13Z	nice!	AReibs
2012-03-21T00:23:10Z	Monte played that one.	Frostyrainstorm
2012-03-21T01:20:46Z	nice!	AReibs
2012-03-23T20:32:07Z	Have a great weekend and enjoy the weather!	Callaways6
2012-03-23T20:46:19Z	U too! i hear it's nice there too	AReibs
2012-03-23T23:53:03Z	It was rainy this morning but getting nice now.	Callaways6
2012-03-30T21:00:20Z	Ur a loser!	6reibec
2012-03-30T21:06:25Z	coming from a loser! u should know!	AReibs

FIGURE 15-13 **Here, /data/com.zynga.words/WordsFramework is queried for its internal messaging with more than 100 messages returned.**

File Interrogation

Every file mentioned in this chapter and any other file within the Android file system can be parsed to recover every bit of data within the particular file. The attack surface within the Android is limitless for an examiner with the right tools and, more importantly, the drive to succeed. The SQLite database queries along with Python scripting can enable an examiner to interrogate most of the files within the Android file system. With the primary files for storage being SQLite databases and the primary file type for settings and preferences being XML, the examiner already has a leg up, since the parsing of both of these file types was covered previously.

Android non-invasive or invasive physical recovery of the partitions also enables the examiner to carve the unallocated space on both the cache and userdata partitions. The unallocated space within the Android device can hold media files, SQLite databases, XML, HTML, deleted MMS media, and other settings that are no longer active within the file system. Using a forensic tool to carve the unallocated space for these particular file types should be attempted by the examiner prior to the conclusion of the examination. Hundreds and possibly thousands of additional files can be recovered. In Figure 15-14, hundreds of images deleted from MMS along with HTML files were recovered using data carving techniques with Oxygen Forensic Detective.

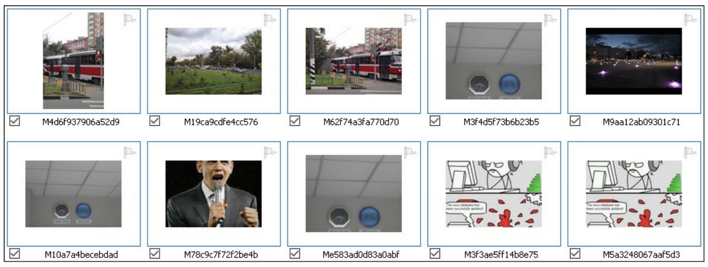


FIGURE 15-14 **Many forensic tools support the data carving functionality.**

SQLite, XML files, and any other files within the file system of an Android should be examined. Some examples of parsing of files using Python parsing are covered next, with files that have been recovered from an Android device. The creation of SQL queries for Android databases was covered in Chapter 14.

Scripts

Often an examiner will need to locate the operating system information from an Android device. This information is often difficult to find within the different files of the file system. If the examiner's mobile forensic tool does not recover this information automatically, the examiner can create a simple script to run against the file located in the userdata partition at /log/recovery_log.txt to search and list only the operating system information:

```
import re
from sys import argv
script,f = argv
fstuff = open(f,'r')

for line in fstuff:
    if re.match("(.*)ro.build(.*)", line):
        print (line,)
```

An abbreviated output from running the script against recovery_log.txt displays the required operating system information:

```
ro.build.id=JZO54K
ro.build.display.id=JZO54K.E210SKSJLL5
ro.build.version.incremental=E210SKSJLL5
ro.build.version.sdk=16
ro.build.version.codename=REL
ro.build.version.release=4.1.2
ro.build.date=Tue Dec 11 10:47:12 KST 2012
ro.build.date.utc=1355190432
ro.build.type=user
ro.build.user=se.infra
ro.build.host=SEP-129
```

Because an Android device holds many XML files, running a script to parse these XML files to make them more readable is often a necessity. This script will output the XML into a form that can be easily read within an examiner's report:

```
import xml.etree.cElementTree as ET
from sys import argv

script, xmlfile = argv
tree = ET.ElementTree(file=xmlfile)
tree.getroot()
root = tree.getroot()
for rootchild in root:
    print (rootchild.attrib,rootchild.text)
```

The output of this script for a common Android file for the built-in browser indicates last search times and the web search engine located at /data/com.google.android.googlequicksearchbox/SearchSettings.xml. The script could be further refined to translate the date and time information as well:

```
{'name': 'psuggest_available', 'value': 'false'} None
{'name': 'search_domain'} .google.com
{'name': 'next_voice_search_hint', 'value': '3'} None
{'name': 'search_history', 'value': 'true'} None
{'name': 'search_domain_apply_time', 'value': '1387471182477'} None
{'name': 'search_domain_country_code'} US
{'name': 'num_visible_suggestion_slots_2', 'value': '4'} None
{'name': 'voice_search_version', 'value': '211'} None
{'name': 'num_visible_suggestion_slots_1', 'value': '4'} None
{'name': 'first_voice_search_hint_time', 'value': '1358826686102'} None
```

Applying scripting to an Android examination, along with building SQL queries, can have an enormous impact on the outcome of an investigation. By using these types of techniques, an examiner will be sure to uncover the maximum amount of data from these treasure troves.

Android App Files and Malware

Android apps are created and installed to a device in the form of an APK file. APK files are compressed binary files containing DEX files, resource files (.arsc), uncompiled resources, and the AndroidManifest.xml file. These files and folders make up the app much like a Windows MSI or Mac .dmg file. The APK is installed to the device and stored in the /app folder within the Android file system for non-system apps; system apps are stored in the system partition. If an app is installed to the sdcard partition, the contents are generally encrypted and stored in a /sdcard/.android secure directory. The filenames have an *asec* extension instead of *apk*. If the device has decrypted these apps from the SD card, they will be stored at /data/app-asec and can be examined. The /dalvik-cache folder stores the installed app code for execution within the Dalvik VM instance when the app is called to run.

The important aspect of these two areas for an examiner concerns malware. Malware on an Android device is far more likely to occur than on any other smart device on the planet. Kaspersky Labs reports that more than 97 percent of mobile malware they detected was from the Android platform. There are various reasons for the malware infiltration on the mobile device, but it typically comes down to monetary gain for the group infecting devices. From SMS for a service to ransomware, an Android device can be susceptible. An examiner must be prepared for malware being involved on an Android device and understand the consequences.

Computer forensic examiners are trained to scan computer digital repositories for known malicious applications, viruses, keyloggers, and other similar files to disprove or prove malware added to, deleted from, or modified the data being examined. Mobile device examiners do not show this same diligence, however, which could possibly discredit any data that might be recovered from an Android device. Android malware is already outpacing Windows for malware infections, and what took Windows 14 years to achieve with infections, Android has overtaken in only 3. An examiner should be ready to complete an examination of malware on a mobile device, particularly an Android device.

Cellebrite offers a malware scanner within its mobile forensic software that uses Bitdefender to locate malware signatures. An examiner should realize, however, that a signature-based examination is the lowest form of malware analysis, because a signature is based upon a known hash. This known hash is placed into a blacklist and uploaded to a signature repository to be used against the app. Because an app can change daily after being morphed into an illegitimate form of a legitimate app, or because a zero-day app can drop at any moment, using signature-based analysis could miss valid malware. (This is not to say that the examiner should not perform an analysis of an Android device with a signature-based tool. Anything is better than nothing, and as long as the signatures are current, the analysis would be comparable to a computer's virus/malware scanner.) Malware companies base the signatures upon the DEX file within the /davlik-cache folder. However, there are instances where malware has injected or patched a legitimate app's DEX file within the davlik-cache, thus evading malware signature-based analysis.

Malware installed on an Android device can cause problems for an examiner, most importantly during the court testimony stage. Malware software can be used to gather information, disrupt service, connect to premium services, and even enable remote access to a device. Malware to disrupt can cause the device to be inoperable, cause redirects to web sites, install ransomware, and result in other problems. The following list shows several types of malware and how they work:

- **Ransomware** After locking the device, the user is shown a screen indicating that illegal material has been found on the device by the authorities and it will not be unlocked until a fine has been paid.
- **Premium services** These apps redirect phone and messaging services to a pay-per-use service to add phone charges. They collect money for every message sent and charge it to the carrier and then to the user's account.
- **Spyware** Spyware can turn on and monitor microphones, cameras, messages, keystrokes, and other data from a mobile device. This type of malware is often sold as a legitimate "monitoring" app. The information gathered is sent from the device to a server under control of the criminal.
- **Trojan** This was the most prevalent type of malware at one time, with designers injecting malicious code into legitimate apps. Google Play at the time failed to identify this malware, and many unsuspecting users downloaded what they thought was the original app. For example, Svpeng, a Russian Trojan, affected Androids by asking for credit card information each time the user visited Google Play. The data was then

sent directly to a server, and credit card details were sold. The Google Play store now uses “Bouncer” to scan the app for malicious code or changes—eliminating some of the risk.

- **Data miner** This app steals data from the device, including contacts and phone information. Often, the malware will steal contacts and extort money from the user—saying an image they have will be sent to all of the user’s contacts. Also, phone details such as IMEIs are stolen to sell to others to clone onto blacklisted handsets.
- **Adware** Messages from apps that should not be advising the user that “Your phone is slow,” “You have been infected,” or “Your device is being tracked,” is a sign of adware, a prevalent malware type.
- **Downloader** This app directs users to additional downloads that appear to be content—such as PDF, JPG, and so on—but are additional malware apps, often Trojans.
- **Browser hijacker** These apps use WebView and not the Android internal browser to redirect and incur browser drive-by hacks, often infecting the device without the user even doing anything on the web page.
- **Botnet** Like computer systems, mobile devices are subject to malware packages that turn the device into a zombie. The command and control (C&C) servers are connected, and the device becomes a part of a network of devices that often distribute more malware.

Malware actions might include modifying the permissions for an Internet game to read the contacts, SMS, or other phone details to transmit that data to another server, or using the details to send messages and even clone the serial number to another device.

Analysis Levels

Three levels of malware analyses—permissions, static, and dynamic—are used to root out malware and determine the legitimacy of the app and its intentions. Permissions is the first level of malware detection and analysis. Static analysis is the second level of malware detection, and dynamic analysis is the third and top-level analysis type.

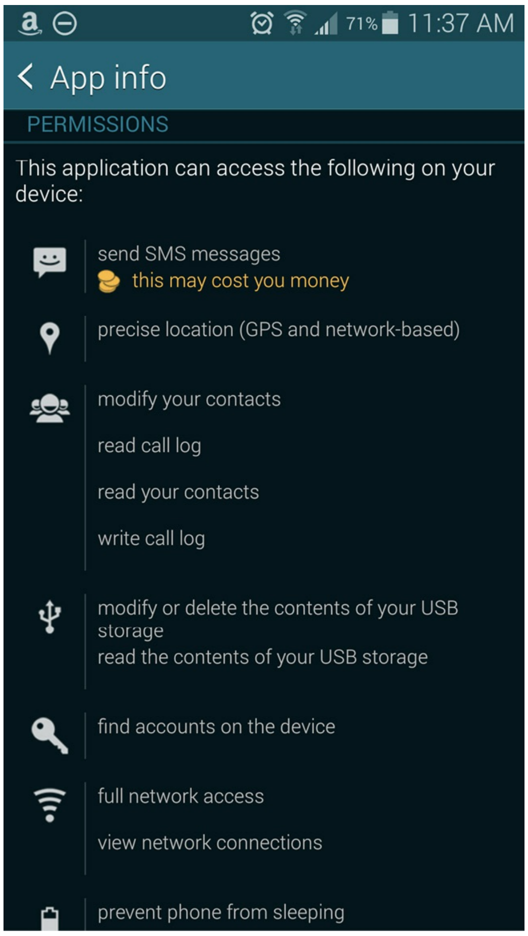
Permissions Analysis

Android apps list permissions (Figure 15-15) that must be granted to the app to enable it to function as the developer intended. The user typically sees these permissions prior to installing the APK file on the device, and they can also be viewed by navigating to app settings. On devices running Android 6 and later, the app will not ask for permission on install. The app must ask the user to grant permissions at runtime. The user can accept or deny each permission one at a time if needed. Permissions for all APK files can also be viewed on the Google Play Store. The permissions granted to an Android app can assist the examiner as to the app’s intentions. Table 15-7 lists Android permissions and their descriptions, including ways in which malware can affect the permission. Permissions are separated into NORMAL, SIGNATURE, and DANGEROUS. Outlined in the table are those that could be dangerous and should be investigated.

TABLE 15-7 **Android Dangerous Permissions and Descriptions (https://developer.android.com/guide/topics/permissions/overview)**

Permission	Description
Make phone calls	Moderate to high importance. Used by VoIP apps. Malware could dial paid services to route calls.
Modify contacts, read call log, read contacts, write call log	Moderate to high importance. Many legitimate apps use this permission. Malware could capture device contacts and other phone call–related functions.
Send SMS messages	Moderate to high importance. Used by many apps to post results, invite others to games, and more. Malware could use paid services to send messages.
Storage—modify/delete SD card contents	Investigate this permission. Many apps must be granted this permission for storage, and there are many legitimate uses for this permission. Allow an app to read, write, and delete anything stored on the device media card. Control is granted to the entire SD card and generally not just the area the app controls.
Modify calendar data, write calendar data, read calendar data	Moderate to high importance. Malware could capture the device calendar information, and if the app is not used for calendar functions, this should be investigated.
Phone call—read phone state and identity	Moderate to high importance. Many apps use this permission to function if another phone function is needed. The IMEI, IMSI, and a 64-bit unique ID that Google provides for your phone is accessible using the same permission.
Precise location—network and GPS	Can be high importance if the app is not one of the previously listed types and location is not needed for the app to function correctly. Many apps use this function for restaurant directories, location finders, mapping applications, health apps, and many others. Malware could use permission to track the device.
Modify global system settings	Allows an app to interact with and change any settings within the General | Settings area of the device.
Automatically start at boot	Allows the app to start with each boot. Could be important if this is not a system type app and is labeled as a game. Understand the app type and need to help determine the intention.

Restart other applications	Allows an app to control another app by stopping the app and then allowing it to be restarted.
Retrieve running applications	Allows an app to determine the running apps. Examine the app to see if it interacts with other apps on the device and to determine whether this permission is legitimate.
Set preferred applications	Allows the app to set the preferred apps to be used for MMS, SMS, e-mail, and other communication apps. Can be used by malware to hijack a function, similar to a toolbar injection with a computer browser.
Take pictures	Used by picture-taking apps. Malware could use this function to take pictures secretly with the device camera. If the app is a game with no function other than playing a game, this could be of importance.
Discover known accounts	Allows the app to discover the accounts on the device. Does not provide credential information, however.


71% 11:37 AM

App info

PERMISSIONS

This application can access the following on your device:

send SMS messages
this may cost you money

precise location (GPS and network-based)

modify your contacts

read call log

read your contacts

write call log

modify or delete the contents of your USB storage
read the contents of your USB storage

find accounts on the device

full network access

view network connections

prevent phone from sleeping

FIGURE 15-15 App permissions

Tip

Permissions for all apps on a device can also be examined within the /system/packages.list if this file can be accessed.

An app cannot access parts of the mobile device or provider apps if permission has not been granted. Most Android users do not understand the implications of accepting all permissions requested by the app and choose to accept all—granting dangerous access to malicious apps. Obviously this is a security concern for the user, but how about the examiner? As an examiner, these permissions should be investigated on apps that are primary evidence within the investigation. An examiner should unequivocally explain the intentions of that app, the expected outcome of using the app, and the evidence it produced. This perceptive investigation of the app's behavior will solidify the stance of the investigation and evidence recovered.

Static Analysis

Static analysis is a more in-depth analysis of any app that the examiner believes to be malware, based upon a permission or signature alert. A static analysis enables an examiner to look into the app code in a controlled environment—to look for "hooks" that might indicate malware, identify calls or posts to a third-party server, or recognize other anomalies within the app. Several tools are available to assist an examiner in the static analysis of Android apps; this type of examination is not automated and can be time consuming, but it is often a necessity.

Note

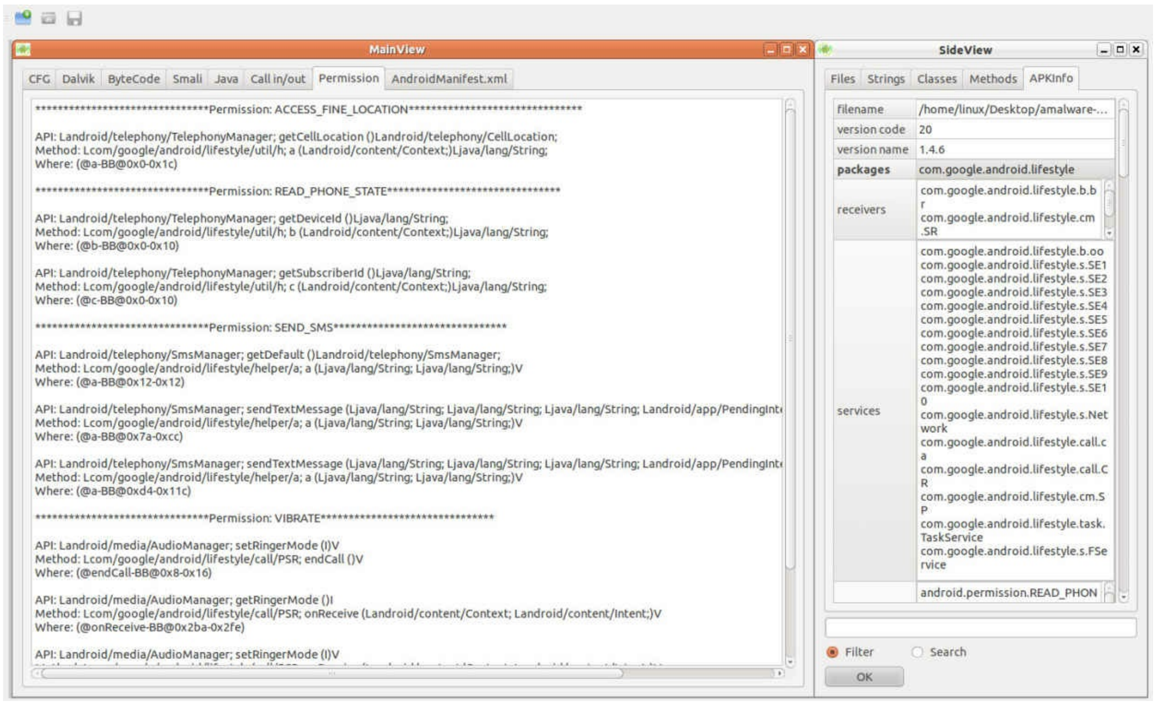
In 2012, as part of course development for a malware analysis course, I created content and a virtual machine to enable students and attendees to analyze actual malware, both statically and dynamically. This method is still viable and utilized. The virtual machine was created using Linux Ubuntu as the OS with Eclipse, Wireshark, Androguard, and APKInspector installed. Eclipse was used for live active analysis using logcat and threading; APKInspector was used for the disassembly and analysis of the APK and DEX code; Androguard was used for disassembly of APK, ODEX, and DEX files; and Wireshark was used for active communication protocols. The benefit of using these tools was easy: they all are open source and freeware, and they allow for the investigation of malware within a virtual sandbox—the Ubuntu VM. The course introduction was given to a group at a High Tech Crime Investigators Association conference, which spawned several other virtual appliances for this type of analysis, using the same setup and tools.

For static analysis, an examiner can use APKInspector, an appliance built with Python, to look for values that would be typical for a malicious app, but some of these permissions can be found in legitimate apps, too. The total number of hooks the examiner uncovers will determine the threat score of the identified app. Table 15-8 represents some of the more common requests found in malicious apps. Using APKInspector, the examiner can search for these terms to determine whether the app is using processes and commands that should not be a part of the app's functioning.

TABLE 15-8 **Common Commands Observed Within the Codebase of Malicious Applications**

Permission	Description
chmod	Change mode: This UNIX command changes permissions for files or directories. An app that changes a file directory from read-only to read-write and executes privileges using chmod 777 could indicate a malicious app. Search for chmod within the app code.
chown	Change owner: This UNIX command changes the ownership of a file. If root is "owned," the device can be compromised.
root	In conjunction with other terms, any app requiring root privileges will undoubtedly be malicious.
rmmod	This process attempts to remove a module from the kernel and should be interpreted as malicious when found.
link	This command/process attempts to link an existing file with another file, often creating a new file with additional content; it can enable data to be injected into another file.
unlink	This deletes filenames outside of the app.
symlink	Symbolic link: Directs a link to a legitimate app to a newly installed malicious app.
open	Common in many apps, this should be evaluated by the examiner to determine what files are being opened.
ifconfig	Configures or changes configuration of a network resource. This command can redirect output to another network address.
mount	Can be used to mount a file system as read-write, allowing access to a file system that should always be read-only.
HttpPost HttpRequest Http	HTTP commands used to post, get, and respond to information sent to or from a third-party server. Should be closely monitored to find out what data is posted and received and to where.
Permission-based terms	These terms can be standard for many apps, but they can be used when searching an app that may not identify these permissions in the AndroidManifest.XML: getDeviceId, getSimSerialNumber, getSubscriberId, sendToHost, sendToPort.

Before the examiner launches APKInspector, Python must be installed within the Linux instance. Then, from the Ubuntu Linux terminal, the examiner types **python StartQT.py** from within the APKInspector directory. The APKInspector UI will open, where the examiner can select to open an APK file by clicking the open folder (blue folder with a plus sign). Using the interface SideView in conjunction with the MainView (Figure 15-16), an examiner can view permissions, code, and calls made in and out, along with the DEX-translated code information. The examiner can view the actual code to determine exactly how the app is functioning, any classes used, and any malicious code calls if they have been identified. Sometimes the examiner can find information that can be used to track the malware developer when comments (`// This is a comment`) are included within the code. APKInspector is not an automated tool, but it can help an examiner conduct a thorough static analysis of Android apps in an attempt to identify known malware hooks. However, a dynamic analysis must also be run to understand the malware and any malicious intent completely.



```
*******************************Permission: ACCESS_FINE_LOCATION*******************************

API: Landroid/telephony/TelephonyManager; getCellLocation ()Landroid/telephony/CellLocation;
Method: Lcom/google/android/lifestyle/util/h; a (Landroid/content/Context;)Ljava/lang/String;
Where: (@a-BB@0x0-0x1c)

*******************************Permission: READ_PHONE_STATE*******************************

API: Landroid/telephony/TelephonyManager; getDeviceId ()Ljava/lang/String;
Method: Lcom/google/android/lifestyle/util/h; b (Landroid/content/Context;)Ljava/lang/String;
Where: (@b-BB@0x0-0x10)

API: Landroid/telephony/TelephonyManager; getSubscriberId ()Ljava/lang/String;
Method: Lcom/google/android/lifestyle/util/h; c (Landroid/content/Context;)Ljava/lang/String;
Where: (@c-BB@0x0-0x10)

*******************************Permission: SEND_SMS*******************************

API: Landroid/telephony/SmsManager; getDefault ()Landroid/telephony/SmsManager;
Method: Lcom/google/android/lifestyle/helper/a; a (Ljava/lang/String; Ljava/lang/String;)V
Where: (@a-BB@0x12-0x12)

API: Landroid/telephony/SmsManager; sendTextMessage (Ljava/lang/String; Ljava/lang/String; Ljava/lang/String; Landroid/app/PendingInte
Method: Lcom/google/android/lifestyle/helper/a; a (Ljava/lang/String; Ljava/lang/String;)V
Where: (@a-BB@0x7a-0xcc)

API: Landroid/telephony/SmsManager; sendTextMessage (Ljava/lang/String; Ljava/lang/String; Ljava/lang/String; Landroid/app/PendingInte
Method: Lcom/google/android/lifestyle/helper/a; a (Ljava/lang/String; Ljava/lang/String;)V
Where: (@a-BB@0xd4-0x11c)

*******************************Permission: VIBRATE*******************************

API: Landroid/media/AudioManager; setRingerMode (I)V
Method: Lcom/google/android/lifestyle/call/PSR; endCall ()V
Where: (@endCall-BB@0x8-0x16)

API: Landroid/media/AudioManager; getRingerMode ()I
Method: Lcom/google/android/lifestyle/call/PSR; onReceive (Landroid/content/Context; Landroid/content/Intent;)V
Where: (@onReceive-BB@0x2ba-0x2fe)

API: Landroid/media/AudioManager; setRingerMode (I)V
```

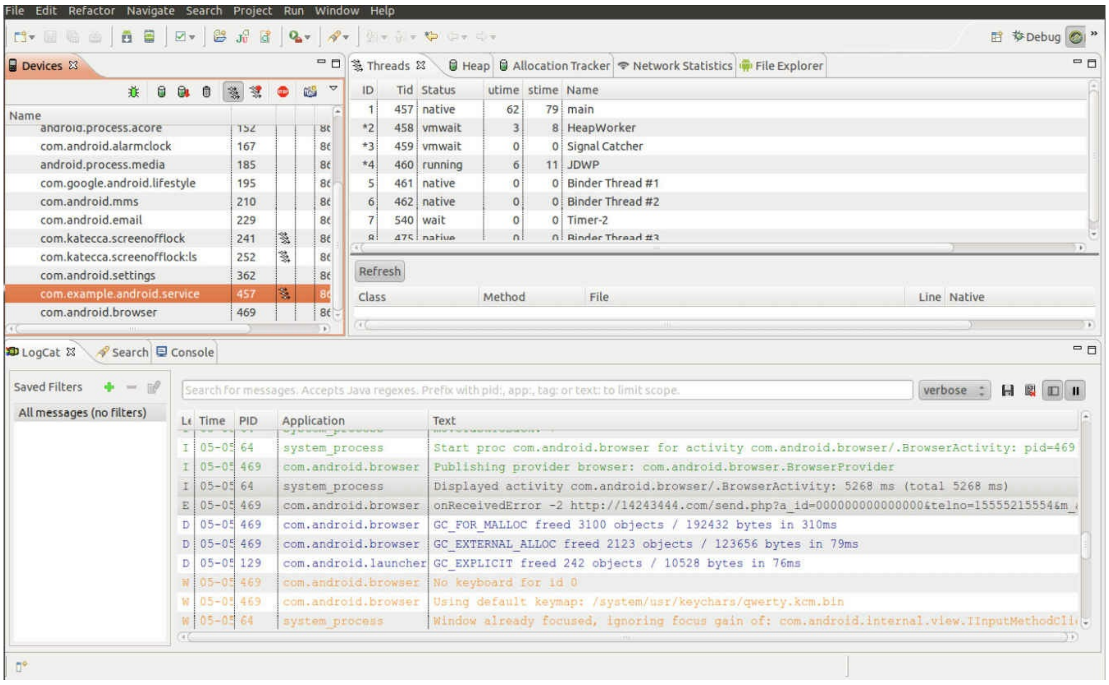
filename	/home/linux/Desktop/amalware-...
version code	20
version name	1.4.6
packages	com.google.android.lifestyle
receivers	com.google.android.lifestyle.b.b r com.google.android.lifestyle.cm .SR
services	com.google.android.lifestyle.b.oo com.google.android.lifestyle.s.SE1 com.google.android.lifestyle.s.SE2 com.google.android.lifestyle.s.SE3 com.google.android.lifestyle.s.SE4 com.google.android.lifestyle.s.SE5 com.google.android.lifestyle.s.SE6 com.google.android.lifestyle.s.SE7 com.google.android.lifestyle.s.SE8 com.google.android.lifestyle.s.SE9 com.google.android.lifestyle.s.SE1 0 com.google.android.lifestyle.s.Net work com.google.android.lifestyle.call.c a com.google.android.lifestyle.call.C R com.google.android.lifestyle.cm.S P com.google.android.lifestyle.task. TaskService com.google.android.lifestyle.s.FSe rvice
	android.permission.READ_PHON

FIGURE 15-16 **APKInspector can be a good static analysis tool that shows permissions, calls, and even code.**

Dynamic Analysis

Dynamic analysis of an Android app can also be accomplished within a Linux Ubuntu virtual machine using an Android emulator and Wireshark, plus an emulator instance with Eclipse. Conducting a dynamic analysis of an Android app enables the examiner to monitor the app as it is running within the Android emulator, to watch the way the app works and monitor the output and input when the app is running certain processes.

Using Eclipse, the examiner can select the app within the interface and monitor the threads, as well as use logcat to monitor the device and app functions. Eclipse can also be used to monitor and debug the app APK while running in the instance of Eclipse, if needed. This can help an examiner monitor the suspected APK on an emulated device and debug/monitor the APK files as well. This information can show HTTP attempts and system communications. Eclipse can also be used to monitor an emulator that contains malicious APK files (Figure 15-17). Using threads along with logcat, the examiner can view occurring changes in the system in real time. A malicious browser can redirect the IMEI and phone number to a URL.


File Edit Refactor Navigate Search Project Run Window Help
Debug
Devices
Threads
Heap
Allocation Tracker
Network Statistics
File Explorer
Refresh
LogCat
Search
Console
Saved Filters
All messages (no filters)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

Name				
android.process.acore	152			8(
com.android.alarmclock	167			8(
android.process.media	185			8(
com.google.android.lifestyle	195			8(
com.android.mms	210			8(
com.android.email	229			8(
com.katecca.screenofflock	241			8(
com.katecca.screenofflock:ls	252			8(
com.android.settings	362			8(
com.example.android.service	457			8(
com.android.browser	469			8(

ID	Tid	Status	utime	stime	Name
1	457	native	62	79	main
*2	458	vmwait	3	8	HeapWorker
*3	459	vmwait	0	0	Signal Catcher
*4	460	running	6	11	JDWP
5	461	native	0	0	Binder Thread #1
6	462	native	0	0	Binder Thread #2
7	540	wait	0	0	Timer-2
8	475	native	0	0	Binder Thread #3

Class	Method	File	Line	Native

```
Le Time  PID  Application            Text
I  05-05 64   system_process         Start proc com.android.browser for activity com.android.browser/.BrowserActivity: pid=469
I  05-05 469  com.android.browser    Publishing provider browser: com.android.browser.BrowserProvider
I  05-05 64   system_process         Displayed activity com.android.browser/.BrowserActivity: 5268 ms (total 5268 ms)
E  05-05 469  com.android.browser    onReceivedError -2 http://14243444.com/send.php?a_id=000000000000000&telno=15555215554&m_
D  05-05 469  com.android.browser    GC_FOR_MALLOC freed 3100 objects / 192432 bytes in 310ms
D  05-05 469  com.android.browser    GC_EXTERNAL_ALLOC freed 2123 objects / 123656 bytes in 79ms
D  05-05 129  com.android.launcher   GC_EXPLICIT freed 242 objects / 10528 bytes in 76ms
W  05-05 469  com.android.browser    No keyboard for id 0
W  05-05 469  com.android.browser    Using default keymap: /system/usr/keychars/qwerty.kcm.bin
W  05-05 64   system_process         Window already focused, ignoring focus gain of: com.android.internal.view.IInputMethodCli
```

FIGURE 15-17 **Monitoring malicious files in Eclipse**

Before the examiner launches Eclipse, an instance of the Android emulator with the suspected Android malware must be running. An instance can be started from the Ubuntu VM from the terminal by typing **emulator <Android Virtual Device (AVD)>** or from the Eclipse instance, as long as the Android ADT plug-in has been installed. If Eclipse and ADT are used, the emulator can be launched from the toolbar. Once launched, the emulated device will be visible within the Eclipse instance. The examiner can launch the suspected malware app on the emulated device and within Eclipse select the logcat tab. This will show the communication with the device, the app, and any other system communication. The examiner can view any outside requests, calls, and network attempts and save this information to a text file to be included in a report if needed.

Tip

Be sure to set the Windows | Preferences | Android | LogCat | Maximum Number Of LogCat Messages In Buffer to 0 to capture an unlimited number of lines, and then restart Eclipse. If this is not set, the default is only 5000 lines.

Data from Wireshark can also be analyzed to determine whether any data was sent or received from an outside source using TCP. If information is transferred or received, the examiner can determine whether this data is reasonable and whether the data is being transmitted or received from the correct server for the app or is going to a third-party location not specified by the app documentation. If the latter is the case, the examiner should analyze exposure to the device along with any received information from the third-party server. This analysis can often refute allegations that an app downloaded material automatically, without the user's knowledge, or that the app installed another unknown app that caused the evidence to appear on the device. This analysis can also help to determine a security breach and what type of data was lost.

Note

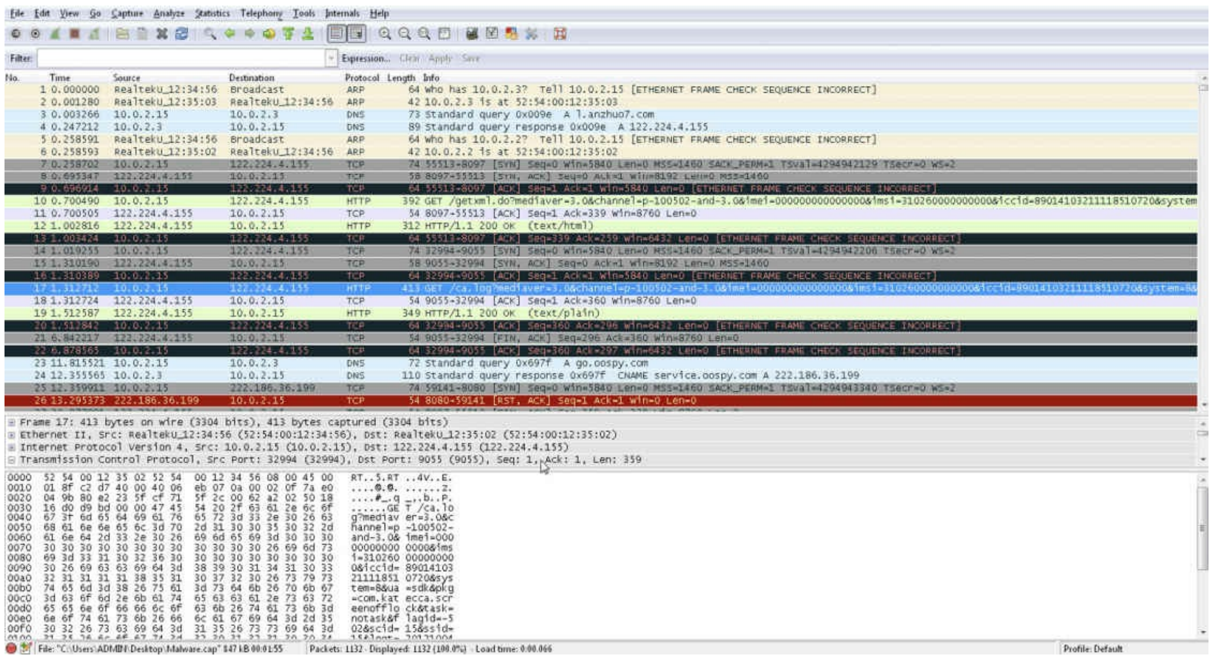
Conducting a dynamic analysis using Wireshark is a good alternative. This is a basic introduction to its use; other resources are dedicated to using Wireshark with advanced methods that are not covered in this section. An examiner should at least comprehend that there are methods for the dynamic analysis of Android apps using Wireshark.

Before the examiner uses Wireshark, he or she should create a TCP dump file when launching an emulator instance. This dump file will contain the recorded network traffic between the emulated device and the outside network. While running a suspected app, the examiner can observe any anomalies with the app and its communication with the network. These anomalies can include sending data, receiving data, and connecting to unknown IP addresses and servers. The examiner can run the following command from the terminal within the Linux virtual machine to start an instance of Wireshark's TCP recording on an instance of the Android emulator:

```
> emulator -tcpdump <name of capture file .cap> <Android Virtual Device (AVD)>
```

The app or apps can then be launched and the information can be examined to determine whether TCP/HTTP traffic has malicious intent. A capture file is then created, and this file will be imported into Wireshark and analyzed for any TCP or HTTP traffic.

Using advanced tools to combat the allegations of malware within an Android examination can be complicated, as shown in Figure 15-18, but an examiner must be prepared to conduct this type of analysis should these allegations be raised.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time       Source             Destination        Protocol Length Info
  1 0.000000   Realteku_12:34:56  Broadcast          ARP       64 Who has 10.0.2.3?  Tell 10.0.2.15 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
  2 0.001280   Realteku_12:35:03  Realteku_12:34:56  ARP       42 10.0.2.3 is at 52:54:00:12:35:03
  3 0.003266   10.0.2.15          10.0.2.3           DNS       73 Standard query 0x009e  A l.anzhuo7.com
  4 0.247212   10.0.2.3           10.0.2.15          DNS       89 Standard query response 0x009e  A 122.224.4.155
  5 0.258591   Realteku_12:34:56  Broadcast          ARP       64 Who has 10.0.2.2?  Tell 10.0.2.15 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
  6 0.258593   Realteku_12:35:02  Realteku_12:34:56  ARP       42 10.0.2.2 is at 52:54:00:12:35:02
  7 0.258702   10.0.2.15          122.224.4.155      TCP       74 55513→8097 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294942129 TSecr=0 WS=2
  8 0.695347   122.224.4.155      10.0.2.15          TCP       58 8097→55513 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
  9 0.696914   10.0.2.15          122.224.4.155      TCP       64 55513→8097 [ACK] Seq=1 Ack=1 Win=5840 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 10 0.700490   10.0.2.15          122.224.4.155      HTTP     392 GET /getxml.do?mediaver=3.0&channel=p-100502-and-3.0&imei=000000000000000&imsi=310260000000000&iccid=89014103211118510720&system
 11 0.700505   122.224.4.155      10.0.2.15          TCP       54 8097→55513 [ACK] Seq=1 Ack=339 Win=8760 Len=0
 12 1.002816   122.224.4.155      10.0.2.15          HTTP     312 HTTP/1.1 200 OK  (text/html)
 13 1.003424   10.0.2.15          122.224.4.155      TCP       64 55513→8097 [ACK] Seq=339 Ack=259 Win=6432 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 14 1.019255   10.0.2.15          122.224.4.155      TCP       74 32994→9055 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294942206 TSecr=0 WS=2
 15 1.310190   122.224.4.155      10.0.2.15          TCP       58 9055→32994 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
 16 1.310389   10.0.2.15          122.224.4.155      TCP       64 32994→9055 [ACK] Seq=1 Ack=1 Win=5840 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 17 1.312712   10.0.2.15          122.224.4.155      HTTP     413 GET /ca.log?mediaver=3.0&channel=p-100502-and-3.0&imei=000000000000000&imsi=310260000000000&iccid=89014103211118510720&system=8&
 18 1.312724   122.224.4.155      10.0.2.15          TCP       54 9055→32994 [ACK] Seq=1 Ack=360 Win=8760 Len=0
 19 1.512587   122.224.4.155      10.0.2.15          HTTP     349 HTTP/1.1 200 OK  (text/plain)
 20 1.512842   10.0.2.15          122.224.4.155      TCP       64 32994→9055 [ACK] Seq=360 Ack=296 Win=6432 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 21 6.842217   122.224.4.155      10.0.2.15          TCP       54 9055→32994 [FIN, ACK] Seq=296 Ack=360 Win=8760 Len=0
 22 6.878565   10.0.2.15          122.224.4.155      TCP       64 32994→9055 [ACK] Seq=360 Ack=297 Win=6432 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
 23 11.815521  10.0.2.15          10.0.2.3           DNS       72 Standard query 0x697f  A go.oospy.com
 24 12.355565  10.0.2.3           10.0.2.15          DNS      110 Standard query response 0x697f  CNAME service.oospy.com A 222.186.36.199
 25 12.359911  10.0.2.15          222.186.36.199     TCP       74 59141→8080 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294943340 TSecr=0 WS=2
 26 13.295373  222.186.36.199     10.0.2.15          TCP       54 8080→59141 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

⊞ Frame 17: 413 bytes on wire (3304 bits), 413 bytes captured (3304 bits)
⊞ Ethernet II, Src: Realteku_12:34:56 (52:54:00:12:34:56), Dst: Realteku_12:35:02 (52:54:00:12:35:02)
⊞ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 122.224.4.155 (122.224.4.155)
⊟ Transmission Control Protocol, Src Port: 32994 (32994), Dst Port: 9055 (9055), Seq: 1, Ack: 1, Len: 359

0000  52 54 00 12 35 02 52 54  00 12 34 56 08 00 45 00   RT..5.RT ..4V..E.
0010  01 8f c2 d7 40 00 40 06  eb 07 0a 00 02 0f 7a e0   ....@.@. ......z.
0020  04 9b 80 e2 23 5f cf 71  5f 2c 00 62 a2 02 50 18   ....#_.q _,.b..P.
0030  16 d0 d9 bd 00 00 47 45  54 20 2f 63 61 2e 6c 6f   ......GE T /ca.lo
0040  67 3f 6d 65 64 69 61 76  65 72 3d 33 2e 30 26 63   g?mediav er=3.0&c
0050  68 61 6e 6e 65 6c 3d 70  2d 31 30 30 35 30 32 2d   hannel=p -100502-
0060  61 6e 64 2d 33 2e 30 26  69 6d 65 69 3d 30 30 30   and-3.0& imei=000
0070  30 30 30 30 30 30 30 30  30 30 30 30 26 69 6d 73   00000000 0000&ims
0080  69 3d 33 31 30 32 36 30  30 30 30 30 30 30 30 30   i=310260 00000000
0090  30 26 69 63 63 69 64 3d  38 39 30 31 34 31 30 33   0&iccid= 89014103
00a0  32 31 31 31 31 38 35 31  30 37 32 30 26 73 79 73   21111851 0720&sys
00b0  74 65 6d 3d 38 26 75 61  3d 73 64 6b 26 70 6b 67   tem=8&ua =sdk&pkg
00c0  3d 63 6f 6d 2e 6b 61 74  65 63 63 61 2e 73 63 72   =com.kat ecca.scr
00d0  65 65 6e 6f 66 66 6c 6f  63 6b 26 74 61 73 6b 3d   eenofflo ck&task=
00e0  6e 6f 74 61 73 6b 26 66  6c 61 67 69 64 3d 2d 35   notask&f lagid=-5
00f0  30 32 26 73 63 69 64 3d  31 35 26 73 73 69 64 3d   02&scid= 15&ssid=
```

File: "C:\Users\ADMIN\Desktop\Malware.cap" 147 kB 00:01:55
Packets: 1132 · Displayed: 1132 (100.0%) · Load time: 0:00.066
Profile: Default

FIGURE 15-18 **Advanced tools can produce complicated results.**

Note

Commercial tools from AppSec Labs and NowSecure also use a Linux Ubuntu VM and include various tools, many of which have been discussed in this section. AppSec Labs' AppUse VM includes a dashboard that directs the examiner to certain functions within the VM and is used primarily for penetration testing, but it can also be used to analyze malware from mobile devices.

Chapter Summary

By obtaining the file system from an Android device using ADB backup, non-invasive, or invasive physical partition recovery techniques, an examiner can uncover artifacts that could prove critical to any investigation. If partition information cannot be recovered, the examiner still has an opportunity to recover data from the device's internal or external media areas using MTP. App data, file settings, and persistent user data can be recovered from this area as well. Not all is lost if the partition-level file system data is not available; many users, and app developers, store data to the media area of the device, so as long as the device is unlocked or the examiner can access the device by circumventing the security, data can still be recovered.

Automating the recovery of Android artifacts is a difficult proposition for any software development team. Staying ahead of the ever-changing app and mobile device technology wave can be extremely difficult. When an examiner uses techniques discussed in this chapter and the two previous chapters in conjunction with the mobile forensic automated solution, a mind-boggling amount of data will be recovered. By using the scripting and query principles, even on databases and files already parsed by an automated tool, the examiner can be sure that recovered data is valid and properly recovered. The search for artifacts within an Android device should extend far below the surface of the "iceberg" to uncover the many files discussed in this chapter—from user settings to application data, this critical information is at the examiner's fingertips.

In addition, "tip of the iceberg data," such as SMS, MMS, call logs, and contacts, can be examined to uncover additional information that the automated tool neglected to report using a standard logical collection. By examining the SQLite databases, an examiner can look into the free pages, write-ahead logs, and the databases themselves to bring missing data to the surface.

Malware is a problem with mobile device forensics, and the examiner, as part of normal procedures, should be sure to scan and report any anomalies. This procedure comes from many years of court proceedings and reports that malware, Trojans, or viruses can install applications, download inappropriate images and videos, and modify the system somehow without the user's knowledge. With a mobile device's ability to extend itself into the Internet and network, along with the massive onslaught of Android malware, this type of issue is important to a court procedure. An examiner must take steps to scan the device's contents when possible to cover this critical examination gap within the current Android investigations.

16 Advanced Device Analysis: IoT, Wearables, and Drones

As discussed in Chapter 3, the era of IoT and other connected devices is upon every investigator and investigation. Like mobile devices, both iOS and Android, these devices and their associated apps can hold valuable information for the examiner. Digital traces left by the billions of devices within the Internet of Things, wearables, and drones could be the smoking guns in today's investigations. The "Wild West" once reserved for the examination of data from a mobile phone or a smart device is now the world of these emerging devices. In Chapters 13 and 15, information hidden beneath the "tip of the iceberg data" was discussed as a way to help uncover additional information from both iOS and Android devices. In this chapter, we will discuss both "tip of the iceberg data" and additional information that will assist in the investigation of these devices.

The limitations imposed by many of these emerging technologies results particularly from the support offered by commercial tool vendors—or lack thereof. These limitations are not the result of a lack of knowledge or research, but result from the sheer pace of deployment of these devices and our inability to keep up. In addition, the lion's share of data for many devices is stored within the Cloud. Although for many commercial tools, cloud storage represents another hurdle, it is often the reluctance of examiners to attempt to access this valuable data that keeps it hidden—and there is little case law or legal precedence regarding accessing and extracting data from the Cloud.

Even with lawful access to the device or cloud stores, the information that is often examined or located represents only a fraction of the data that is available to the examiner. This is not for a lack of trying, but results from the examiner's lack of support and ultimately lack of understanding of where to look. Now that IoT, wearables, and drones are so mainstream and are found in many homes, on many wrists, in the skies, and in vehicles, it is imperative that examiners understand not only what data is available, but where to look for it.

This chapter is not an exhaustive listing of the data, but more of a starting point to locations within the Cloud and mobile device backups, application sources, and fringe devices such as drones. "Fringe" might be an inaccurate descriptor, however, because the data contained inside these types of devices can be monumental to a case, and is more often used in investigations. Nevertheless, they are "fringe" devices simply because of a lack of documentation and commercial support for them—they are truly the Wild West in forensic examinations.

"Tip of the Iceberg Data"

The success of an investigation often comes down to the examiner's ability to access the data on these types of devices, but the examiner must also understand the format of the data and realize that not all automated forensic tools are created equally. For example, many IoT devices, vehicle systems, wearables, and drones can be accessed only using atypical connectors, or they may require invasive connections. Often, only "tip of the iceberg data," or the lowest-hanging fruit, will be available.

Unlike the data in iOS and Android devices (discussed in Chapters 13 and 15), however, which are generally parsed and obtainable using most commercial tools, IoT, wearable, and drone devices are not easily investigated. Although many of these devices' file systems may resemble those of Android devices, or they may be based on Android systems, commercial forensic tools often lack the "knowledge" of where this data is located on the devices. Knowledge, or logic, in the development sense, rests completely on the shoulders of investigators to code the correct path and then parse and extract the needed data. This data is then revealed to the investigator within the interface of the application. Because many, if not all, of these devices are not documented by manufacturers or on the Web, commercial forensic tool developers face plenty of research and development tasks. Often, reliance on the digital forensic community is essential to assist in their continued development to support more information about gathering evidence from these devices.

The following sections cover the information extracted from cloud stores and backups of various devices to include Google Home, Alexa, Apple Watch, Fitbit, and unmanned aircraft systems (UASs), or drones. Note that the information presented here is based on the devices mentioned, and it is not a global description for every smart watch, smart home device, or drone. However, this information offers the investigator a starting point in the investigation of these devices.

Smart Home Devices

Smart home devices have become popular around the globe, as more and more of these devices are made available each day to consumers. The types of smart home devices available today include doorbells, thermostats, and lighting—anything in the home capable of receiving an electrical current. The more the investigator knows about the type of data that is available from these devices, even at the "tip of the iceberg," the better qualified he or she will be to form a collective forensic picture for the investigation. Crimes have been solved and people have been exonerated with data gathered from smart home devices. Two devices and systems, Google Home using Assistant and Amazon Echo using Alexa, are discussed in the following sections.

Google Home

Access to "tip of the iceberg data" on a Google Home device is gained within the apps that access the device via the Google Assistant (such as Chromecast and others), as well as within the Google Cloud account for the device. The data available within these apps is extremely limited, however, because the apps are active web-based applications, meaning the data is not accessible if the device is offline. In other words, the app data will be available on the device only with a connection to the Google server. Without this connection, the data available on the device is limited to usernames and other general account information. As you know, to determine whether an app relies only on server data, forensic investigators would need to open the app with the device isolated. And if the app is not populated with data, there is a good chance app data will need to be retrieved from the app's server. Furthermore, the Google Assistant and Google Home app data is generally available only with a physical extraction of an iOS device and via root on an Android. A Standard iTunes backup or ADB backup will not recover most of the valuable data.

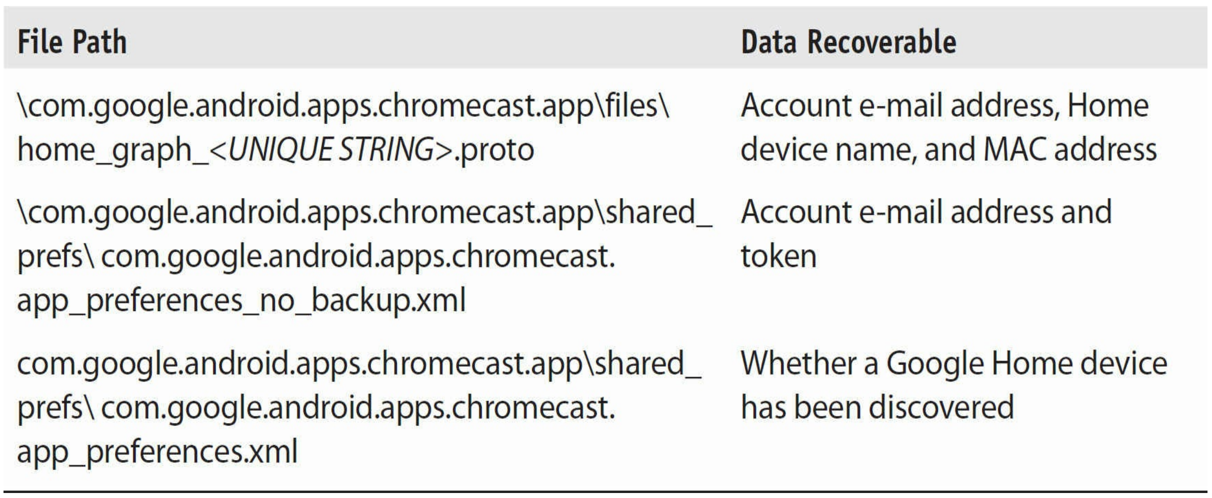
The user needs a Google account to set up the Google Home device. Applications on the smart home device will also need to use the associated Google account to communicate with the device. The smart home device is visible via Bluetooth and Wi-Fi to any device that has paired with it and can play media over the device. For example, data from the Chromecast app (which was replaced by the Google Home app, but the Chromecast name was kept for the app bundle) in an Android device can be found within com.google.android.apps.chromecast.app/sp/com.google.android.apps.chromecast.app_preferences.xml with the following flags:

```
<boolean name="hasAssistantLinkedKey" value="true" />
<boolean name="assistantDeviceDiscovered" value="true" />
```

Imagine if, during the investigation, information was found within an app on a mobile device, but the Google Home device was not located in the area. This definitely could mean more investigation would be needed, especially if Assistant was not installed on the mobile device.

Commercial tools, as well as public APIs, can be used to extract data from Google Home cloud activity, but few parse the data from the apps on the mobile devices. Apps such as Home are almost entirely active, so the data is not accessible if the device is offline. Within the Android file system, the investigator can find the Google Home app data at com.google.android.apps.chomecast.app and Assistant app data at com.google.android.apps.googleassistant.app. Assistant data is also accessible via the My Activity cloud account, as is Google Home data. Table 16-1 lists some information that can be recovered from files within the Google Home application on the mobile device.

TABLE 16-1 **Google Home App Contains Limited Data, But Can Assist with an Investigation**



File Path	Data Recoverable
\com.google.android.apps.chromecast.app\files\ home_graph_<*UNIQUE STRING*>.proto	Account e-mail address, Home device name, and MAC address
\com.google.android.apps.chromecast.app\shared_ prefs\ com.google.android.apps.chromecast. app_preferences_no_backup.xml	Account e-mail address and token
com.google.android.apps.chromecast.app\shared_ prefs\ com.google.android.apps.chromecast. app_preferences.xml	Whether a Google Home device has been discovered

Information that can be obtained from both direct API communications with the Google Home device and Google My Activity are immense, however. The API for Google Home is primarily exposed to handle the creation of "actions" that can be used by the Google Home device. For cloud tools, the support for My Activity (which tracks the user's activity) is initiated with the Google Account, username/token, backup passwords, Authenticator, or app-specific passwords. Both are discussed in the following sections.

Direct to Device

Using the Google API, the investigator can interrogate the device(s) as long as the IP address of the device is known. The device's IP address can be collected using one of several network scanning apps run from a mobile device, the Google Home app, or the router. Note that access to the Wi-Fi network that is connected to the devices is required for both identifying the Home device if using an app and during the interrogation of the device. This type of access requires that the investigator weigh the pros and cons of access in this situation. The information that can be extracted over TCP within the internal network is quite impressive, however, to include but not be limited to the following:

- Device name
- Time zone
- Language
- MAC address
- Public encryption key
- All Bluetooth devices in the area
- Connected Wi-Fi

Once connected to the device, the investigator can retrieve the data using `GET` statements. (The opposite command, `POST`, is one the investigator should not employ, for obvious reasons.) Using `GET` will allow for the extraction of the data without adding data to the specific records requested. The data is returned in JavaScript Object Notation (JSON) (see Chapters 12 and 13). To communicate and obtain the information from the device, the investigator will need to use an interface that will allow for communication over TCP using `GET` commands. Several REST web-based tools are available for this. In the following exercise, the Advanced REST client (https://github.com/advanced-rest-client) is used. The commands covered here are all undocumented calls by Google but are outlined by rithvikvibhu on Github (https://rithvikvibhu.github.io/GHLocalApi).

The HTTP port that is currently used by Google Home is 8008. To access the device using the REST utility, the investigator would use the following template:

```
GET http://<discoveredIP>:8008/setup/<file>
```

Table 16-2 lists some of the files and also the expected results from the data retrieval.

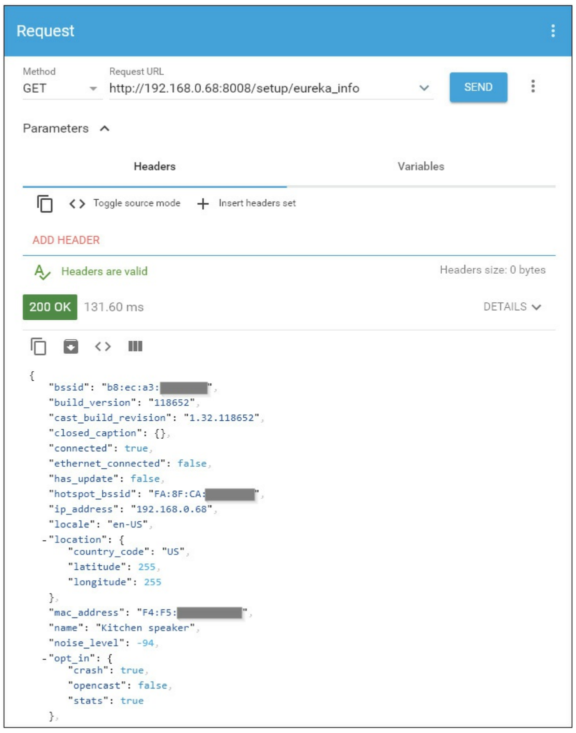
TABLE 16-2 **Important Files Within the Google Home Device Retrieved Using HTTP**

File	Result
eureka_info	By far the most verbose file containing BSSID (MAC address of the wireless access point), hotspot BSSID, IP address, current country, MAC of device, name of the device, currently connected SSID, time zone, uptime, and more. Hence the name, Eureka! (Figure 16-1).
/assistant/alarms	All active alarms. Includes the date and time, and the fire_time key is UNIX (ms).
configured_networks	All the reachable and configured Wi-Fi networks to include the SSID.
get_bonded	Any device that has been paired with the Google Home device to include its name, if it's currently connected, the date the device was paired, the last data that was paired, and more. The date keys are UNIX (ms) (Figure 16-2).
scan_results	All active alarms. The date and time are listed, and the fire_time key is UNIX (ms).

Google Home Cloud (My Activity)

The majority of Google Home device data will be stored in the user's My Activity account. The information stored in the user's account will comprise the actions that the device performed based upon the user's suggestions. Both the Assistant subcategory and the Voice & Audio within My Activity can provide a gold mine of Google Home user data. The Google Home device will capture any phrase the user spoke, as well as the return data by the Home device. The actual voice is recorded and analyzed on the Google servers and the voice is stored. So not only can an investigator see what the Google Home device transcribed, but he or she can hear the actual recording used to make the transcription! This could be monumental to a case when a subject denies being at a location, or when a subject utters the words to search for "closest store to buy duct tape," and his or her actual voice has been recorded.

The subsection in My Activity, Voice & Audio, is displayed in Figure 16-3 from an Oxygen Forensic Cloud extraction, and the platform Google Home is identified. Also, the Assistant is a subsection that lists usage of the Assistant app on both iOS and Android devices, as well as using Assistant with the Home device. Figure 16-4 shows data collected from My Activity's subsection Assistant.

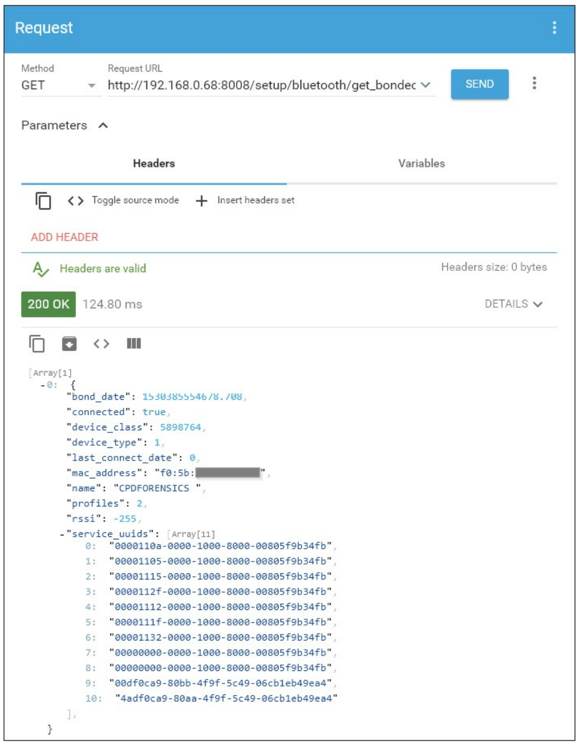

Request

Method: GET
Request URL: http://192.168.0.68:8008/setup/eureka_info

200 OK 131.60 ms

```
{
   "bssid": "b8:ec:a3:[redacted]",
   "build_version": "118652",
   "cast_build_revision": "1.32.118652",
   "closed_caption": {},
   "connected": true,
   "ethernet_connected": false,
   "has_update": false,
   "hotspot_bssid": "FA:8F:CA:[redacted]",
   "ip_address": "192.168.0.68",
   "locale": "en-US",
  -"location": {
      "country_code": "US",
      "latitude": 255,
      "longitude": 255
   },
   "mac_address": "F4:F5:[redacted]",
   "name": "Kitchen speaker",
   "noise_level": -94,
  -"opt_in": {
      "crash": true,
      "opencast": false,
      "stats": true
   },
```

FIGURE 16-1 Running a GET command on an active Google Home device to retrieve the contents of the file eureka_info

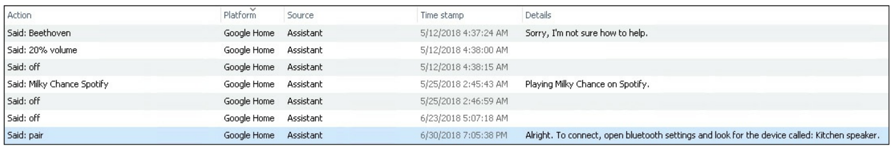

Request

Method: GET
Request URL: http://192.168.0.68:8008/setup/bluetooth/get_bondec

200 OK 124.80 ms

```
[Array[1]
 -0:  {
      "bond_date": 1530385554678.708,
      "connected": true,
      "device_class": 5898764,
      "device_type": 1,
      "last_connect_date": 0,
      "mac_address": "f0:5b:[redacted]",
      "name": "CPDFORENSICS ",
      "profiles": 2,
      "rssi": -255,
     -"service_uuids": [Array[11]
         0:  "0000110a-0000-1000-8000-00805f9b34fb",
         1:  "00001105-0000-1000-8000-00805f9b34fb",
         2:  "00001115-0000-1000-8000-00805f9b34fb",
         3:  "0000112f-0000-1000-8000-00805f9b34fb",
         4:  "00001112-0000-1000-8000-00805f9b34fb",
         5:  "0000111f-0000-1000-8000-00805f9b34fb",
         6:  "00001132-0000-1000-8000-00805f9b34fb",
         7:  "00000000-0000-1000-8000-00805f9b34fb",
         8:  "00000000-0000-1000-8000-00805f9b34fb",
         9:  "00df0ca9-80bb-4f9f-5c49-06cb1eb49ea4",
         10:  "4adf0ca9-80aa-4f9f-5c49-06cb1eb49ea4"
      ],
   }
```

FIGURE 16-2 Running a GET command on an active Google Home device to retrieve the contents of the get_bonded file



Action	Platform	Source	Time stamp	Details
Said: Beethoven	Google Home	Assistant	5/12/2018 4:37:24 AM	Sorry, I'm not sure how to help.
Said: 20% volume	Google Home	Assistant	5/12/2018 4:38:00 AM	
Said: off	Google Home	Assistant	5/12/2018 4:38:15 AM	
Said: Milky Chance Spotify	Google Home	Assistant	5/25/2018 2:45:43 AM	Playing Milky Chance on Spotify.
Said: off	Google Home	Assistant	5/25/2018 2:46:59 AM	
Said: off	Google Home	Assistant	6/23/2018 5:07:18 AM	
Said: pair	Google Home	Assistant	6/30/2018 7:05:38 PM	Alright. To connect, open bluetooth settings and look for the device called: Kitchen speaker.

FIGURE 16-3 **Voice & Audio data extracted from Google My Activity that clearly displays the action and the response**

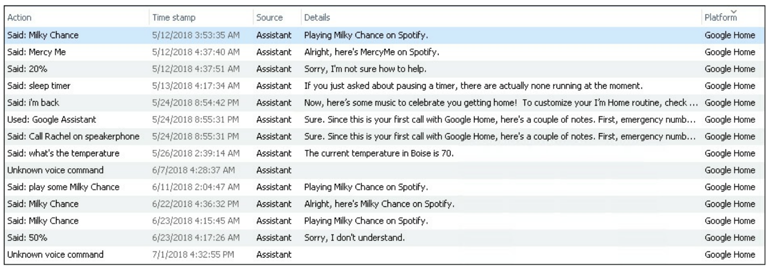
An investigation into the Google Home device, as well as the artifacts found both within the app and in the Cloud service, can often uncover additional data for any investigation. From information regarding searches a user has conducted using the Home device, to other connected devices that have been attached to the Home device, to locations that are captured upon each request, this important data should not be overlooked.

Alexa

Devices that access Alexa (such as Echo, Echo Dot, and so on) store data primarily in the Cloud. These devices communicate with and to mobile phones via the Amazon Alexa app and control requests to play music, integrate other IoT devices, and provide weather or other news updates. Again, much like other devices, these devices use an API to communicate to and from the Alexa device over the user's internal network and then outbound to the Alexa Cloud. Communication from the Alexa Cloud to other IoT devices can be used to control smart TVs, lights, thermostats, and more.

"Tip of the iceberg data" will be discussed in this section, as well as some additional artifacts. The API for Alexa is not public, but this information is hosted at several documented sources, which will be covered in detail. A mobile device app is generally used to set up the device and control many of the "skills" that define the usage of the Alexa-powered device. Users of the Amazon brand of smart devices will need an Amazon account to set up the device, and to access the data within the account or Cloud, investigators will need the user's credentials.

Investigators can use commercial tools as well as unpublished APIs to extract data from the Alexa Cloud, but few tools parse the data from the apps on mobile devices. Currently, Magnet Forensics, Cellebrite, Paraben, and Oxygen provide limited support for the apps on a mobile device. As mentioned, apps such as Alexa are almost entirely active, and the data is not accessible if the device is offline. Some account data can be useful for interrogation of the cloud service, but much of the user data is generally encrypted. Alexa, the app bundle, and folders can be found at com.amazon.dee.app for Android and com.amazon.echo for iOS devices. The folder is obtainable only in a physical collection of the Android device. The folder structure contains multiple folders and files (Figure 16-5). Table 16-3 lists some information that can be recovered from the Alexa application on the mobile device.

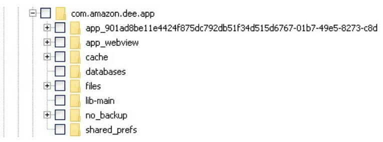


Action	Time stamp	Source	Details	Platform
Said: Milky Chance	5/12/2018 3:53:35 AM	Assistant	Playing Milky Chance on Spotify.	Google Home
Said: Mercy Me	5/12/2018 4:37:40 AM	Assistant	Alright, here's MercyMe on Spotify.	Google Home
Said: 20%	5/12/2018 4:37:51 AM	Assistant	Sorry, I'm not sure how to help.	Google Home
Said: sleep timer	5/13/2018 4:17:34 AM	Assistant	If you just asked about pausing a timer, there are actually none running at the moment.	Google Home
Said: i'm back	5/24/2018 8:54:42 PM	Assistant	Now, here's some music to celebrate you getting home! To customize your I'm Home routine, check ...	Google Home
Used: Google Assistant	5/24/2018 8:55:31 PM	Assistant	Sure. Since this is your first call with Google Home, here's a couple of notes. First, emergency numb...	Google Home
Said: Call Rachel on speakerphone	5/24/2018 8:55:31 PM	Assistant	Sure. Since this is your first call with Google Home, here's a couple of notes. First, emergency numb...	Google Home
Said: what's the temperature	5/26/2018 2:39:14 AM	Assistant	The current temperature in Boise is 70.	Google Home
Unknown voice command	6/7/2018 4:28:37 AM	Assistant		Google Home
Said: play some Milky Chance	6/11/2018 2:04:47 AM	Assistant	Playing Milky Chance on Spotify.	Google Home
Said: Milky Chance	6/22/2018 4:36:32 PM	Assistant	Alright, here's Milky Chance on Spotify.	Google Home
Said: Milky Chance	6/23/2018 4:15:45 AM	Assistant	Playing Milky Chance on Spotify.	Google Home
Said: 50%	6/23/2018 4:17:26 AM	Assistant	Sorry, I don't understand.	Google Home
Unknown voice command	7/1/2018 4:32:55 PM	Assistant		Google Home

FIGURE 16-4 **The Assistant section in Google My Activity holds data that is captured by Google Home.**

TABLE 16-3 **Alexa App Data from Within Both iOS and Android Devices**

File Path	Information
com.amazon.dee.app/app_<*GUID*>/eventsFile	Device information app is installed on, time zone, Android version, and more (Figure 16-6)
com.amazon.dee.app/databases/map_data_storage_v2.db	Account information: table account_data Amazon ID string: account_data_directed_id (Figure 16-7)
com.amazon.dee.app/databases/arn:aws:remote-config:<*aws region user ID*>:appconfig:<*UNIQUEID*>	Configuration table has Amazon Mobile Service URL used to gather contacts, messages, and media (Figure 16-8)
com.amazon.dee.app/databases/RKStorage	Cards and features found in catalystLocalStorage table
com.amazon.dee.app/cache	Folder with files downloaded to the app from Amazon server by the user; could contain audio files or other artifacts
com.amazon.dee.app/databases/comms.db	Stores devices using messaging (calls/messages); data container is encrypted



```
com.amazon.dee.app
    app_901ad8be11e4424f875dc792db51f34d515d6767-01b7-49e5-8273-c8d
    app_webview
    cache
    databases
    files
    lib-main
    no_backup
    shared_prefs
```

FIGURE 16-5 **Folder structure of Alexa app within a Samsung SM-G900T running 5.11 Android**

The data is primarily stored as JSON strings either in individual files or within the SQLite database. Some of the information can be pertinent to a case, such as the account, cards, and endpoint strings. Brian Moran of BriMor Labs has presented research on the recovery of contacts and messages as well as call history by combining the endpoint URL in the configurations table of the arn:aws file and the Amazon ID string found in map_data_storage_v2.db. If the investigator is authenticated in the Amazon account and runs a script, call logs and messages can be retrieved. This is covered in the next section.

Direct to Amazon Cloud

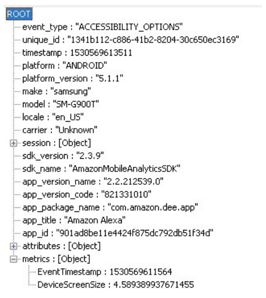
As with the Google Home device, the data from the Cloud will be paramount to the overall investigation. This information can be obtained using HTTP calls as well as by using many cloud extraction tools. Using information that is obtained from the mobile app, to include the endpoint URL from the configurations table and Amazon ID string, an investigator can query the account and recover additional information.

Using Burp Suite Professional and a trusted SSL certificate, Karl Fosaaen (https://blog.netspi.com/speaking-to-a-city-of-amazon-echoes/) reported he was able to capture traffic from an Alexa device. In doing so, he obtained the commsID; this can be used by all investigators, along with the data from the mobile app, to create an HTTP call and recover additional data unavailable even from within the web portal for Alexa (https://alexa.amazon.com), messages, and call log data. The only prerequisite for the investigator is that he or she must know the Amazon login information and be signed into the account. The user's contacts, even if the contact does not use an Alexa-enabled device, are populated in the Alexa Cloud. Calls and messaging, if they occurred via the Alexa app to other Amazon device users, are recoverable using this method as well. The investigator need only run the following commands on an authenticated account.

To recover contact data, run the following:

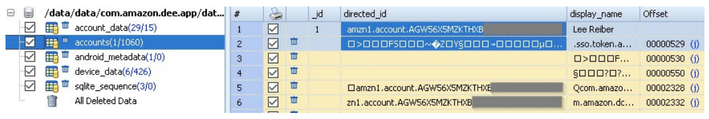
```
<Endpoint URL obtained in configurations table of the arn:aws file>.
amzn1.comms.id.person.amzn1~<Amazon ID string found in map_data_storage_
v2.db >/contacts
```

The results, in Figure 16-9, are in Notepad++ with a JSON viewer and clearly show all contacts information after running the contact script.



```
ROOT
├ event_type : "ACCESSIBILITY_OPTIONS"
├ unique_id : "1341b112-c886-41b2-8204-30c650ec3169"
├ timestamp : 1530569613511
├ platform : "ANDROID"
├ platform_version : "5.1.1"
├ make : "samsung"
├ model : "SM-G900T"
├ locale : "en_US"
├ carrier : "Unknown"
⊞ session : [Object]
├ sdk_version : "2.3.9"
├ sdk_name : "AmazonMobileAnalyticsSDK"
├ app_version_name : "2.2.212539.0"
├ app_version_code : "821331010"
├ app_package_name : "com.amazon.dee.app"
├ app_title : "Amazon Alexa"
├ app_id : "901ad8be11e4424f875dc792db51f34d"
⊞ attributes : [Object]
⊟ metrics : [Object]
    ├ EventTimestamp : 1530569611564
    └ DeviceScreenSize : 4.589389937671455
```

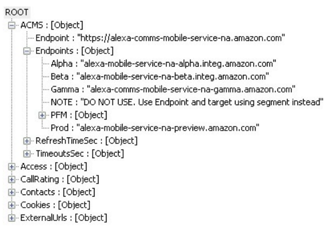
FIGURE 16-6 The eventsFile is a JSON file containing identifying information about the device.



/data/data/com.amazon.dee.app/dat...
- ✓ account_data(29/15)
- ✓ accounts(1/1060)
- ✓ android_metadata(1/0)
- ✓ device_data(6/426)
- ✓ sqlite_sequence(3/0)
- All Deleted Data

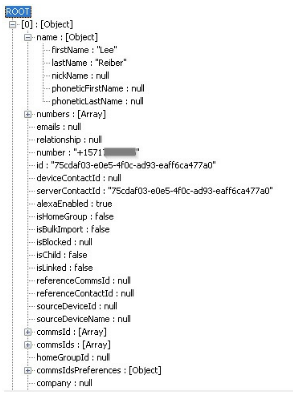
#			_id	directed_id	display_name	Offset
1	✓		1	amzn1.account.AGW56X5MZKTHXB[redacted]	Lee Reiber	
2	✓	🗑		□>□□□FS□□□~�Z□Y§□□□-□□□□□µ□...	.sso.token.a...	00000529 (j)
3	✓	🗑			□>□□□F...	00000530 (j)
4	✓	🗑			§□□□?□?...	00000550 (j)
5	✓	🗑		□amzn1.account.AGW56X5MZKTHXE[redacted]	Qcom.amazo...	00002328 (j)
6	✓	🗑		zn1.account.AGW56X5MZKTHXB[redacted]	m.amazon.dc...	00002332 (j)

FIGURE 16-7 The map_data_storage_v2.db SQLite file contains the account name as well as the Amazon ID string.



```
ROOT
  ACMS : [Object]
    Endpoint : "https://alexa-comms-mobile-service-na.amazon.com"
    Endpoints : [Object]
      Alpha : "alexa-mobile-service-na-alpha.integ.amazon.com"
      Beta : "alexa-mobile-service-na-beta.integ.amazon.com"
      Gamma : "alexa-comms-mobile-service-na-gamma.amazon.com"
      NOTE : "DO NOT USE. Use Endpoint and target using segment instead"
      PFM : [Object]
      Prod : "alexa-mobile-service-na-preview.amazon.com"
    RefreshTimeSec : [Object]
    TimeoutsSec : [Object]
  Access : [Object]
  CallRating : [Object]
  Contacts : [Object]
  Cookies : [Object]
  ExternalUrls : [Object]
```

FIGURE 16-8 **File showing endpoint URL that, when combined with data from map_data_storage_v2.db, investigators can use to obtain user calls, messages, and media**



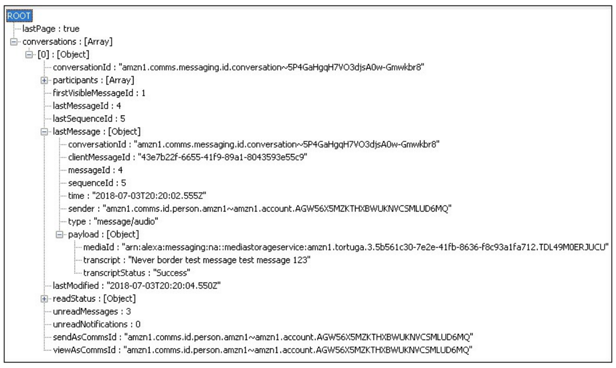
```
ROOT
  [0] : [Object]
    name : [Object]
      firstName : "Lee"
      lastName : "Reiber"
      nickName : null
      phoneticFirstName : null
      phoneticLastName : null
    numbers : [Array]
    emails : null
    relationship : null
    number : "+1571[redacted]"
    id : "75cdaf03-e0e5-4f0c-ad93-eaff6ca477a0"
    deviceContactId : null
    serverContactId : "75cdaf03-e0e5-4f0c-ad93-eaff6ca477a0"
    alexaEnabled : true
    isHomeGroup : false
    isBulkImport : false
    isBlocked : null
    isChild : false
    isLinked : false
    referenceCommsId : null
    referenceContactId : null
    sourceDeviceId : null
    sourceDeviceName : null
    commsId : [Array]
    commsIds : [Array]
    homeGroupId : null
    commsIdsPreferences : [Object]
    company : null
```

FIGURE 16-9 **All contacts are shown in Notepad++ after running the contacts script.**

For messaging and calls data, run the following:

```
<Endpoint URL obtained in configurations table of the arn:aws file>.
amzn1.comms.id.person.amzn1~<Amazon ID string found in map_data_storage_
v2.db >/conversations
```

You can think of conversations as "threads" between contacts. If an investigator wants to extract a single conversation, the conversationId (shown in Figure 16-10) must be appended and used in the HTTP request, as shown here:



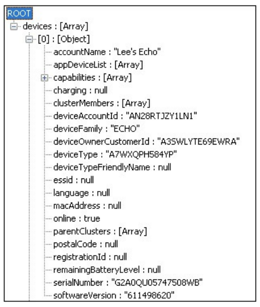
```
ROOT
  lastPage : true
  conversations : [Array]
    [0] : [Object]
      conversationId : "amzn1.comms.messaging.id.conversation~5P4GaHgqH7VO3djsA0w-Gmwkbr8"
      participants : [Array]
      firstVisibleMessageId : 1
      lastMessageId : 4
      lastSequenceId : 5
      lastMessage : [Object]
        conversationId : "amzn1.comms.messaging.id.conversation~5P4GaHgqH7VO3djsA0w-Gmwkbr8"
        clientMessageId : "43e7b22f-6655-41f9-89a1-8043593e55c9"
        messageId : 4
        sequenceId : 5
        time : "2018-07-03T20:20:02.555Z"
        sender : "amzn1.comms.id.person.amzn1~amzn1.account.AGW56X5MZKTHXBWUKNVCSMLUD6MQ"
        type : "message/audio"
        payload : [Object]
          mediaId : "arn:alexa:messaging:na::mediastorageservice:amzn1.tortuga.3.5b561c30-7e2e-41fb-8636-f8c93a1fa712.TDL49M0ERJUCU"
          transcript : "Never border test message test message 123"
          transcriptStatus : "Success"
      lastModified : "2018-07-03T20:20:04.550Z"
      readStatus : [Object]
      unreadMessages : 3
      unreadNotifications : 0
      sendAsCommsId : "amzn1.comms.id.person.amzn1~amzn1.account.AGW56X5MZKTHXBWUKNVCSMLUD6MQ"
      viewAsCommsId : "amzn1.comms.id.person.amzn1~amzn1.account.AGW56X5MZKTHXBWUKNVCSMLUD6MQ"
```

FIGURE 16-10 **Results shown in Notepad++ with JSON viewer of messages from the Alexa Cloud**

```
<Endpoint URL obtained in configurations table of the arn:aws file>.
amzn1.comms.id.person.amzn1~<Amazon ID string found in map_data_storage_
v2.db >/conversations/<conversationId>messages?count=<number of messages
you want, and a max of 100>&sort=<asc or desc>
```

This information was documented by Brian Moran and Karl Fosaaen and can be used by an investigator to conduct live analysis of the Alexa Cloud, along with the following information on accessing additional data using undocumented and documented APIs.

A list of HTTP requests is provided in Table 16-4 (obtained from http://analyticphysics.com/Diversions/Accessing Amazon Echo Data with JavaScript.htm and www.dfrws.org/sites/default/files/session-files/paper_digital_forensic_approaches_for_amazon_alexa_ecosystem.pdf as a reference. As with the previous HTTP commands, the investigator must have access to the device and be authenticated for the Amazon account. The response returned will be in JSON format (see Figure 16-11). As with other responses, using a tool such as Notepad++ with a JSON viewer will help to decode and format the data for easy investigation.



```
ROOT
⊟ devices : [Array]
    ⊟ [0] : [Object]
        accountName : "Lee's Echo"
        appDeviceList : [Array]
      ⊞ capabilities : [Array]
        charging : null
        clusterMembers : [Array]
        deviceAccountId : "AN28RTJZY1LN1"
        deviceFamily : "ECHO"
        deviceOwnerCustomerId : "A3SWLYTE69EWRA"
        deviceType : "A7WXQPH584YP"
        deviceTypeFriendlyName : null
        essid : null
        language : null
        macAddress : null
        online : true
        parentClusters : [Array]
        postalCode : null
        registrationId : null
        remainingBatteryLevel : null
        serialNumber : "G2A0QU05747508WB"
        softwareVersion : "611498620"
```

FIGURE 16-11 **A query of the device information provides the name of the device, ID, customer ID, and serial number.**

TABLE 16-4 **Listing of HTTP Request URLs and Data Returned**

URL	Return
https://pitangui.amazon.com/api/devices/device	Device name, device type, serial number, address, other specifics (Figure 16-11)
https://pitangui.amazon.com/api/device-preferences	Device account ID, serial number, postal code by GPS, customer ID, time zone, and more
https://pitangui.amazon.com/api/bluetooth	Paired devices and device SSID
https://pitangui.amazon.com/api/wake-word	Device type, serial number, and current wake word
https://pitangui.amazon.com/api/authentication	User e-mail, username, and customer ID
https://pitangui.amazon.com/api/notifications	Account information and Amazon ID string in tables account_data field and account_data_directed_id, respectively (Figure 16-7)
https://pitangui.amazon.com/api/media/phoenix	Networked hardware with location information
https://pitangui.amazon.com/api/cards	Voice commands
https://pitangui.amazon.com/api/utterance/audio/data?id=*<ID from Link in card for Audio of Utterance>*	User's voice before transcription is stored indefinitely unless user removes it manually

Cards are the voice instructions given to Alexa devices; the investigator can access these individually if they are not included in the response. In addition, the audio files that are referred to within the JSON format are stored and are retrievable from the Amazon servers. This audio data is stored indefinitely, unless the user removes it. Amazon clearly states the following in the user's account under Your Devices | Device Actions | Manage Voice Recordings:

> When you use your device, we keep the voice recordings associated with your account to learn your voice and how you speak to improve the accuracy of the results provided to you and to improve our services.

This type of information can be critical to understand if your investigation involves the retrieval of voice information from the Alexa device.

Currently, only two commercial cloud tools can access data from a user's Alexa account: Oxygen Forensic Cloud Extractor and Cellebrite Cloud Analyzer. The extraction techniques are built upon the data that has been provided in this section as researched, with generally the same extracted data. Commercial tools also need to have access to the username and password combination and/or token. The advantage of using a commercial cloud tool will be the data formatting as well as the automated logic. However, the examiner's understanding of the information obtained from the Cloud by using the commands provided throughout the section should prove valuable if testimony is required.

Wearable Devices

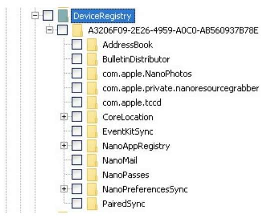
Wearables are everywhere—people wear mobile computing devices around their wrists, on their heads, and on their bodies. With the Fitbit and Apple Watch almost mainstream, most users wear these devices on their wrists. As with any other form of digital storage device, an investigator must consider what type of data can be recovered. The technology of these devices continues to grow and will continue to advance. An investigator must be prepared to look for these types of devices at the digital forensic scene and be prepared to look for ways to extract the information from the associated mobile app, if applicable, directly from the device via advanced methods, and from the device's cloud service.

No longer are investigators simply dealing with a mobile device; our connected world demands the investigator start thinking of these additional devices and their content. Failing to become educated on the new possibilities of the wearable device, and its data, could result in an investigator missing data that could likely close a case. In Chapter 3, we discussed some high-level data within the backups of the Apple Watch as well as some data that could be lingering within the Fitbit. Like the data from smart home devices, much of the information on these devices is stored within the Cloud.

This section covers the Apple Watch and the Fitbit. The "tip of the iceberg data" will be discussed as well as information from the Cloud. With the number of wearables, particularly watches, on the rise, the investigator should be prepared to delve into the file systems of these devices.

Apple Watch

Data for the Apple Watch is backed up within a standard iTunes backup as well as an iCloud backup. There is, however, a caveat to this: if health and fitness information is to be backed up, then the backup must be done via iCloud or the iTunes backup must be encrypted. Within an iTunes backup, the investigator should look to the path \private\var\mobile\Library\DeviceRegistry for data that is backed up to the mobile device and subsequently found in the iPhone backup (Figure 16-12). Also, the investigator can find current settings for the device at \private\var\mobile\Library\DeviceRegistry.state.



```
DeviceRegistry
  A3206F09-2E26-4959-A0C0-AB560937B78E
    AddressBook
    BulletinDistributor
    com.apple.NanoPhotos
    com.apple.private.nanoresourcegrabber
    com.apple.tccd
    CoreLocation
    EventKitSync
    NanoAppRegistry
    NanoMail
    NanoPasses
    NanoPreferencesSync
    PairedSync
```

FIGURE 16-12 **File system location for the backup of the Apple Watch in a standard iTunes backup**

Information that is shown on the watch is often simply a reference to the information within the stock app databases. Examples of this are the AddressBook folder, located at \private\var\mobile\Library\DeviceRegistry\<*Device ID*>\AddressBook; and within the ABSyncState table of ABSABShadow.db SQLite guid field is a GUID that can be matched within the ABPerson table of \private\var\mobile\Library\AddressBook\AddressBook.sqlitedb. The examiner can use a simple SQL command to show the names accessible with the Apple Watch. The investigator can also find data at additional locations within the Apple Watch backup file system, as discussed next.

Settings File

The main watch settings file is located at \private\var\mobile\Library\DeviceRegistry.state\history.plist. The history.plist contains a history node with several keys that are made up of arrays. Each array first lists a ClassName NRDeviceCollectionHistoryEntry with an associated date. The date is represented as an OS X UNIX time and identifies the date and time of the backup snapshot. Each array under the history node will have the ClassName NRDeviceCollectionHistoryEntry and its associated date of collection. Within each array, if changes have been made, additional investigative values can include the following:

- Hardware model (hwModelStr)
- Pairing device ID (pairingID)
- Type of device (systemVersion)
- System build (systemBuildVersion)
- Location of the backup (localPairingDataStorePath
- Wi-Fi MAC address (WIFIMACAddress)
- Serial number (serialNumber>UUID)
- Preferred languages (preferredLanguages)
- Name of the operating system (systemName)
- If the device is paired (bluetoothPaired)
- Last active date (lastActiveDate) OS X Epoch UNIX
- Date device paired (pairedDate) OS X Epoch UNIX
- Version or product type (productType)
- IMEI, MEID, CSN (UUID is key)
- Name of device (deviceNameString)

Generally, the first collection history array will contain the majority of strings.

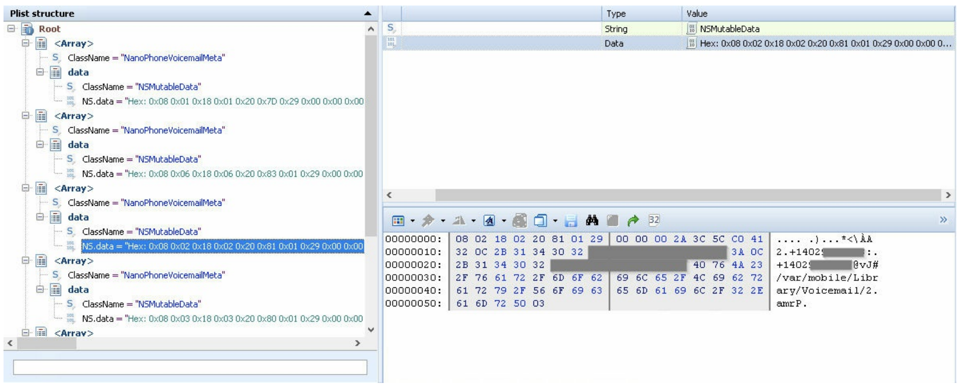
User Information

Within the iOS device's backup that was paired with the Apple Watch the investigator can find valuable information to assist with the case. This data can range from applications that have been installed on the device to voicemails that have been delivered to the wearable. The information listed next includes some paths within the device backup that can hold this critical data.

- **\private\var\mobile\Library\DeviceRegistry\<*DEVICE GUID*>\NanoMail\registry.sqlite** This file contains many tables with GUIDS and paths of e-mail sent and received to the watch. Tables include COMPOSED_MESSAGE, DELETED_MESSAGE, MAILBOX, and SYNCED_ACCOUNT. Within the SYNCED_ACCOUNT table, the user ID as well as e-mail addresses can be located for the account. Within the MAILBOX table, the investigator can match the user ID from the SYNCED_ACCOUNT table to the correct e-mail path URL. Also within the MAILBOX table, the SYNC_ENABLED field will indicate whether the mailbox was currently being synched (0 = no and 1 =

yes).

- **\private\var\mobile\Library\DeviceRegistry\<*DEVICE GUID*>\NanoPreferencesSync\NanoDomains\com.apple.Carousel** This file is a property list that clearly defines the applications that are installed on the Apple Watch. The FavoriteApplications node lists the built-in Apple Watch applications, and the IconPositions node contains an embedded property list with all installed apps in a series of arrays.
- **\private\var\mobile\Library\DeviceRegistry\<*DEVICE GUID*>\com.apple.tccd\WatchKitApplications** This property list contains the version numbers of all the third-party applications that are installed on the Apple Watch.
- **\private\var\mobile\Library\DeviceRegistry\<*DEVICE GUID*>\NanoPreferencesSync\NanoDomains\com.apple.mobilephone** This property list contains the path for voicemails with the key kVoicemailForReplicationKey. This key contains an embedded property list with multiple arrays with a data key (NS.data). The data includes the phone numbers and also the path to the voicemail within the iOS device file system. See Figure 16-13.



```
Plist structure
Root
  <Array>
    ClassName = "NanoPhoneVoicemailMeta"
    data
      ClassName = "NSMutableData"
      NS.data = "Hex: 0x08 0x01 0x18 0x01 0x20 0x7D 0x29 0x00 0x00 0x00
  <Array>
    ClassName = "NanoPhoneVoicemailMeta"
    data
      ClassName = "NSMutableData"
      NS.data = "Hex: 0x08 0x06 0x18 0x06 0x20 0x83 0x01 0x29 0x00 0x00
  <Array>
    ClassName = "NanoPhoneVoicemailMeta"
    data
      ClassName = "NSMutableData"
      NS.data = "Hex: 0x08 0x02 0x18 0x02 0x20 0x81 0x01 0x29 0x00 0x00
  <Array>
    ClassName = "NanoPhoneVoicemailMeta"
    data
      ClassName = "NSMutableData"
      NS.data = "Hex: 0x08 0x03 0x18 0x03 0x20 0x80 0x01 0x29 0x00 0x00
  <Array>
```

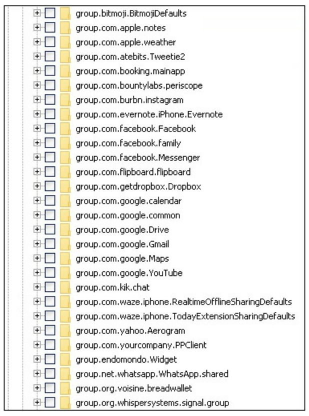
	Type	Value
S	String	NSMutableData
	Data	Hex: 0x08 0x02 0x18 0x02 0x20 0x81 0x01 0x29 0x00 0x00 0...

```
00000000:  08 02 18 02 20 81 01 29  00 00 00 2A 3C 5C C0 41  .... .)...*<\ÀA
00000010:  32 0C 2B 31 34 30 32 [redacted] 3A 0C  2.+1402[redacted]:.
00000020:  2B 31 34 30 32 [redacted] 40 76 4A 23  +1402[redacted]@vJ#
00000030:  2F 76 61 72 2F 6D 6F 62  69 6C 65 2F 4C 69 62 72  /var/mobile/Libr
00000040:  61 72 79 2F 56 6F 69 63  65 6D 61 69 6C 2F 32 2E  ary/Voicemail/2.
00000050:  61 6D 72 50 03                                    amrP.
```

FIGURE 16-13 **Data in the kVoicemailForReplicationKey showing the phone number and path to the voicemail**

- **\private\var\mobile\Library\DeviceRegistry\<*DEVICE GUID*>\NanoPreferencesSync\NanoDomains\com.apple.nanoprefsyncd** This plist file lists all the settings for the device that may help an investigator understand the following watch settings:
  - Voice dialing enabled (allowVoiceDialing)
  - Camera enabled (allowCamera)
  - Whether backup can occur via iTunes (allowiTunes) or iCloud (allowCloudBackup) or both
  - If chat is enabled (allowChat)
  - If Find My Device is on (allowFindMyDevice)
  - If a PIN is enabled (forcePIN)

Remember as discussed in Chapter 3 that the Apple Watch uses the shared area of many of the applications that are added as applications and the default Apple apps. The shared area is denoted by group.<*Reverse DNS app bundle name*>, as shown in Figure 16-14. Also, it is not necessary to have an Apple Watch paired to the device to recover user information from health information.



```
group.bitmoji.BitmojiDefaults
group.com.apple.notes
group.com.apple.weather
group.com.atebits.Tweetie2
group.com.booking.mainapp
group.com.bountylabs.periscope
group.com.burbn.instagram
group.com.evernote.iPhone.Evernote
group.com.facebook.Facebook
group.com.facebook.family
group.com.facebook.Messenger
group.com.flipboard.flipboard
group.com.getdropbox.Dropbox
group.com.google.calendar
group.com.google.common
group.com.google.Drive
group.com.google.Gmail
group.com.google.Maps
group.com.google.YouTube
group.com.kik.chat
group.com.waze.iphone.RealtimeOfflineSharingDefaults
group.com.waze.iphone.TodayExtensionSharingDefaults
group.com.yahoo.Aerogram
group.com.yourcompany.PPClient
group.endomondo.Widget
group.net.whatsapp.WhatsApp.shared
group.org.voisine.breadwallet
group.org.whispersystems.signal.group
```

FIGURE 16-14 **Location for the sharing of application data between apps given permissions as well as Apple Watch**

Two database files should also be investigated:

- **\private\var\mobile\Health\healthdb.sqlite** This database contains several tables that can help an investigator determine whether an Apple Watch was paired and authorized. Two tables, authorization and source_devices, will list the devices, operating system versions, and dates and times of devices using the health functions.
- **\private\var\mobile\Health\healthdb_secure.sqlite** This database file contains several tables with additional settings and user information. The table key_value_secure contains the user's birthdate (birthdate), name (NameKey), sex (sex), and body composition (height, body_mass).

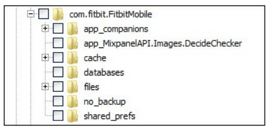
The data that is created during the use of the Apple Watch can be valuable to an investigation. Whether this information assists the investigator in understanding what additional information is in an iTunes or iCloud backup, where it is located, or what was installed on the Apple Watch, it can provide additional clues or artifacts that may help in a forensic case.

Fitbit

Like the Apple Watch, the Fitbit device can be used to receive and send messages, receive alerts from applications, and keep track of the user's health. However, unlike the Apple Watch, the Fitbit's primary function is for health purposes; only recently has the device required an accompanying GPS-enabled device to properly document location and activity. The Fitbit has an app for both iOS and Android and also stores data in the Cloud; both can be sources of information. The app includes meaningful data such as account, health, and location information, to name a few.

Accessing the data on the mobile device is difficult, as the Android or iOS device requires root or physical access. In both iOS and Android operating systems, the data is not available with just a basic file system backup, as is obtained using the Apple Mobile Sync or ADB backup protocols. The app was developed not to back up that valuable investigative data, for obvious privacy concerns. What has been found, however, is that data that is available is stored in SQLite databases and files compressed using gzip.

The file system (Figure 16-15) of the Fitbit app package is similar to that of many other Android apps and can be viewed by using a reverse DNS name of com.fitbit.FitbitMobile with a cache folder, download folder, files, no_backup, and shared_prefs. First, the application will be analyzed to see what type of "tip of the iceberg data" can be uncovered, and then the investigation can move to the cloud resource, where additional historical data may be hiding.



```
com.fitbit.FitbitMobile
    app_companions
    app_MixpanelAPI.Images.DecideChecker
    cache
    databases
    files
    no_backup
    shared_prefs
```

FIGURE 16-15 **Text files in the httpcache folder contain valuable information to identify the data within the associated data file.**

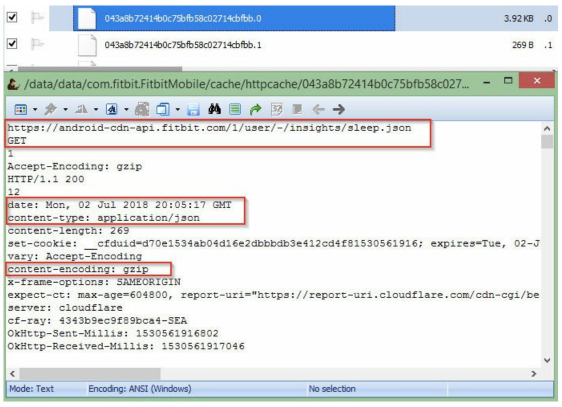
com.fitbit.FitbitMobile

As mentioned, the folder structure for the app is similar to that of other Android apps that have been referenced throughout the book. In this section, the data that is available will be exposed in an effort to help the investigator gather intelligence when the Fitbit app is part of the investigation or when it may corroborate other data recovered.

The first folder, app_companions, contains JavaScript companion files that have been shown to hold no investigative data but that are important for the function of the running processes. Another folder, app_MixpanelAPI.Images.DecideChecker, will generally be next in the folder structure; it contains an image of the device that Fitbit has been set up to use.

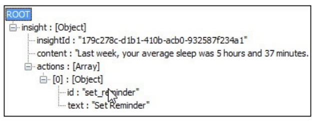
The cache folder holds a tremendous amount of information, particularly the httpcache folder. This folder is where synced data is stored, from the Fitbit to the Cloud and from Fitbit to the mobile device. The mobile device then submits this data to the user's account or receives it in the form of GET or PUT statements using the Fitbit API. This data includes not only app settings but also health settings. The files in this folder contain both text files and data files. Each data file has a unique filename with a *1* extension, and most have an associated text file, with the same name, with a *0* extension. The text file identifies the content, path to the data, and format of the associated data file.

Figure 16-16 shows the information that can be gleaned from the text file, indicating the path on the server via the API. The content of the associated file is in JSON format and in a compressed gzip file. The file was exported, and 7-Zip was used to uncompress it, while Notepad++ was used to format the JSON content. As you can see from Figure 16-17, the user needs more sleep!


043a8b72414b0c75bfb58c02714cbfbb.0 3.92 KB .0
043a8b72414b0c75bfb58c02714cbfbb.1 269 B .1
/data/data/com.fitbit.FitbitMobile/cache/httpcache/043a8b72414b0c75bfb58c027...
Mode: Text
Encoding: ANSI (Windows)
No selection

```
https://android-cdn-api.fitbit.com/1/user/-/insights/sleep.json
GET
1
Accept-Encoding: gzip
HTTP/1.1 200
12
date: Mon, 02 Jul 2018 20:05:17 GMT
content-type: application/json
content-length: 269
set-cookie: __cfduid=d70e1534ab04d16e2dbbbdb3e412cd4f81530561916; expires=Tue, 02-J
vary: Accept-Encoding
content-encoding: gzip
x-frame-options: SAMEORIGIN
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/be
server: cloudflare
cf-ray: 4343b9ec9f89bca4-SEA
OkHttp-Sent-Millis: 1530561916802
OkHttp-Received-Millis: 1530561917046
```

FIGURE 16-16 **The data file is uncompressed and the JSON file viewed to locate valuable content.**



```
ROOT
insight : [Object]
    insightId : "179c278c-d1b1-410b-acb0-932587f234a1"
    content : "Last week, your average sleep was 5 hours and 37 minutes.
    actions : [Array]
        [0] : [Object]
            id : "set_reminder"
            text : "Set Reminder"
```

FIGURE 16-17 **Parsed JSON file for sleep records recovered**

The database folder contains all of the SQLite databases that one would expect in a mobile app. These databases are outlined with an overview of the data that can be obtained from each. This is not an exhaustive list of all the databases in the Fitbit app, but more of a guideline of places to look for pertinent data.

- **\data\data\com.fitbit.FitbitMobile\databases\activity_db** This SQLite database will hold tables ACTIVITY_ITEM, ACTIVITY_LOG_ENTRY, MOST_RECENT_ACTIVITY, and others. The three mentioned tables are the most important because MOST_RECENT_ACTIVITY and ACTIVITY_LOG_ENTRY contain date and time information.
- **\data\data\com.fitbit.FitbitMobile\databases\com.fitbit.FitbitMobile.devmetrics** This file reports on the setup, particularly the setup of the device the app is running on, the Wi-Fi SSID, and TimeZone offset from GMT within the STORED_FSC_EVENT table.
- **\data\data\com.fitbit.FitbitMobile\databases\coreux.db** This SQLite file represents the built-in notifications for the Fitbit.
- **\data\data\com.fitbit.FitbitMobile\databases\exercise_db** This file records the exercises of the wearer. The data that is contained in EXERCISE_SESSION and EXERCISE_EVENT can be used when looking for event times, since date and times are logged in these tables.
- **\data\data\com.fitbit.FitbitMobile\databases\fitbit-db** This is the main SQLite database that contains the user information in the table PROFILE. This table contains the FULL_NAME, TIME_CREATED, TIME_UPDATED, DATE_OF_BIRTH (Epoch UNIX (ms), HEIGHT, GENDER, TIME_ZONE, and many others. Also in this database are several other tables, including ALARMS and DEVICE, which outlines date and time information of the last sync and the type of the device.
- **\data\data\com.fitbit.FitbitMobile\databases\heart_rate_db** This file reports on the daily average in the HEART_RATE_DAILY_SUMMARY table and can be JOINed with table HEART_RATE_ZONE

to show the high and low values for the dates and times in the first table. This information can be important, especially when working cases to determine time of death when a user is wearing the Fitbit.

- **\data\data\com.fitbit.FitbitMobile\databases\sleep** The Fitbit tracks the user's sleep patterns. Within this database, the information on sleep patterns include dates and times, durations, and types of sleep.
- **\data\data\com.fitbit.FitbitMobile\files\User\<*USERID*>\companion_data\platform.db** The Fitbit has several built-in apps, but the user can also add apps to the device, referred to as *companions*. Using this database, an investigator can uncover what apps are installed on the device. Within the companion table of the database is a listing of the apps along with the installation path, date modified, and the app name.

As discussed, an investigator can find data within the Fitbit app to help during an investigation, but the device must be collected at a physical level to expose the verbose information described earlier. However, the data that is sitting on the mobile device is often duplicated within the user's Fitbit cloud account. The following section will describe collecting this type of information.

Fitbit Cloud

Like all the other IoT devices mentioned in this chapter, data for the Fitbit device is often just a username and password away. This mobile device communicates with the cloud sever using the Fitbit API. This is how commercial tools are able to collect and extract the information. However, the account will need to be authenticated by the investigator to extract the information successfully. Some tools, such as Cellebrite Cloud Analyzer, do not necessarily support Fitbit, but they do allow for the capture of the information from the front-facing user dashboard. Currently no commercial tools support the Fitbit Cloud. The method to obtaining the information from the Fitbit Cloud is very similar to obtaining Alexa Cloud data with the use of GET calls in a secure HTTPS instance. What is quite interesting is that Fitbit appears to have secured the extraction of this process with the utilization of a token process that differs from the authentication process of the Alexa Cloud. The information that can be received includes, but is not limited to, the following:

- Profile
- Friends
- Location and GPS
- Weight
- Device information and settings
- Sleep
- Food and water logs
- Activity and exercise
- Heart rate

The API information can be located at https://dev.fitbit.com, which an investigator can build on if a collection from the Fitbit cloud source is necessary. As mentioned, manually collecting this information is tedious and should be considered an advanced technique. However, the steps will be clearly documented here.

First, the investigator must visit the web page https://dev.fitbit.com/build/reference/web-api/oauth2/#obtaining-consent and create an account. This page documents the steps in obtaining an access token (OAuth2.0) that will allow the use of a secondary page to run API calls and retrieve data from the user's account.

The user ID from the Fitbit app will be needed for this process. The user ID can easily be located in the file tree at \data\data\com.fitbit.FitbitMobile\files\User\<*USERID*>. When creating the app, the investigator should elect READ-ONLY. This will allow only `GET` commands and not `POST` commands, helping to keep the integrity of the data within the user's cloud store.

Once the testing app is created using this ID, the investigator selects all options for data on the Fitbit permissions screen. An OAuth 2.0 token will be granted to the investigator, who can move to the next stage using the online Fitbit testing tool at https://dev.fitbit.com/build/reference/web-api/explore/#/.

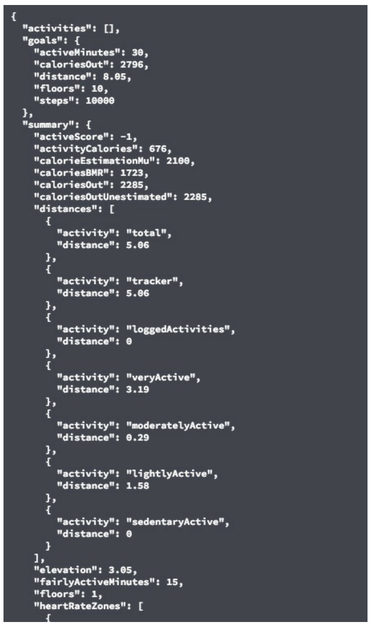
This online tool will allow for the query of data from the Fitbit server with the USERID that can be used to create the testing app and OAuth2.0. The investigator must be logged into https://dev.fitbit.com to utilize the functions and receive a JSON return. Several API calls are provided on the page, including profile, friends, location and GPS, weight, device information and settings, sleep, food and water logs, activity and exercise, and heart rate. For example, all API calls are prefixed with *https://api.fitbit.com/1/user/*<USERID> and then the data requested to extract.

Recall that the httpcache folder contains numerous text files that identify a data file obtaining data from the server. In the text file, the URL will contain a hyphen (-) instead of the USERID. This hyphen indicates that the data will come from the currently logged in user. These URLs can be used to query data from the server to obtain the data viewed on the mobile device. When using the Fitbit development page, the investigator will also use a hyphen (-) since he or she is logged in and using the target USERID.

Table 16-5 lists some of the commands and the expected results.

TABLE 16-5 **GET Commands Using the Fitbit API to Retrieve User Information**

GET URL	Data Returned
/1/user/-/activities/date/<*date*>.json	Date format is *YYYY-MM-DD*, and the user's activity for that date will be returned (Figure 16-18)
1/user/-/activities.json	Like the `GET` command, but will recover the lifetime stats of the user; this can be a large file
/1/user/-/devices.json	Lists all Fitbit devices attached to user account
/1/user/-/devices/tracker/{tracker-id}/alarms.json	For the tracker ID use the `GET` command and insert the ID for the device; lists the alarms that are set for the device
/1/user/-/friends.json	Listing of the user's friends from within the Fitbit app
/1/user/-/friends/invitations.json	All friend invitations sent by the user
/1/user/-/sleep/lists.json	Lists of naps, sleep cycles, between dates set with parameters
/1/user/-/profile.json	User profile
/1/user/-/activities/heart/date/<*date*>/<*period*>.json	User's heartrate; <*date*> field is the end date of the specified period field, and <*period*> field is designated as 1d, 7d, 30d, 1w, 1m



```
{
  "activities": [],
  "goals": {
    "activeMinutes": 30,
    "caloriesOut": 2796,
    "distance": 8.05,
    "floors": 10,
    "steps": 10000
  },
  "summary": {
    "activeScore": -1,
    "activityCalories": 676,
    "calorieEstimationMu": 2100,
    "caloriesBMR": 1723,
    "caloriesOut": 2285,
    "caloriesOutUnestimated": 2285,
    "distances": [
      {
        "activity": "total",
        "distance": 5.06
      },
      {
        "activity": "tracker",
        "distance": 5.06
      },
      {
        "activity": "loggedActivities",
        "distance": 0
      },
      {
        "activity": "veryActive",
        "distance": 3.19
      },
      {
        "activity": "moderatelyActive",
        "distance": 0.29
      },
      {
        "activity": "lightlyActive",
        "distance": 1.58
      },
      {
        "activity": "sedentaryActive",
        "distance": 0
      }
    ],
    "elevation": 3.05,
    "fairlyActiveMinutes": 15,
    "floors": 1,
    "heartRateZones": [
      {
```

FIGURE 16-18 **User activity for the Fitbit user**

Currently, the API `GET` commands listed enable commercial tools to extract data from the user cloud service. As mentioned, the Fitbit service does contain authentication hurdles that an investigator must overcome if the data is going to be extracted. However, like the Apple Watch data, the Fitbit data that can be obtained can bolster any digital forensic investigation.

Unmanned Aircraft Systems

In Chapter 3, drones were discussed along with their impact on several types of investigations. Drones are used to deliver prison contraband and drugs, to stalk victims, and to support terrorist efforts. These devices are easily available to anyone who wants to become a UAS pilot.

When the word *drone* is brought up in conversation, some people immediately picture the military drone, soaring above the earth on a secret mission. This chapter does not discuss the data within those types of drones, but focuses on data that can be collected from recreational drones that can be purchased from any electronics reseller. These devices have become a real problem for law enforcement, and tools have been developed to thwart the flying of these devices in restricted areas. However, continued illegal operations and activities will not cease even with certain countermeasures in place. An investigator must be prepared to tackle these devices if a drone is encountered in the lab.

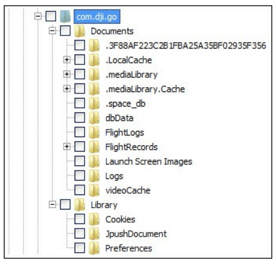
This section will cover "tip of the iceberg data" to help an investigator establish a starting point when dealing with drones, particularly drones created by Dà-Jiāng Innovations (DJI). Because DJI is one of the largest and most prolific drone manufacturers, an investigator is more likely to encounter these recreational drones in an investigation than others. Data from these devices can be obtained from several locations, including the controller, the mobile app, the actual drone, the associated media card, and the associated cloud storage. An investigator will have to tackle several different angles, because the data from each is often unique. It is important that the investigator have a general understanding of the types of data that can be stored in the various locations. In the following sections, each area will be discussed and an outline of the data from each will be listed.

Mobile App: DJI GO

DJI uses its own mobile app, in two versions: DJI GO 4 and DJI GO. DJI GO 4 serves the Phantom 4 and newer devices, while DJI GO is used for devices that preceded the Phantom 4 series. Other apps specific to the DJI drone can also be accessed in an investigation. The non-DJI brands that may be located on a mobile device can include the following:

- **Litchi** Mavic Pro, Phantom 3 and 4, Inspire 1 and 2, Spark
- **Airmap** Mavic Pro, Phantom 3 and 4, Inspire 1 and 2, Spark
- **UgCS for DJI** Mavic Pro, Phantom 3 and 4, Inspire 1 and 2, Matrice

This section covers the DJI GO 4.0 in an effort to inform the investigator on the type of data that can be recovered from within the application. DJI GO contains a similar structure, and the examples discussed here can also be utilized. Both iOS and Android devices can be used as controllers using the DJI GO application. In an investigation, information such as flight logs, images, videos, account information, and flight information may provide critical data. Figure 16-19 shows the file system view of the com.dji.go app (com.dji.pilot for DJI GO app) that can be recovered from a standard iTunes backup extraction. An Android device will contain comparative data, and root access to the device is generally needed to recover the amount of records available in the iOS app. The Android app package is dji.go.v4, and some of the path information to data is covered next.



```
com.dji.go
  Documents
    .3F88AF223C2B1FBA25A35BF02935F356
    .LocalCache
    .mediaLibrary
    .mediaLibrary.Cache
    .space_db
    dbData
    FlightLogs
    FlightRecords
    Launch Screen Images
    Logs
    videoCache
  Library
    Cookies
    JpushDocument
    Preferences
```

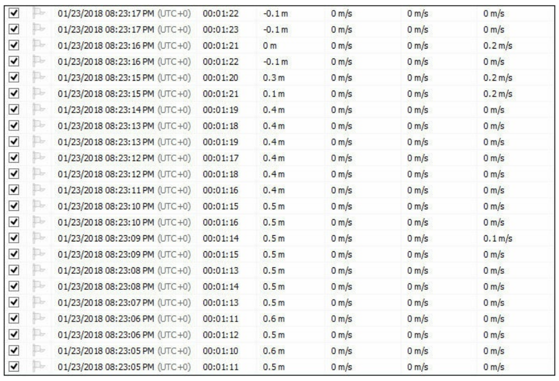
FIGURE 16-19 **DJI GO 4 app file structure from an iOS device**

The recoverable data that can help with an investigation is found throughout the folder structure. Some notable locations are shown in the following list. This information is by no means an exhaustive list, but these locations hold user data relative to the drone and its flight.

- **\private\var\mobile\Applications\com.dji.go\Documents\.LocalCache\agps\agps_info\agps** This file contains a binary plist file that contains a key, time_since_1970, that is a UNIX date for the time the app was launched and obtained a signal. This file also contains location keys, lat and lng. If location services are not provided to the app, the lat and lng will be set to 0.
- **\private\var\mobile\Applications\com.dji.go\Documents\.mediaLibrary\djifile\** This folder contains file names by GUID. These are plist files that contain the metadata for the image files found in the .mediaLibrary.Cache Screennail and Thumbnail folders. Metadata includes the device used to produce the image as well as the created time.
- **\private\var\mobile\Applications\com.dji.go\Documents\.mediaLibrary.Cache** This path contains two folders, Screennail and Thumbnail. These folders contain the full images and thumbnails cached to the app either because the SD card was not used for storage, was default to the app, or was not inserted. All filenames are GUIDs, and the associated metadata can be found under the matching GUID in the djifile folder.
- **\private\var\mobile\Applications\com.dji.go\Documents\videoCache\** This folder contains MP4 files of flights as well as the associated thumbnails stored to the app from the device. The filenames are the date and time of the recording (*YYYY-MM-DD-HH-MM-SS*).
- **\private\var\mobile\Applications\com.dji.go\Documents\.3F88AF223C2B1FBA25A35BF02935F356** JSON file will be named with the registered user's e-mail.
- **\private\var\mobile\Applications\com.dji.go\Library\Preferences\com.dji.go.plist** This property list file contains keys and values that identify milestones for the device. Values such as the total flight time,

maximum height, user e-mail address, date user ID created, flight records with date/time, along with the country in which the flight took place.

- **\private\var\mobile\Applications\com.dji.go\Documents\FlightLogs** This folder contains flight log .dat files that are prefixed with a date *YYYY-MM-DD*. This .dat file is a formatted text file that contains take-off dates, home point recorded date with altitude, and obstacles avoided with date and time.
- **private\var\mobile\Applications\com.dji.go\Documents\FlightRecords** This folder contains a file prefixed with DJIFlightRecord_<*Date*> with a .txt extension. This encrypted file contains the data records for the flight(s). The data contained within the text file includes the device and serial numbers along with GPS location, time, speed (meters per second), number of satellites used, altitude (barometric pressure and displays in meters above sea level), direction, power, and movement (meters per second) along the X, Y, and Z axes. The X axis is front to back, Y is side to side, and Z is up and down. This information can help the examiner determine flight path at a granular level. Note that the air pressure is set initially on preflight and helps with hovering and auto-landing. Some commercial tools such as Oxygen Forensic Detective parse and display this data for investigations (Figure 16-20). Also, the encrypted text files can be uploaded to online services such as www.phantomhelp.com/LogViewer/Upload/, which will also parse the information.



✓		01/23/2018 08:23:17 PM (UTC+0)	00:01:22	-0.1 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:17 PM (UTC+0)	00:01:23	-0.1 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:16 PM (UTC+0)	00:01:21	0 m	0 m/s	0 m/s	0.2 m/s
✓		01/23/2018 08:23:16 PM (UTC+0)	00:01:22	-0.1 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:15 PM (UTC+0)	00:01:20	0.3 m	0 m/s	0 m/s	0.2 m/s
✓		01/23/2018 08:23:15 PM (UTC+0)	00:01:21	0.1 m	0 m/s	0 m/s	0.2 m/s
✓		01/23/2018 08:23:14 PM (UTC+0)	00:01:19	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:13 PM (UTC+0)	00:01:18	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:13 PM (UTC+0)	00:01:19	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:12 PM (UTC+0)	00:01:17	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:12 PM (UTC+0)	00:01:18	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:11 PM (UTC+0)	00:01:16	0.4 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:10 PM (UTC+0)	00:01:15	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:10 PM (UTC+0)	00:01:16	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:09 PM (UTC+0)	00:01:14	0.5 m	0 m/s	0 m/s	0.1 m/s
✓		01/23/2018 08:23:09 PM (UTC+0)	00:01:15	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:08 PM (UTC+0)	00:01:13	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:08 PM (UTC+0)	00:01:14	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:07 PM (UTC+0)	00:01:13	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:06 PM (UTC+0)	00:01:11	0.6 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:06 PM (UTC+0)	00:01:12	0.5 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:05 PM (UTC+0)	00:01:10	0.6 m	0 m/s	0 m/s	0 m/s
✓		01/23/2018 08:23:05 PM (UTC+0)	00:01:11	0.5 m	0 m/s	0 m/s	0 m/s

FIGURE 16-20 **Flight information displayed in Oxygen Forensic Detective**

Some versions of the app also have a folder called MCDatFlightRecords that contains a .DAT file that can be found on the SD card in the drone if inserted. This file contains fewer reading points than are in the associated encrypted text file, and it is not compressed or encrypted. The smaller file does not change the validity of the GPS data; it shows data generally every second of the drone flight.

Note

From the DJI SDK documentation (https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html) the X, Y, Z concepts are described as "aircraft translation in positive X, Y, and Z is therefore defined in the Body Coordinate System as forward, right and downward translation respectively. Aircraft rotation is also described with these same axes using the coordinate right hand rule to define the direction of positive rotation. When describing rotational movement, the X, Y, and Z axes are renamed Roll, Pitch, and Yaw."

For Android device apps and the dji.go.v4 app in a physical collection, the information and path to some important data are found within the media path \data\media\0\DJI\dji.go.v4\. This location contains several folders that hold the cached imagery files as well as the log files for flights.

- **\data\media\0\DJI\dji.go.v4\CACHE_IMAGE** This folder contains thumbnail images. Since most Android devices can have SD cards, the main images will be stored on the external media.
- **\data\media\0\DJI\dji.go.v4\DJI_RECORD** This folder contains MP4 files of flights as well as a matching .info file that lists the metadata for the video to include GPS information, device type, and date and time.
- **\data\media\0\DJI\dji.go.v4\FlightRecord** This folder contains the encrypted .txt flight logs.
- **\data\media\0\DJI\dji.go.v4\LOG\MAP** This folder contains log files with a date and time. Many of the log files will contain a value addHomeMarker, which provides the GPS coordinates and can be valuable to the investigation.
- **\data\data\dji.go.v4\databases\dji.db** This SQLite database contains all of the devices and their serial numbers and the user's registered e-mail.

By using multiple files from the app that controls the drone, an investigator can gain good information on the flights for the drone along with imagery captured and account information. Because multiple apps can be used to control and capture the data from the drone, not just the manufacturer app, the investigator must be aware of the currently available apps. The flight logs that can be extracted from the associated app can also be parsed online as well as using commercial tools supporting the data format.

Physical Acquisition

Drones capture information and store the data to non-volatile flash media. Like mobile devices, the drone's data can be captured physically by invasive and non-invasive methods. Invasive methods include the removal of the chip from the device and reading in the associated reader/programmer; non-invasive physical collection includes obtaining root access via USB or another method and copying the user/system data. The chip-off procedure can yield a pristine copy of the data that had been recorded during flight(s). Because the device does not have to be powered on to receive GPS coordinates, several different flights and multiple registered home locations can be recovered. To perform the non-invasive physical collection, the device needs to be powered on, and this can cause the device to obtain a GPS lock. DJI protocol will then zero-out .txt and .dat files from the device. Note that the .txt files that had been logged to the mobile com.dji.pilot application can still remain, however. While the chip-off procedure does have advantages, the investigator should understand the circumstances of the case. Removal of the chip is destructive, and the drone will be inoperable; a non-invasive physical collection will not destroy the drone.

The data observable in a chip-off extraction—at the time of writing this manuscript, the entire blackbox folder—contains only incremented flight data as compared to the robust flight data, .dat files, and associated vision folder in the USB physical collection described in the following section (see Figure 16-21). This could result from a change in firmware between the time of chip-off extraction to the current USB extraction, which was approximately four months. An investigator can use the following non-invasive landmark sections for the investigation of a chip-off since the paths have been shown to be almost identical.

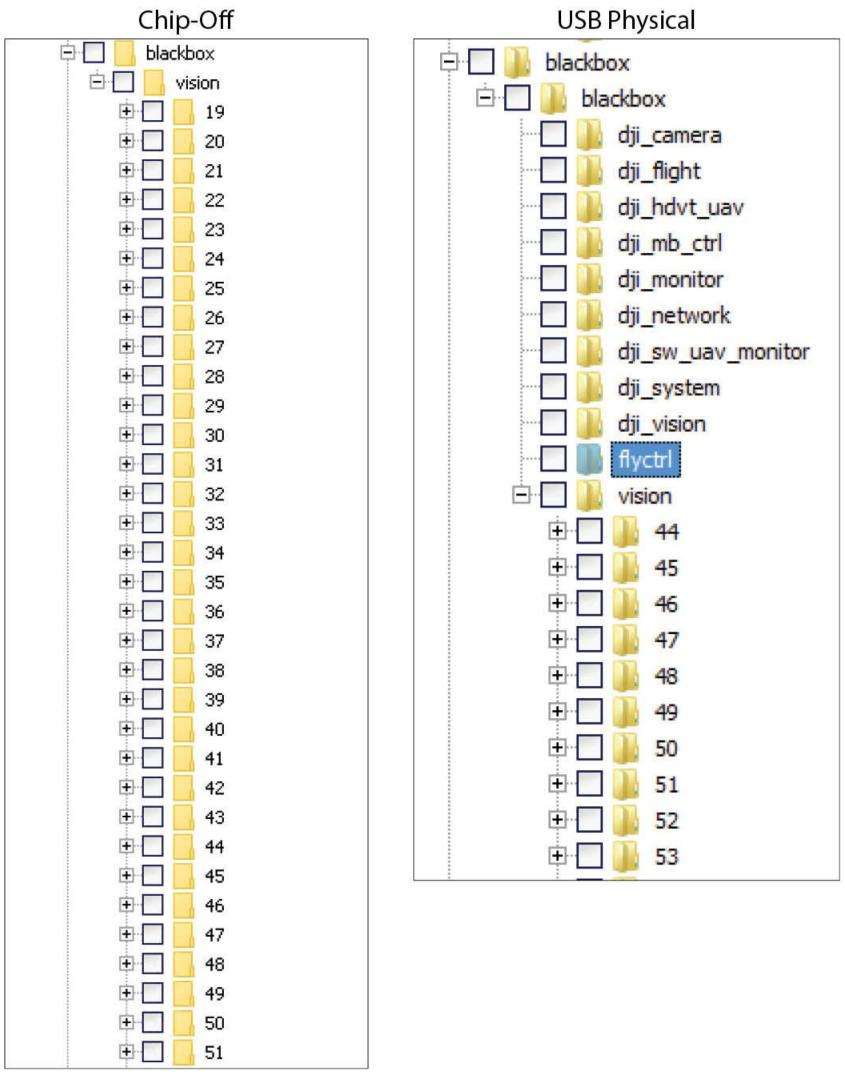


FIGURE 16-21 **Primary folders in chip-off extraction are not observed, versus a USB physical collection.**

Non-invasive physical collection offers many benefits, being a non-destructive means of extraction using a USB cable and Android root exploit. DJI, and many drone manufacturers, continue to upgrade the firmware on the devices in an effort to update the no-fly zones and to conform to changing legal requirements on recreational flying. However, the Android operating system that is still being utilized, often 4x, allows for the extraction of the user data with admin privileges. Note that DJI controllers run almost a duplicated OS version along with the firmware. This also allows for the physical extraction of the data from the device, using the same methods currently employed only (at the time of writing) within Oxygen Forensic Detective.

The user data folder structure is almost identical to that of the invasive chip-off technique, with the only limitation being that some files are zeroed out by the startup and acquisition of a location fix. Some working in the community of drone examiners advise that data is not available using extraction via USB and only via chip-off, but it seems that DJI devices simply increment the flight data file upon a new flight and remove the old data because of limited space on the 4GB onboard flash. Often the drone investigation comes to the lab without a mobile device or controller, and the drone will be the only evidence collected, so it is imperative that the examiner understand where some of the information can be located in an extraction with a USB.

- **\amt\amt\wifi.config** This file contains the SSID of the drone as well as the passcode in clear text.
- **\amt\amt\BT\BT_Address.txt** This file displays the Bluetooth address of the device.
- **\amt\amt\nfz\nfx.db** This SQLite file contains all of the current latitude and longitude information for the no-fly zone that is often replicated in the mobile app. The table airmap_geofence_polygons outlines the country number as well as the radius for the geofence. As an investigator, this information can assist when dealing with no-fly zones to determine whether the device had the proper information about the restricted area.
- **\amt\amt\npi_test\sn_dut** This text file contains the serial number of the drone.
- **\blackbox\blackbox** Much like the blackbox on an airplane, drones have a controller box that records information from all aspects of the device to .bb files. Information from the camera, the network, the controller, the flight system (vision), and all other running processes are stored in these folders. Generally, files are zeroed out by the DJI system with all containing a latest file that shows the last file that was being used to record flight data and calibrations.
- **\blackbox\blackbox\flyctrl** This folder contains all of the .DAT files of the logged flights and the contents described in the previous mobile app section. The files are prefixed with the date with FLY<*n*>.DAT and the number is incremented. These flight files are encrypted, but they can be decrypted in most commercial tools.
- **userdata\data\misc\wifi\hostapd.conf** This file contains the Wi-Fi SSID as well as the country this Wi-Fi was obtained in, and the password is in clear text.
- **userdata\data\misc\wifi\wpa_supplicant.conf** This file contains the Wi-Fi SSID, and the password is in clear text.

The information that can be obtained from a physical extraction can help to supplement data recovered from the mobile device that had been used to control the drone. Also, if the mobile device is not available, important data, including flight logs, Wi-Fi, country where the flight took place, and user data, is recoverable. Remember that avoiding the acquisition of signals (Wi-Fi, Bluetooth, GPS) is important if the recovery takes place via the USB port. Likewise, recall of the DJI drones will zero-out files included in the blackbox as well as flight records. However, I have been able to recover multiple flights both from the mobile device (more probable) and drone device.

Media Card

Media cards, generally miniSD cards, are used by the drone to store flight logs as well as imagery. An SD card does not have to be inserted into the drone to function. If an SD card is not inserted, the data is stored directly to the mobile device that has been paired with the drone. If the drone is using a radio control (RC) controller independently, the data will be stored to the drone's internal flash. This section discusses the data structure an investigator should expect to see on an SD card.

The file system on the SD card is formatted as FAT and can be read in tools such as FTK Imager, Oxygen Forensic Detective, Cellebrite UFED4PC, and Magnet AXIOM. As with acquiring data from removable media, the investigator should take all precautions when creating a forensic image of the card (photograph, write block, image, and work on copy). Once the image is created, the folder structure is similar to what would be expected from a digital camera. However, in the root of the file system are the drone's .DAT files. The .DAT files utilize encryption that differs from that of the internal .TXT files or mobile device .DAT files, but the data is consistent between files. These files can be brought into many commercial tools such as Oxygen Forensic Detective, Cellebrite UFED4PC, and MSAB XRY. The data is then overlaid on a map to produce a flight path and route.

Two additional folders are created on the SD card: the DCIM folder and MISC folder. The DCIM folder contains the images and videos captured by the drone's camera. The drone captures and stores EXIF metadata for both videos and images, which can then be used to identify location—a fantastic piece of evidence for the investigation. The following illustration shows the skyline in Singapore taken with a DJI Spark. A great piece of evidence, the SD card and UAS were seized in another country. Within the MISC folder is a secondary folder, THM, which will contain the thumbnails of the files found in the DCIM folder. The prefix of the THM file will correspond to the media file within the media folder.

Data on the SD card should not be overlooked, because it will contain information that may not be found within the mobile device app or even within the drone itself.

Cloud Services

Drone storage, like storage on many other devices, has moved to the Cloud. The information transitions from the mobile device, as a controller, and into the appropriate cloud service. This storage location can provide a treasure trove of flights, account information, and imagery. The user must subscribe to this service and is not obligated to sign up to use the Cloud for the drone to be operational. However, as an investigator, the knowledge of another source of data is paramount, especially if the drone, controller, or both are not available.

Users of DJI devices have the opportunity to sign up to use the DJI Cloud via the DJI GO app, where they can store their profile, device info, media, and flight logs. Users can also upload the information to SkyPixel to share their videos with the community. The uploading of the data is not automatic for the user.

The user must set up the account and elect to sync the logs to the Cloud. Currently, Oxygen Forensic Cloud Extractor is the only commercial tool that can extract profile, devices, media, flight logs, username, password, and token. Look for the following in the extraction; these files are relative only to the extracted data with the Oxygen Forensic Cloud Extractor.

- **\cloud.dji.data\<*UserEmail*>\DroneFlights.json** This JSON file contains the general flight data for a single event for each device by serial number. The serial number of the drone is listed, along with longitude and latitude, total distance, flight time, and a date and time.
- **\cloud.dji.data\<*UserEmail*>\Main.db** This SQLite database is a gold mine of user data and drone data. Within the database are three tables: account_info, dbver, and drones_info. In account_info, the investigator will find the e-mail address, password, token, username, date signed up, and physical address (if entered). Within the table dbver is an integer indicating the database version. Table drones_info contains anything and everything pertaining to each drone using the account. This information includes but is not limited to serial number, the countries the devices has flown in, the drone type and name, and totals on flight time, distance, speeds (m/s), and altitudes (m). All are the maximum numbers from all flights, but also shown are the dates the maximum was recorded.
- **\cloud.dji.data\<*UserEmail*>\Avatars\account.jpg** This is the image file the user uploaded to the DJI Cloud account.
- **\cloud.dji.data\<*UserEmail*>\RawLogs\*DJIFlightRecord*<*Date*>.txt** These text files contain flight data much like the records found on the mobile device controller and the onboard flash for the drone. These text files use a form of encryption that differs from that of the mobile device and onboard flash, and they are supported by tools such as Oxygen Forensic Detective. What is extremely important to understand is the fact that logs in the cloud can include every flight the user has taken, along with images. This data is not zeroed out like the onboard flash or mobile app, making this repository extremely important and valuable to the investigation. Figure 16-22 shows flight paths from DJI cloud extraction.

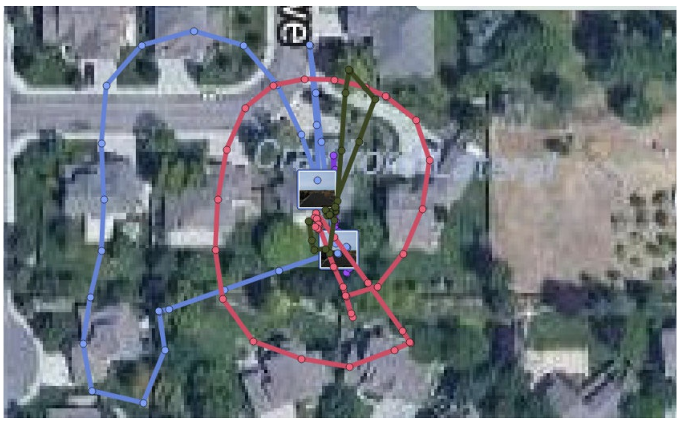


FIGURE 16-22 **Flight paths produced with data from DJI cloud extraction overlaid in Oxygen Forensic Maps**

The information that can be obtained from the drone's user storage cloud service cannot only complement the data that has been extracted from the device but also can open the door to additional data that could not be recovered or is not accessible from the controller, mobile app, and even the drone. The information within cloud services should never be discounted.

Chapter Summary

From IoT devices in the home, to wearable devices, to recreational drones available in any electronics store, today's investigator must be prepared to handle this emerging trend and the extraction of data from these state-of-the-art devices. Some may want to use the word "threats" instead of "trends," but that is not an accurate statement. These devices can simplify the lives of users, help to notify people of health problems, and protect the public safety. However, as with any device, some users will push the limits, or rather use the limits, to satisfy a nefarious purpose to their gain. IoT forensics is not much different from the infancy stages of mobile device forensics. At that time, classes were being taught on how to take pictures of the data on the device in an effort to document the information, or in utilizing tools that sync software, since there were no "forensic" tools back then. This is where digital investigations of IoT devices must rise above the days of Wild West forensics. Nevertheless, the industry—and both commercial software and forensics examiners—is not there yet.

The collection of data from these device types is all but simple point-and-click. Acquiring data from these devices can be extremely time consuming, if not frustrating. As indicated throughout the chapter, extracting the information can involve several methods as well as several locations in which the data may reside (such as mobile device, Cloud, controller). Autonomous extraction and parsing of IoT devices to include wearables, vehicles, and drones is in the infancy stages of development by commercial tools. It will be up to the investigators and examiners to work through the weeds to develop strategies to attack these new data repositories.

IoT devices such as the Amazon Echo series using Alexa and Google Home will only grow in popularity. There have already been documented cases of this data being used, or rather attempted to be used, in civil and criminal cases. With most, if not the majority, of the available home smart speakers, TVs, and other connected devices, the data is available within a cloud service dedicated to a user. If this was not the case, the user would not be able to log in using a mobile device to check the status of the contents of the refrigerator or to access the front door. It is simply a matter of investigation and observation.

With wearable devices, data is also available within the Cloud, and within the produced backup in the case of the Apple Watch. Because this is such a new frontier, however, the extraction of devices such as the Fitbit is tedious work, since currently a lot of manual work is involved using the published API. With wearables, from clothes to eyewear, the data is available—it just comes down to the location of the repository, the method used to access data, and ultimately making sense of the output. Sounds easy. Unfortunately, as outlined in this chapter, it will continue to be a lot of work for the foreseeable future.

The number of drones available to anyone in the world continues to grow daily. Coupled with easy availability and a plethora of features, the use of these once recreational devices has shifted to assist in devious behaviors. Stalking, terrorism, drug delivery, prison contraband delivery, border crossing, and national security breaches have been attributed to alternative use of drones. The astonishing fact through investigation indicates that the interrogation of these devices by digital investigators is almost non-existent outside of military investigations. With the number of recreational drones in use anticipated to reach more 450,000 devices by 2022 and 300,000 commercial drones by 2022, as reported by the Federal Aviation Administration, it is a certainty that these devices will be in every investigator's lab.

The data presented in this chapter has been grown from research, communication with industry experts, and a lot of work. Simply said, there is not a lot of documentation on this type of data collection anywhere in the world, and within this chapter, some information has never been documented and discovered until now. To those experts mentioned in the chapter: thank you for your diligence and hard work. To all investigators: follow-up with your documentation to gain a better understanding, rather than relying solely on what was presented in this chapter.

To sum up the information discussed in this chapter, there is an unbelievable wave of data and data sources that are coming and that are currently here. Whether you are a seasoned digital forensic guru or a green investigator, the data collected from these devices will be invaluable in painting a total picture of our digital world. Be prepared.

17 Presenting the Data as a Mobile Forensics Expert

Making a case after collecting a significant amount of data from a mobile device examination is only as good as the documentation. If the entire process, from seizure to data analysis, is not properly documented or explained in a way that's appropriate for the audience, then a comprehensive and proper exam is often pointless.

Documentation is an examiner's most important tool, especially when many different exhibits, methods, or custodians are involved. Without proper documentation, an examiner will be relying on memory or only the extracted and analyzed data. Recalling details from even a day prior is often difficult for most, let alone recalling every detail from the start of a process that may have taken weeks or even months to complete. An examiner must take detailed notes during the entire process. Notetaking at the onset and throughout the entire process is mandatory.

Before beginning the formal documentation process, an examiner should be aware of who will be reading the documentation. Often, if the evidence seems too complex or technical, the intended audience will not comprehend the message, and if they cannot comprehend the message, the examiner's work will be pointless, no matter how significant the data. An examiner often neglects the technicality of what he or she has accomplished and regurgitates this technical journey in the documentation, without considering the audience. When the documentation presentation is too technical and is too difficult for the readers or listeners, the information provided may be dismissed. The examiner must find a way to express the details of the examination in a format and using terms that help the audience understand the gravity of the evidence.

An examiner should also be mindful of the data that *should* be a part of the completed report. Some examiners simply dump all the information into a document, place the information on a storage device, and hand it over to the requester/audience. But this often creates additional work for the requester/audience, because the examiner has confused details of the examination with what was actually requested. The documentation should be concise and to the point, but an examiner should always be able to elaborate, if requested, with additional information.

The journey to becoming a mobile forensic expert is long, and it is often misconstrued as gathering certifications, awards, and endorsements. But this is not the true path. Becoming a mobile forensic expert comes from a desire to find the truth within the data. In finding this truth, the examiner must follow processes and procedures, constantly research new technologies, meticulously document the process, and, most important, understand and supplement what the automated solutions are doing to recover mobile device data. The road to becoming an expert is paved with examiners who think mastering mobile forensic tools and achieving certifications are all it takes to get there. By this point, however, every examiner reading this book should recognize that this statement could not be further from the truth.

There is a difference between an expert in the field and an expert who has been qualified as an expert. The examiner should recognize the distinction between individuals who classify themselves as experts in the field by experience and accolades and those who give qualified expert testimony. One is not better than the other, but the examiner must understand the difference. One aspect of being a qualified expert witness requires that the following criteria be met and decided upon by the court—and no one else. In the Federal Rules of Evidence, "Rule 702. Testimony by Expert Witnesses" (www.law.cornell.edu/rules/fre/rule_702), specifically states that expert opinions may be admissible if

(a) the expert's scientific, technical, or other specialized knowledge will help the trier of fact to understand the evidence or to determine a fact in issue;
(b) the testimony is based on sufficient facts or data;
(c) the testimony is the product of reliable principles and methods; and
(d) the expert has reliably applied the principles and methods to the facts of the case.

Even if Rule 702 is not followed, most U.S. states and many nations adhere to this philosophy, with similar requirements. Courts require that the witness be competent on the subject and know the underlying methods and procedures used in the examination and relied upon as a basis on the opinion.

Presenting the Data

When presenting the data, an examiner must consider several factors that are generally based on the audience or individual requesting the work. The format should be established at the onset of the examination so that the examiner can tailor the examination to meet the requirements of the requestor. Determining the presentation format up front saves the examiner time and resources. Failing to comprehend how the data should be formatted from the beginning can result in the examiner following the wrong path, which can result in an exorbitant amount of time and sometimes money to remedy.

An examiner should also understand the target of the examination at the start. The investigation of the mobile device data should provide insight into the investigation, with not only the "tip of the iceberg data" but also a massive amount of relevant data located within the device file system. An examiner should be able to target specific apps for investigation according to the requirements of the investigation, but he or she should also not dismiss any other relevant data uncovered during the investigation.

The relevance of the collected data is determined by the presentation of what data was collected and how. For example, merely stating that a message was sent from Google Voice to another device that received it via an SMS is often not enough—the data should correlate the sent message in the Google Voice app with the data from the receiving device's SMS message container, relating the date/time, time zone, content, and user ID to the relevant factors of the investigation. Furthermore, the examiner must detail and document SMS messages, which can be sent from third-party apps to other mobile devices. Using apps for messaging instead of the carrier SMS conduit occurs frequently. Consequently, the sender using Google Voice can hamper a mobile solution's ability to recover a message from the standard SMS messaging because it was sent from a third-party app. Moreover, since third-party data messaging content is not kept, stored, or cached at the telco server, it is important that the examiner document when content was recovered on the device. Including this type of information within the presentation documentation is critical, especially for data that is significant to the outcome of the case. Outlining the importance of the data as it pertains to the overall case, using common language and including lots of information, is critical.

Unfortunately, however, use of the many available mobile forensic solutions can lead to confusion when it comes to documentation—not only because of the number of varying reporting formats but also because of several different aspects of the documentation. An examiner may complete the processing of the mobile device, select to output the data into a report format, and move on to the next piece of mobile forensic software. The data is processed in the secondary software, or the device is collected again and another report is generated. This secondary report may contain additional data that the first solution did not contain, but it may also duplicate data included in the first software's report. The problem is not in the employment of multiple tools within an examination, but the improper presentation of this material.

No matter whether one or multiple tools are used in a mobile forensic investigation, one report should outline the complete seizure, collection, and analysis, and any reports created by the mobile forensic tools should be used to supplement the main document, because they often do not contain enough detail or commentary.

I can recall several instances when a Cellebrite report comprising thousands of pages was entered as evidence. Legal teams pored over the pages, searching for germane data, with absolutely no idea how the information was obtained, where it was obtained from, or what the data actually meant to the investigation. This issue is not limited to Cellebrite, because practically all mobile forensic solutions ordinarily output generated data compiled by the automated methods that place no value on the process by which it was collected or the requirements of the overall case—that is left up to the examiner. Unfortunately, many examiners do not add information to the solution's automatically created report. But without including important details in the presentation, the audience may find it difficult or impossible to comprehend the meaning behind the data hidden within the many pages. *An expert is not defined by the number of pages generated by an automated tool, but by the explanation of the content that is presented with the supporting tool's output.*

The Importance of Taking Notes

The entire process, from seizure to analysis, should be documented in chronological order in the examiner's notes. The date and time of each function should accompany any details of the event, along with all observations. Using these detailed notes, an examiner can paint a picture of the event precisely how it occurred. An examiner may attempt to recall the entire event during the final documentation phase, but in doing this, he or she may leave out many forgotten but valuable details.

Guessing and filling in the blanks of an examination is never the right choice. By documenting every detail at every step, the examiner will be sure to include even the minutest detail in the final report. Suppose, for example, that an examiner "always" places a device in airplane mode as soon as a device is seized, and he never documents this process because it's always the same. He always writes the report and adds this detail at the end of the examination. However, on this particular case, he's not entirely sure he remembered to place the device in airplane mode because he was interrupted by an important phone call just as he collected the device. Still, because he believes he always follows the same procedures, he writes the report assuming that he followed his usual routine. Another examiner is called to validate the findings of the first examiner and happens to look into the settings file. She determines that the mobile device was *not* put in airplane mode at the time of seizure. The entire first examination is discredited and may be deemed unusable.

Instead of guessing, the examiner can create a Chronology log. This log is a detailed grid similar to Table

17-1, which can be filled out to include the various tasks with clear details for later recall. This is similar to a log of the events, and it will help the examiner immensely when recalling each task and resolution when later writing a report.

TABLE 17-1 **Sample Examiner's Notes Table Used to Document Every Detail of the Investigation, from Reception to Reporting**

Date	Time	Task	Description	Comments
3/3/18	900	Reception	Det. Smith requested an analysis of a mobile device	Device was obtained from the property room and transported to lab. Det. Smith completed processing request form.
3/3/18	945	Open evidence	Mobile device and UICC	Photographed, sealed, and then when the bag was opened—mobile device was off.
3/2/18	1015	Document	UICC and mobile device	Gathered mobile device information and UICC information using photographs.
3/3/18	1100	Collection	SIM	UICC was photographed and collected using Oxygen Forensic Detective. File system and PIN data recovered.
3/3/18	1130	Isolation	Mobile device	Mobile device was isolated from cellular by its lack of UICC. Network was isolated after pressing power button and selecting Airplane Mode while in Faraday box. Device was unlocked.
3/3/18	1300	Collection	Mobile device	Device was collected using non-invasive means physically using UFED. Binary file was produced of the userdata partition.
3/4/18	900	Collection	Mobile device	Device was collected logically using UFED. SMS, MMS, Calls, Contacts.

When completing any notes of the event, the examiner must understand that any documentation produced as part of the investigation is *discoverable*. In other words, the examiner's notes can be requested as part of the proceedings if he or she was not acting as a consulting expert based upon Federal Rule of Evidence 502 and Federal Rule of Civil Procedures 26(b)(A). If the examiner is acting as a consulting expert and will not be testifying but is advising counsel, his or her notes are a work product and protected under attorney-client privilege, understanding that these drafts are legal advice and the attorney rendered legal advice based on the draft. With that being said, the information contained within the examiner's notes should be free from generalizations and opinions and should simply state the facts. The only facts that should be included in the examiner's notes are the procedures taken and the outcomes of completing the procedures. Listing opinions, comments, or even doodles in the margin could result in considerable hurdles during the court proceedings, particularly if the court requests these documents. The notes should be used for an examiner's recollection of the entire process, and they serve as a guide as the examiner creates the final presentation of evidence.

The Audience

The *audience* is any person or group of people who will be exposed to the investigational report created by the examiner. The audience could be the examiner's supervisor, human resources, a company CEO, an attorney, a jury, a judge, or any other entity coming into contact with this compiled information. Knowing the audience of the produced documentation can help guide the examiner during the final documentation phase. The presentation of the materials should be compiled in a form that caters to each group or individual person mentioned, but it should also contain information as it relates to the specific audience that requested the report. For example, a report created for the CEO of a company may contain sensitive information that is not appropriate for an employee of the company. This does not mean the examiner should permanently omit data, but he or she should be selective in providing the information needed and requested for particular audience members. When the examiner creates reports using automated tools, this can be difficult unless some sort of filtering is available. This is an important reason why an automated report should supplement the main documentation: the primary document will contain information specific to the request, outlining the details of the collection and analysis of the scoped data, and the supplemental solution report can be used in a redacted form. Of course, if the requesting party wants to see the information originally omitted, that would not be a problem, because a full collection would have already been completed and a more verbose report created.

Note

A full collection should always be completed unless a full collection exceeds the scope of the request or warrant. In those limited cases, only a portion of the information will be available, as outlined by the requestor or warrant. If additional information is requested outside of scope, an examiner should follow proper procedures in obtaining the right to gather this additional information from the device.

Knowing the audience and creating a pointed report that details the information specific to the audience will resonate more than producing a generic solution report containing all data. Having regard for the viewer and targeting the report based upon his or her request will often mean little follow-up work by the examiner. By giving the audience member the data that was specifically requested, in a form detailing the examination, analysis, and findings, the examiner will satisfy the requirements set forth by the audience. Delivering a 1000-page report that requires the reader to decipher technical details can result in many questions and more work for an examiner after the fact. If an examiner bases the presentation on the audience, outlining and detailing the data to fit the request, he or she is sure to meet the needs of the request.

Format of the Examiner's Presentation

In addition to knowing who the report's audience is, an examiner should know the type of format that is required for the report. The format can be dictated by the working standard operating procedure (SOP) by the company or agency sponsoring the collection and analysis. Most automated tools' reporting systems output reports as Microsoft Word documents, PDFs, HTML documents, CSV documents, and sometimes XML. Because the output produced by these tools will be used to supplement the main document, the format of this report often is not as important to the audience as the main document's format. However, if the audience has requested the solutions output in a format such as CSV or XML for specific reasons, such as to import it into intelligence software such as i2 Analyst's Notebook, the type of supplemental documents provided does significantly matter. This formatting information should be discussed and settled prior to beginning the collection and analysis of the mobile device evidence.

The main document may be created in a word processor such as Microsoft Word or Apple's Pages and then saved and printed as a PDF. Some agencies and companies have internal word processing applications specific to their business; these are completely acceptable to use since most will output to a PDF. A PDF is important for final documentation, because the content within the PDF remains read-only. Of course, if the company or agency format is different based upon an SOP, the SOP should be followed so long as the final output cannot be modified. Some agencies and companies use HTML for their reporting to organize and display their findings. This method can be used as well, but a main report should always be included as a PDF, and the HTML format should be used for navigation purposes only.

Note

It used to be common practice to use HTML with an auto-run function, but because modern operating systems allow this function to be disabled, using this methodology is not advised.

Whatever the format of the completed report and supplemental data, the examiner should organize the information using the following guidelines:

- All information should be placed on storage media (such as CD, DVD, flash drive).
- The information should be categorized within a README file in the root of the storage media that outlines the contents. An example is shown here:

```
Case #12345  Examination of Mobile Device 1234, 1238, 1123
DVD Contents
README.txt - This file
Case12345-MDF.PDF - The main report containing case details, findings
and conclusions
Supplemental Docs - Folder containing output reports from UFED, MPE+
and XRY
Evidence - Folder containing native files exported from mobile forensic
solutions organized in separate folders Images, Audio, Video, Docs,
Misc
```

- If the examiner will be using HTML pages, the main index.html and associated files should be placed into a separate folder named HTML Report and listed in the README.txt file with instructions on its operation.
- If the examiner has placed media files within the evidence folder, the audience might need special software to view the material (such as .mov, .mp4, .qcp files). The viewing software should be included in a folder

called MISC Software and its use and need described in the README.txt file.

When the information is organized by the examiner, the audience is less likely to request an explanation or require assistance from the examiner when viewing the results. Making information clear and concise will save the examiner from unnecessary requests, saving time for both the audience and the examiner. If documents are produced as a hard copy, all pages should be initialed by the examiner to ensure the integrity of the documented data.

Why Being Technical Is Not Always Best

When documenting the process, procedures, and analysis, the examiner should remember to write in a way that the audience will understand. Often, a forensic examiner will create a document that contains information that only another examiner will understand, but this defeats the purpose of the report. An examiner should think in terms of customer satisfaction when creating the main document and any supplementary documents; all should be written and compiled so that a person with absolutely no forensic background or technical experience can visualize and comprehend the contents.

Tip

If the client requests that the examiner create a PowerPoint presentation to explain his or her report, this is a clue that the information contained in the report does not paint an understandable picture of the procedure and data.

Examiners must understand that the report's audience may have an entirely different perspective regarding the data. Using jargon and acronyms known only to mobile forensic examiners can confuse and frustrate a reader who is not versed in these terms. A report can be quickly dismissed if it is simply a well-written document that only the examiner understands. The most frequently requested change clients make when receiving an examiner's final report is to explain the technical jargon used in the document. Some examiners go the opposite route, "dumbing down" the content. This has the same effect on readers, who realize the author is "talking down" to them. The examiner is challenged to find middle ground to discuss technical aspects, procedures, analyses, and conclusions in simple terms, but not so simple as to demean the readers.

An examiner may find that using analogies can help in documenting the technical aspects of the device analysis. By using everyday, real-life examples, an examiner can describe a technical procedure within the document or during testimony by comparing the information to something more familiar. For example, if the examiner includes information about unallocated space, he or she might compare it to a shelf in the public library. The user goes to the catalog system, looks up a book by the author or title, and is directed to a shelf where the book is located. However, not all books are referenced in the catalog system—perhaps they have been removed to discard—yet they still exist on a shelf somewhere in the library. The books may still be read, and they still physically exist, but they cannot be found using the catalog. Unallocated space is like the shelf where the discarded books are stored, and the information stored here is similar to the discarded books. This storage space contains deleted files, file remnants, and empty space. Because the files within the unallocated space are not referenced within a file table (catalog), an examiner must search to find each file within this space.

"Tech speak" is best left to conversations with other forensic examiners, but sometimes technical terms and concepts must be included in a forensics report. In this case, the document should contain a list of terms used in the document and simple definitions. The definitions should explain each concept in an appropriate way so that the audience will understand. An examiner can then reuse a definition document for each examination as needed.

By creating a nontechnical document that caters to the intended audience, describes the processes and procedures, and supports the conclusion based upon the evidence, additional explanations to the content are seldom needed. An examiner should strive to produce a concise, detailed document that can stand on its own without additional explanation or commentary.

What Data to Include in the Report

The content of an investigatory report should be consistent and structured so that the examiner can use a similar format for each mobile device investigation report. When an examiner creates a report using boilerplate language and format, he or she can use the same configuration for each examination, with adjustments as appropriate. Creating a well-functioning document starts with a well-formed outline. The examiner can use the outline during the examination and analysis phase to help with documenting the appropriate details and focus on the particular requirements of the outline, which often results in a more detailed report.

Note

The information contained within this section is only a recommendation. An examiner can add, remove, or modify these suggestions to fit his or her own needs. Some critical items should always be included in investigatory reports, and these are noted within the context.

What Right Do You Have?

The initial information in the report should indicate not only how the examiner received the mobile device and conducted subsequent analysis but also what right the examiner has to collect and analyze its contents. If a search warrant had been obtained to search the device contents, the examiner must indicate this. If an examination does not require a search warrant (for example, consent is given to search, a private investigator is searching, the device is enterprise owned, or the examination is part of a contract), these details must be included in the report to describe why this search and analysis could be legally conducted. Failing to describe the examiner's right to conduct an investigation into mobile device data commonly results in additional questions and concerns from the intended audience.

Documentation that describes the legality of the examination should be included in the Supplemental Documents folder. The text within the rights section of the final presentation will distinctly outline the contents of the supplemental documents and will refer to the location in case the audience needs more detail. Also, the chain of custody form for all exhibits should be included in this folder and should also be documented within the main document. By including this information in the overall presentation, the examiner lays the foundation for the validity of the entire mobile device collection and data analysis. This section should be a mandatory part of all documentation.

The Five W's

Who, *what*, *when*, *where*, and *why* are known as the "Five W's" and are important in all types of writing efforts. Most notably, law enforcement practitioners use these to develop questions to answer when creating reports and also while handling investigations. The answers to these questions should appear at the beginning of the mobile device examination report to set the stage for the processing section of the piece.

The content in the initial portion of the document should clearly introduce the Five W's, setting up the rest of the content and the examination process. By outlining this information first, the examiner helps the audience better understand the rest of the document and supporting materials. When the examiner quickly dives into "This is what I found," readers may be overcome with information overload. Setting the stage for the readers and slowly outlining the necessary materials will give them context for the upcoming content. This section should be a mandatory part of the documentation.

The *who* question is not necessarily "Who dunnit" for an examiner, but a statement of who requested the mobile device processing—often this is the primary investigator, but it may be the person performing the examination. When a request comes from the primary investigator, the examiner should document not only the name of the individual but also the information surrounding the request. If an investigator has requested that the examiner process "a Samsung SGH-i535 mobile device for documents sent from Bob Kelly to Major Tom sometime between April 1st and April 5th, containing classified materials," this information should be included as part of the *what, when,* and *where* questions. In addition, the *who* may be other persons involved in the case: "Mary Todd, the plaintiff, allowed her iPhone 8 to be examined as part of an investigation surrounding an internal workplace complaint. Ms. Todd stated she had received threatening messages on June 1st, 2017, from Mr. Roger Dokken, her supervisor." The *what* and *when* along with *where* can also be satisfied within this statement. Of course, an examiner should elaborate on each point, making sure every detail is outlined within the processing document completed by the subject who requested the mobile device examination.

The *why* question should explain to the audience the reasons the examiner has been tasked with the collection and examination of the mobile device. Furthermore, it should describe what this particular request for processing has to do with the overall case.

With mobile devices today, just "dumping" a phone, hoping to find something, is like trying to find a needle in a haystack. Massive amounts of data are stored within mobile devices as well as within the Cloud, and an examiner must have a clearly defined objective prior to starting the examination. This objective is the *why* portion of the document.

Tools Used

The next section of the report should outline the tools used throughout the process, including software used to isolate, collect, analyze, view, and report data. The information can be in table form (see Table 17-2) and should include the company that produced the software, a link to the software, the version number used in the analysis, and a comment section. By listing this information within the main documentation, an examiner can duplicate the entire process if necessary.

TABLE 17-2 **Example Table Listing Software Used During the Examination Process**

Tool	Version	Download	Notes
UFED4PC	7.2	https://www.cellebrite.com/en/support/product-releases/	
Cellebrite Physical Analyzer	7.2	https://www.cellebrite.com/en/support/product-releases/	
Oxygen Forensic Detective	10.1	https://www.oxygen-forensic.com/en/products/oxygen-forensic-detective	

Examples Using the Five W's

Suppose a mobile device was seized in a drug case and another device was seized in a case involving a workplace harassment complaint.

In the drug case, the *why* is often defined in the search warrant affidavit that describes how this device was used to communicate via text messages, pictures of drugs, and documents or ledgers stored on the device and used in the sale and purchase of drugs. With this information, the examiner can formulate a plan for the examination and then document this plan within the *why* portion of the report:

> *Detective Ryan indicated the mobile device examined was used to communicate using text messaging and an app known as SnapChat with customers. Detective Ryan also has reason to believe the subject used the mobile device to store and compile records of sales and purchases of drugs using a mobile app and has used the built-in camera to take pictures of drugs and drug paraphernalia as well as receive pictures of drugs.*

In a workplace complaint, examining the *why* portion is similar, with the content again derived from the complaint document. If the document alleges that an employee received messages and photographs of a harassing nature from another employee, the examiner should be able to articulate the *why* using this content and an understanding of what types of data will be examined:

> *Human Resources Director Smith received a complaint from employee Amy Christine, who received nude photographs from another employee, Dale Dummy, along with inappropriate messages in Facebook Messenger, Google Voice, and Tinder in violation of company polices. Ms. Christine relinquished her company-owned mobile device to Director Smith for the examination and recovery of any information involving the complaint.*

Usually, two experts are consulted in a trial—one will validate the findings of the other. By listing the specific details for each hardware and software title, along with version numbers, a second examiner can set up the same environment used in the original analysis. Another reason for including the software titles and version numbers is to assist the initial examiner. By being able to recall the software used in the examination and the version number, the examiner can resolve problems if current software versions do not provide the same results at a later date. Using the software version, the second examiner can install the exact version that was used for the initial examination to ensure a proper verification and validation platform. This may seem like overkill, but with the constant updates to mobile forensic software, there are bound to be changes that could affect an evidence image. Examiners should maintain all the versions of the software, if possible.

Note

Maintaining all versions of the software is not mandatory, because storing many versions can be difficult and expensive. However, most software vendors provide a repository where examiners can download past versions, so maintaining the storage site is unnecessary. An examiner should become familiar with these locations for all software that will be part of his or her mobile forensic toolbox and know how to obtain old versions when needed.

Isolation Methods

As part of the main report, the examiner should include how the device was isolated from the cellular and data networks, when it was received, and when it was examined. As covered in several chapters in this book, isolating a mobile device from outside networks is mandatory and should be done as soon as possible. Allowing a mobile device to remain connected can have devastating effects on the data within the device, ranging from altering the file metadata to deleting the entire file system.

When conducting a mobile device collection, the examiner is essentially working with a snapshot of the device data—a picture of the device from a time period leading up to the device's seizure, which in most occasions is right around the time of the event. If a device connects to a network, the probability of data contamination is great and the entire time line, frozen in time by isolation, could be spoiled. The documentation must include this valuable information.

At the Scene If the examiner seized the mobile device at the scene, the documentation of the isolation technique will be straightforward. The examiner must make sure to include the steps taken at the scene to isolate the device and the state the device was in, along with date and time information when the device was isolated.

At the Lab If the mobile device was seized by someone other than the examiner, the examiner must indicate the state the device was in when he or she received it. This includes whether the device was powered on or off, inside a Faraday bag, powered on and in airplane mode, or, in some cases, powered on with no form of isolation.

If the examiner is receiving the device either directly from the field or from a property room, he or she must ensure that clear documentation has been received from the person who seized the device from the location. This information should indicate how the device was isolated (airplane mode, isolation bag), how the device was packaged at the site (Faraday box, bag, or container), and the date and time of the isolation. Sometimes this information is difficult to obtain, so the examiner must be diligent and contact the person who seized and isolated the device if this information has not been documented. If it has been documented, the examiner can refer to the report created by the person who isolated the device, which should be placed into the Supplemental Documents folder.

Tip

If a device is brought to the examiner powered on and not isolated, the examiner must immediately take steps to isolate the device using his or her standard operating procedure.

Whatever the case, the examiner must outline the details, indicating the removal of the device from the network, how it remained isolated from the network during the examination, and if it was not isolated, the circumstances as to why the device was not isolated. An examiner failing to include this information will often have to explain away the possibility of contaminated data, altered file dates, or other anomalies within the device file system.

Collection Methods

As covered in Chapter 10, examiners can use various collection methods to obtain data from a mobile device, ranging from an invasive chip-off to photographing the mobile device screen. Whatever the technique used, the examiner must outline in detail the way in which the data was collected.

If an invasive chip-off physical collection, JTAG collection, or ISP collection occurred, the examiner must document the reasons behind such a technical collection. Typically, these methods are used because gaining access to the mobile device using any other means was impossible—perhaps because the device was password protected, because the device had been destroyed by chemical or mechanical means, or because no mobile forensic solution supported the collection of the device's internal file system.

In most cases, a mobile device will be tethered to a personal computer with a cable during the examination. This cable is the conduit between the software and the mobile device. Mobile devices communicate using protocols and commands that have been outlined throughout the book for both logical and physical collections. The commands used by mobile forensic software and other collection solutions travel via the attached cable, negotiating a backup of the device's internal memory stores or files. This backup can occur either via these commands or after the installation of an application to the mobile device, such as an APK in Android. Of course, communication can occur via other means such as Bluetooth, but the documentation is generally the same; communication to the device using commands causes the device to deliver the requested data to a location specified by the mobile forensic solution.

In all cases, data changes occur within the mobile device, but the examiner must provide a detailed explanation of any system data changes that occurred within the internal store that was not user data. The examiner should be prepared to explain how this is known in case this information is challenged. The explanation of verification and validation does not need to be a part of the final documentation, but it should be a part of the examiner's notes and knowledge base, since this information will surely be brought up in a trial.

An examiner must be confident in detailing the collection methods within the documentation because credibility of the process will often be tested. If an examiner has not documented the details of why an invasive method was or was not employed, why one software solution collected and displayed a piece of data that another did not, he or she may have to answer to the report audience. This information should be mandatory in the final documentation.

Collected Data

As you know, an examiner should not simply rely on the output from the mobile solution to document the collected data. The examiner must detail the main information within the body of the final document and there refer to the report or reports created by the tool. The audience does not want or need to delve into the thousands of contacts and SMS messages; they need a clear and concise view of the requested data, not a data dump they have to sort through.

The collected data portion of the report should include images and clearly defined, measurable details. If, for example, the UFED Touch 2 was used to complete a physical collection of a Samsung Fascinate SCH-I500 and the userdata.rfs and dbdata.rfs partitions were collected, the report should specify the time of the collection, the size of the binary files, and the overall hash of both binary files within the main body. If possible, and if the tool supports this information, the examiner should list the total number of files in the partition's file system. This information can be valuable when describing the massive amount of data within a mobile device to the audience.

When possible, the examiner should detail the information that was extracted via the automated tool, indicating the number of items that were contained in each user data container and the pertinent information that was recovered for each. If possible, the examiner should include the file from which this data was recovered along with an overall hash of that file. This can ensure that another examiner can duplicate the results using the same file, even if he or she has to use another tool to complete the analysis.

Here's an example of a statement that can be used when describing call history records:

> *500 Call History records were collected from the mobile device. The call history database was located within the file system at \data\com.sec.provider.logsprovider\logs.db. A MD5 hash of the file was obtained: c7da421a8de57e813f947b713f676405. Of the 500 records, 212 calls were made to the mobile number of Ms. Smith and there were no incoming calls from Ms. Smith's mobile number between the dates of June 1, 2015, and June 4, 2015. Of the 212 calls, 15 calls showed a duration of 5 minutes or less, with the balance indicating 1 minute or less.*

The examiner should detail this information for all collected data from a mobile solution as well as information manually collected using advanced methods such as SQL queries or scripting.

When an examiner must use advanced methods, these methods do not need to be explained in detail within the document, but the collected data should include context. Context will help to set the stage for the audience on the process that was taken and the reason why the automated tool was unable to locate all of the important data. This data is often at the center of the audience's attention because the automated tool "didn't find it." As you know, an automated tool not finding app data is a regular occurrence, primarily because of the plethora of available apps (more than 5 million in iOS or Android alone). This goes for free file parsing for deleted data from SQLite databases, file carving, and string searching. When using this type of analysis, the examiner should include a section of the report dedicated to outlining this material, apart from what the automated tools collected. This is particularly important when the collected information is relevant to the case requirements and outline.

The collected data portion of the document should reflect all the collected areas that contain valuable data, as requested or as part of the overall data picture. Also, a brief narrative should describe any collected data containers that are void of significant data:

> *Out of the 525 contacts located on the mobile device, the name Ms. Smith, her phone number, e-mail address, or other identifying information was not located.*

If an examiner neglects to include information indicating that something was not located, this missing data is often questioned. Why did the examiner not indicate there was no information? What is the examiner trying to avoid? What other information might be left out? These are all valid questions. This section should be mandatory for the final report.

Tools That Collected and Analyzed the Data Usually, the examiner uses more than one tool to collect and examine the data: one tool is used to collect the device image, while other tools are used to parse and dissect the data. The document should distinctly outline the tools used to complete the collection of the image and ensuing analysis. This information should be documented in the collected data section in each area where multiple tools were used. Within the collected data section, examiners can add multiple subsections defined by the tool used, and any data that was analyzed is listed in each subsection. This is often the best method, because the information can be easily followed by both the audience and the examiner creating the report.

In the tools section, the examiner can list any discrepancies in the data parsed when using multiple tools. One tool may parse additional fields from a SQLite database that another tool did not, providing a different set of results. When this occurs, this information should be part of the documentation. Again, when this information is not documented, the examiner's report can lead to serious questions on the use of the additional tools. As an examiner who has completed and reviewed hundreds of mobile device examinations and their data, I know that this will happen more often than not. Commonly, the differences in the data result from the queries the tools are using to collect the data from a SQLite database. By outlining this information in the report, an examiner will generally quell any potential challenges. This section should be mandatory for the final report, even if a single tool was used for collection and analysis.

What Does the Data Mean?

The final section of the main report will outline the key points of the case, what was requested, and the information that was located that supports the case. This summary area includes the opinions of the examiner based upon the facts of the case. This section should be supported by the collected data area and should not include any data not already mentioned. The conclusion area of the report should show the audience what the collected data means. Data from a mobile device is fact; it is a snapshot of information from the device's internal store. This data can be confusing if an examiner does not provide a conclusion in the report.

Sometimes the data found within a mobile device does not include the particular information that was requested, or it may include information that contradicts the original hypothesis. This information should be emphasized in this section. An examiner should also use this section to educate the audience on the differences between the types of data collected. The examiner should be prepared to explain the multiple types of digital data to the audience—data on its own and in the context of the examination focus.

This section of the report is often the most powerful one, because it provides the big picture, an interpretation of the data collected. Whether multiple devices were collected or data was collected from a single event, including a collective interpretation of the data is recommended, if not mandatory.

Data on Its Own Reports are often spit out by automated tools as single investigations, and this data is interpreted as a singular event, which looks much different from data collectively analyzed. When a mobile device is the only piece of evidence requested and collected, the data is specific to that device only. Mobile devices are seldom used independently, however, and are instead used for communication either directly, with another mobile device, or via an intermediary medium such as a server.

Collective Interpretation Seldom is only a single device involved in a mobile device investigation. The examiner usually creates a report detailing the data collectively. An examiner should create subsections within the overall report that define the individually collected data elements, showing where the data was collected and what data was collected from each piece of evidence as an individual exhibit. This enables the audience to see the data collected from multiple sources, such as mobile devices, cloud stores, computers, servers, and flash media, individually. This is important, particularly if one piece of evidence contains unique information not found in the other evidence.

The conclusion should contain the examiner's collective analysis of all the data contained within each device as it relates to the others. The examiner should show relationships among the collected data based on time lines, conversations, and any other artifacts that show collective association.

To Include or Not to Include

Examiners should know what information will be distributed, what will not be distributed, and what will be archived. The documentation should present a list of the included materials to introduce the audience to the contents. This information can be in a README.txt file included with the distributed media. The reader can use this information to identify and refer to the material within the main report. Usually, the materials the examiner includes with the report are governed by the SOP developed by the examiner's sponsoring agency or company.

Some items should not be distributed in the completed report—that is, items that do not provide value to the audience. However, this information should always be available if requested. No materials should be removed or deleted from the case until directed to do so by the SOP or other authority. Following are some basic suggestions on what to include and what not to include.

Include Media, Documentation, and Screenshots

All media extracted from the mobile device that is part of the case should always be included in the evidence folder within the distributed media. Any pictures, video, and audio taken of the evidence by the examiner or subjects seizing the data do not need to be included. (See the upcoming section "Do Not Include Forensic Evidence Images.")

References to screenshots of software or the analysis can also be made, though the actual screenshots themselves need not be included in the main report. Because screenshots are created during the investigation, either during the collection of document settings or during the analysis of document settings and data, references to the screenshots should be included on the distributed media. The screenshot files should be clearly named and referenced in the main report. The actual screenshot files should be placed in the Supplemental Documents folder in a subfolder named Screenshots. The reader can then reference this material, if needed, to support the text in the main report.

Written documentation outlining the process should be included in the case report, clearly indicating the existence of the media. However, media files such as notes do not need to be included unless requested. (See the upcoming section "Do Not Include Examiner Notes.") These files should be saved and archived so that they can be produced if needed.

Include References

It is often important to include reference material for the audience, such as common mobile device terms and acronyms with understandable descriptions, infographics showing the number of mobile devices collected, graphics showing the way a mobile device communicates with a network, and breakdowns of data as appropriate. A picture may convey the message better than text. This, of course, will be directed by the type of investigation the examiner is conducting.

For common mobile forensic terms, the examiner should compile a list to be included within every report. As mentioned, the examiner should always strive to avoid technical jargon. This information should be included in the Supplemental Documents folder and referred to within the main report text, if needed.

When imagery or details such as infographics or created images outlining the decoding of an artifact are included, the image should be referenced to support a written description within the main documentation. The actual image should not be a part of the main report. This information should be contained with the Supplemental Documents folder.

Do Not Include Examiner Notes

Notes compiled by the examiner, such as the formatted note template discussed earlier, should be used by the examiner to compile the final report and to support any challenges to the reception, condition, and handling of the evidence. Unless the notes are required or requested, they can be left out of the main report. However, all notes and material used to compile the final report should be archived, along with the data and the final submitted report. Each page of written material compiled by the examiner should contain a signature with a date. This security precaution can assist both the examiner and audience if the authenticity of the documents is challenged.

Do Not Include Forensic Evidence Images

The examiner should not include the actual forensic images of collected devices used during the investigation and subsequent analysis in the main report. Forensic images are used for collecting intelligence, not for explaining or detailing the evidence recovered. Instead, the original forensic images should be archived with all other data and documents compiled and created for the case. The images will be maintained as outlined in the agency or company SOP.

The forensic images, however, can be requested by the audience if it is determined that the evidence will be examined by a second examiner. At that time, the forensic images should be copied and placed in their own storage media and given to the appropriate persons.

In some cases, if data contained within the image includes criminal images (such as child pornography) or sensitive company information, the images are generally not released for obvious reasons. In such cases, the audience requesting the images often views the information at a designated location; this maintains the legality and sensitivity of the evidence, since distributing or making copies of this evidence could technically be a crime. The original examiner should inform the reader of the sensitive content to avoid questions as to why the forensic images cannot be disclosed.

On December 1, 2017, two new Federal Rules of Evidence were added to FRE 902: Rule 902(13): Certified Records Generated by an Electronic Process or System and Rule 902(14): Certified Data Copied from an Electronic Device, Storage Medium, or File. Rule 902(14) pertains the most to the forensic examination and production. FRE 902 lists the evidence that is self-authenticating and outlines the items that do not need additional authentication to be admitted into evidence. These rules were added because of the large amounts of electronically stored information (ESI) used in trials today. Bringing in experts to testify as to how a forensic tool works to extract data, or to authenticate the text message existing on the telco's server, or to explain how a page on a Facebook account is extracted and produced is, in fact, valid, but doing so proved to be overly time-consuming for the courts, and the cost of hiring these experts (if one could even be found) was difficult to gauge.

The most important rule to the mobile forensic examiner conducting the extraction is FRE 902(14):

> Data copied from an electronic device, storage medium, or file, if authenticated by a process of digital identification, as shown by a certification of a qualified person that complies with the certification requirements of Rule 902(11) or (12). The proponent also must meet the notice requirements of Rule 902(11).

In other words, if the process can be authenticated by a hash that can be verified as consistent, and the person who conducted the extraction and hashing is competent in using the tool and processing and has certified the results, the data can be admitted as evidence without additional authentication from another party.

Becoming a Mobile Forensic Device Expert

This book is devoted to the examiner's journey to becoming a mobile forensic expert. Becoming a mobile forensic expert has nothing to do with certifications received for performance on aptitude tests, subjective practical exams, or other written tests. The word *journey* best describes how becoming an expert should feel.

Unfortunately, a lot of individuals who are starting out in the field of mobile forensics would rather it be a called a *sprint*. Most look for certifications as a quick way to find a career and believe that a certification will somehow propel them into becoming an expert. It is this belief that has cast a shadow on a field that is growing at breakneck speed, where the number of examiners considered experts has flatlined.

The word *certification* has been mentioned throughout the book. Certifications currently come in two main flavors: vendor-created and tool-based. A *vendor-created certification* "certifies" that the examiner knows how to use the vendor's product; this is not the same as being an expert in mobile forensics. A *tool-based certification* indicates that the examiner has completed the necessary steps to use the vendor's product; it may simply be a label to attach to a name on a calling card.

Currently, mobile forensics lacks a *general certification*, which would expose an examiner to processes and procedures, various tools, a variety of methods, and a core set of proficiencies. This should be changing soon, however, with the addition of a general mobile device certification board at the International Association of Computer Investigative Specialists (IACIS). However, an examiner does not become an expert in the field merely by getting certified—it takes a lot of hard work.

A mobile forensic expert is an examiner who never relies completely on an automated tool or process. This examiner resists the temptation to click the "Easy" button in search of the truth within an ever-growing amount of data. The expert examiner is never satisfied but will fine-tune the examination processes and procedures whenever possible. The mobile forensic expert will spend countless hours exploring in an effort to uncover massive amounts of data, including never-before-found artifacts.

The ability to locate data using manual techniques does not, on its own, define a mobile forensic expert, but it is part of the equation. Once the expert examiner locates this data, he or she will research and test to arrive at a conclusion as to why this data is written to the device's storage area. Furthermore, the expert will seek to comprehend the processes that took place or that needed to occur for this type of evidence to be written to the device's storage. Then the examiner can determine what this data might mean to the investigation—by looking at the totality of the evidence, not just a located artifact.

The road to becoming an expert is full of switchbacks and curves, and examiners must navigate undocumented terrain. Experts do not wait for the instructions; they often write the instructions themselves. One of the most critical aspects of the investigational process is peer review. When instructions, procedures, and methods are created and tested, the examiner should allow for the review of the process and artifacts. This critical stage will not only maintain the processes' integrity but will also help processes continue to improve. Changes are needed to maintain the most up-to-date and relevant procedures consistently in the ever-changing mobile device environment.

In 1990, psychologist Anders Ericsson determined that 10,000 hours of repetition was the secret to becoming an expert. However, how the 10,000 hours are spent matters. To become an expert means to practice, conform, and evolve to refine the discipline, constantly updating and improving the methods. For a mobile forensic examiner, this includes not only collections but also research, documentation, and testimony of findings.

The legal world often describes a forensic expert as an individual who provides an opinion more knowledgeable than that of a typical layperson, based upon specialized knowledge or training. The expert's opinion is deemed more valuable because it is based upon scientific, technical, and specialized training, education, or experience.

Obtaining testimonial experience of the findings of an examiner's mobile device investigation is also extremely important. Whether it be simply explaining the findings to a council, board, magistrate, judge, grand jury, or state or federal juries, providing testimony is critical. This type of practice is essential for an examiner on his or her journey to becoming an expert in the field. Providing testimony will help an examiner discern the value of meticulous research and conscientious documentation—either good or bad. During testimony, the examiner will be expected to support any opinion with facts presented in his or her written documentation, as well as evidence uncovered. If an examiner cannot support his or her examination with insight into the data's existence within the mobile device by documentation or testimony, the data is difficult, if not impossible, to use in a case.

By investigating the mobile device, formulating the documentation, and grasping the principles outlined in this book, the examiner will have the best chance of success. The more an examiner is able to testify to the findings uncovered in a mobile device examination, the better the testimony will be, and the better the examinations of the data will become.

Importance of a Complete Collection

A *complete collection* is one of the most important considerations for a mobile forensic examiner. A complete collection of a mobile device contains a file system, not just an outputted report with data pulled from the mobile device. Of course, in some situations, an examiner will be unable to produce a collection that includes a file system because of limitations of the mobile device or supported software. However, if the mobile device can be collected to include a file system, which is possible with the majority of today's devices, there is no excuse not to include a file system in each collection.

Unfortunately, mobile forensic examiners often need to collect as much information as possible in the least amount of time. As a result, software vendors often highlight their product's collection speed over other features. But this speed can also result in a partial collection or a partial parsing of the data. Some examiners contend that there is not enough time to complete a collection that includes the file system, but an expert examiner's logical iOS collection *must* include the entire file system. An expert is not the examiner who merely pushes a button to extract data from the mobile device.

Full Examination Importance

In the early years of training federal, state, and local law enforcement officers, while explaining the necessity of understanding where the data came from when conducting a collection using the UFED Classic forensic solution, many instructors used the following real-life story to convey the importance of credibility:

The officer who took the stand had performed the collection of an LG VX6100 legacy telephone. The officer was called to testify about the call logs and text messages that had been recovered from the mobile device and were being used as evidence in the trial. He was asked for his credentials and training, along with his number of years of service. The evidence was then presented to the jury. The officer testified that the call logs and text messages were extracted from the mobile device using a UFED device. The prosecutor had the officer explain how the UFED device is attached to the mobile device and how data is extracted to a flash drive. The drive is then inserted into another computer where the report can be viewed. The report was also presented as evidence. The prosecutor did not have further questions.

The officer was then cross-examined by the defense counsel, who again asked about the evidence. The officer again explained that the information was obtained from the mobile device using UFED Classic, which extracted the data to the attached flash drive. He was then asked if any other data was collected from the phone. The officer replied that there were contacts and some pictures. The officer explained that all this information was in the report created automatically by the UFED. The word "automatically" perked up the defense counsel, who said this:

"So…. Can you explain 'automatically' to me?"

The officer then stated, "The data is pulled from the phone by the UFED unit automatically after I put in the make, model, and the items I want to extract."

"OK, that makes sense," the attorney said. "But, where does the data come from?"

"From the phone," the officer added.

"Yes, I understand it comes from the phone, but where in the phone? Is there a file that the text messages, contacts, or pictures come from?"

The officer responded, "It came from the phone, I pushed the button on the UFED and the data came out."

The attorney finally asked, "So, the data in the report is automatically produced by the UFED after it magically connects to the device to obtain the data, and you have no idea if there are actual files on the device?"

The prosecutor then objected, saying that "the officer is not the manufacturer of the UFED and should not have to testify to how it works." The objection was sustained, but then the defense called its own expert, who had also obtained the mobile device file system. He showed the changes made to the device and pointed out the actual files that contained the call logs and SMS messages. At that point, the credibility of the first officer's testimony was significantly challenged.

This true story is from many years ago, but similar stories are still typical during mobile forensic testimony. By completing a full and complete collection of a device, understanding the methodology, process, and procedures, an examiner can overcome such challenges.

Conforming to Current Expectations May Not Be the Best Approach

Based upon misconceptions of the process and procedures along with the methods of collection, mobile forensics has sometimes been compared to magic. Many software vendors early on believed that ease of use, limited access to device file systems, immediate reporting, and a simple output was what the mobile forensics community needed. The tool would be so easy to use that little or no training would be required. This backward approach to mobile forensics continues to be the current expectation for examinations across the globe.

This book was written to inform the mobile forensic examiner that by conforming to this type of collection expectation, an examiner becomes a person who simply presses a few buttons and smiles when the data materializes on the screen. This type of collection shows zero regard for how that information was obtained, where it was obtained, and whether or not additional information was collected by the tool. This conforming attitude among examiners will continue unless more examiners seek to become experts in the field of mobile forensics.

Today's mobile forensic examiners must realize that thousands of files are available within a mobile device's file system, and these are often viewable using free open source tools. These files and file systems represent digital gold to the examiner who knows what to look for. An examiner today should be digging into these areas, creating automated scripts, researching the various file types, and discovering new methods.

Mobile forensics expert examiners are pioneers who know that the forensic processing of a mobile device and all the associated data should include the application of the methods and theories expressed in this book, instead of conforming to conventional mobile device collection processes.

Additional Suggestions and Advice

This book has provided solid techniques and approaches to mobile device forensics, plus real-life examples from the author's many years of being a practitioner in the field. It has offered suggestions on how to make mobile device collection and analysis a process that can be duplicated when needed.

An examiner should never look for a faster way to perform a collection or complete the parsing and analysis of the collected data. Instead, an examiner should look for the best methods and means that will cost the least amount of time in explaining what was done, or not done, during the examination. Once a report of the examination has been distributed, a good measure of the work can be indicated by the questions that follow. If an examiner shortcuts the examination, the odds are good that he or she will face multiple challenges and questions by the readers and audience.

The following sections provide some additional suggestions for the examiner who seeks to become an expert in the field of mobile forensics.

Constant Research and Testing

An examiner must stay up-to-date with the rapidly changing technology, continuous updating of mobile device software, and massive influx of mobile forensic solutions and features. The diverse and ever-changing mobile device landscape creates a challenge to the dedicated mobile device forensic examiner.

Apple sales surpassed those of Samsung in 2016 but fell behind in 2017 when the Samsung Galaxy 8 was released. In 2017, Samsung again maintained that 1 million devices were being sold each day, and Google reported in 2015, and again confirmed in 2017, that there were more Internet searches completed using mobile devices than personal computers. Moore's law suggests that technological growth doubles every 18 months, but at times the exponential growth of mobile device technology exceeds this estimate. This is compelling information, but unfortunately mobile forensic examiners and software solutions are having a difficult time keeping up. An examiner must take it upon himself or herself to research continually the various devices coming to market, the newest file systems, and the trending apps that will surely make their way onto the next mobile device submitted for analysis. An examiner who relies on forensic solutions alone will only be as good as the tool that he or she is using. By staying in pace with the technology, an examiner will be adapting, practicing, and gaining more time toward the 10,000 hours needed to be an expert in the field.

More Mobile Forensics Information

Compiled in Table 17-3 are web sites where an examiner can find an abundance of information to assist on the road to becoming an expert examiner.

TABLE 17-3 **Web Sites with Helpful Information for the Examiner**

Site	Location	Description
Apple Developer	https://developer.apple.com	Main Apple site for iOS developers. Great place to get pre-release versions, if registered, or use the support area to learn about the iOS file system.
iPhone Developers	http://forum.iphone-developers.com/	Run by XDA developers and dedicated to analysis and exploits within the various iOS file systems, apps, and so on.
Android Developers	http://developer.android.com/	Main Android site for developers. Many support areas discuss internal classes, ADB commands, and much more. A must for examiners.
Android Central	www.androidcentral.com/	Up-to-date information on all things Android, including new releases, coming releases, and predictions on technology.
DroidForums	www.droidforums.net/forums/	Much like Android Central, but with hacking, modding, and rooting information.
CyanogenMod	http://forum.cyanogenmod.org/	Leader in custom ROMs for Android devices and frequented by mobile forensic software developers who create non-invasive physical bypass ROMs for Android.
Windows Dev Center	https://dev.windows.com/	Main Windows Phone development site. Use the search bar to find many tips on file system locations, files, and more within the forum.
XDA Community Forums	http://forum.xda-developers.com/	Offers a wealth of information on mobile device internals and contains many repositories used by software solutions in their own tools—for iOS, Android, Windows Phone, BlackBerry, and more.
Forensic Focus	www.forensicfocus.com/Forums/viewforum/f=14/	The mobile phone forum, plus other forums for applicable laws and other digital forensic forums. This active forum includes many experts in the digital forensic field.

Chapter Summary

The path to becoming a mobile forensic expert is an arduous one that does not depend on quick solutions. An expert is defined by his or her successful presentation of abundant facts, knowledge and training within this specialized field, and constant need for growth within the discipline.

The examiner presents his or her discoveries in a fact-filled document to a predetermined audience—ranging from a single individual to a judge and jury. A skilled examiner prepares the final document to suit a multitude of readers by creating a report that is structured to introduce the reader to the investigation's five W's, present the process workflow, and then define the conclusion reached based upon the evidence. As this chapter explains, an automated report generated by a forensic solution does not provide the big picture. The automated report can contain exceptional data, but it provides no context. A reader needs context to see the big picture of the captured data, and the final document created by the examiner provides this context.

Collecting valuable data from a mobile device and creating a tremendous final document doesn't make an examiner an expert by any means. Becoming an expert in any field requires that an individual constantly adapt to the changing environment, to seek new ways to solve problems, practice the craft, redirect the course if needed, and become immersed in ideas to improve the discipline. Becoming an expert is not about becoming better than someone else, but involves challenging yourself to become the best you can be. The only way to accomplish such a lofty goal in mobile forensics is to be vigilant in training, research, and testing of new methodologies and ideas. An expert is not made with a certification document, but through hard work and dedication.

The road to becoming an expert in the field of mobile device forensics will be long and often filled with potholes. However, along the way, the examiner will meet many other great examiners and practitioners, and those meetings and conversations will show the examiner the benefits of hard work and perseverance.

Index

Please note that index links point to page beginnings from the print edition. Locations are approximate in e-readers, and you may need to page down one or more times after clicking a link to get to the indexed material.