GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at http://www.giac.org/registration/gcfa

Reverse Engineering the Microsoft Extended FAT File System (exFAT)

GIAC (GCFA) Gold Certification

Author: Robert Shullich, rshullic@earthlink.net
Advisor: Aman Hardikar

Accepted: 12/01/2009

ABSTRACT

As Technology pushes the limits of removable media - so drives the need for a new file system in order to support the larger capacities and faster access speeds being designed. Microsoft's answer to this need is the new Extended FAT File System (exFAT) which has been made available on its newer operating systems and which will be supported on the new secure digital extended capacity (SDXC) storage media. This new file system is proprietary and requires licensing from Microsoft and little has been published about exFAT's internals. Yet in order to perform a full and proper digital forensics examination of the media, the file system layout and organization must be known. This paper takes a look under the hood of exFAT and demystifies the file system structure in order to be an aid in the performance of a digital investigation.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

1 Introduction

In the US DOJ Special Report released in April 2004, *Forensic Examination of Digital Evidence: A Guide for Law Enforcement* (US Department Of Justice (2004)) one of the steps for evidence examination under Application and File Analysis is:

> "Examining the users' default storage location(s) for applications and the file structure of the drive to determine if files have been stored in their default or an alternate location(s)"

How does the forensics examiner accomplish such a feat when the file system is unknown or not documented? This task becomes a real challenge when having to do an analysis on proprietary systems such as embedded systems. But now, with the drive towards storage media with larger capacities, the limits on many of the existing file systems will be reached during the newest wave of storage technology.

To accommodate these advances, a new file system has been developed by Microsoft a few years ago, and it is called the Extended FAT File System, abbreviated as exFAT, and what some are nicknaming as FAT64. Microsoft is licensing this technology, so in order to implement an exFAT file system a license will be required from Microsoft. In January 2009 a new Secure Digital Extended Capacity (SDXC) specification was announced (Hissink, 2009), with capacities that could reach up to 2 TB, and will use this new exFAT file system. This new file system may actually fly and gain momentum in 2010 when device support reaches the market.

But today, there is no real Linux support, very few tools support this new file system, and even the commercial forensics tools are behind in support. There are very few, if any, open source tools that understand the file organization, and just recently the specifications of the exFAT file system got released with one of Microsoft's patent applications (Microsoft Patent 0164440 (June 25, 2009)).

How does the forensic examiner "examine the file structure of the drive" when the tools don't know, and there is no how-to book to help him? This paper is intended to provide basic insight to the file system structure to allow the forensics examiner to make sense of the structure beyond just a blob of bytes.

Robert Shullich rshullic@earthlink.net

© 2010 The SANS Institute Author retains full rights.

2 Definitions

Steps in Processing Digital Evidence – Assessment, Acquisition, Examination, Analysis, Documenting and Reporting. (US Department Of Justice, 2004)

Digital Evidence – Any data stored or transmitted using a computer that support or refute a theory of how an offense occurred or that address critical elements of the offense such as intent or alibi. (Casey, 2004)

Digital Forensics – Digital forensics involves the identification, collection, preservation, examination, and analysis of digital evidence. It is a technical, computer-related field involved in the collection and examination of evidence from computers, including audio, video, and graphical images. (http://www.ncfs.org/digital_evd.html)

Forensic Examiner – Conducts the examination process to extract and analyze digital evidence. Extraction refers to the recovery of data from its media. (US Department Of Justice, 2004)

File Fragmentation – for the purposes of this paper, a file is considered fragmented if the clusters that the file is stored in either are not in order or there are gaps in the physical cluster layout, or both. A file is considered not fragmented when the file is physically stored in order within contiguous clusters.

Removable Media – is storage media that can be removed from its reader and stored or transported to another location, possibly to be used on a different machine. Examples of removable storage media are floppy disks, magnetic and paper tape, flash drives, flash cards, CD/DVD, and ZIP/JAZ. This paper will address removable media that is random access, which eliminates purely sequential devices such as magnetic and paper tape.

Superfloppy – a configuration where the entire storage media is a single file system and there is no partitioning. There is no MBR record and when the media is booted the VBR is loaded by the BIOS. Not all BIOS firmware will support a superfloppy. The concept of the superfloppy was introduced when media such as 3M's LS-120 and Iomega's Zip disks surpassed the conventional 1.44MB capacities.

© 2010 The SANS Institute Author retains full rights.

3 Prior Work

There does not appear to be much research released at this time. The exFAT file system has been in the market since 2006 with its introduction in Windows CE 6.0, but exFAT didn't hit the desktop/server market until the release of Vista SP 1 in March 2008. The support has effectively existed on the desktop for almost 2 years.

At the Techno Forensics Conference that was held at NIST in Oct 2009 (http://www.thetrainingco.com/html/TechnoForensics2009.html) Jeff Hamm from Paradigm Solutions gave a presentation on the internals of the exFAT file system. He provided a presentation and paper on the topic, which provided a good foundation for the work being presented here. His work is based on a forensic class he teaches that includes exFAT internals.

4 Setting a Foundation

4.1 Purpose, Disclaimer and Scope

4.1.1 Purpose

The purpose of this paper is to describe the format and layout of the Microsoft exFAT file system as currently released in the Microsoft desktop and server platforms. The intent is to aid in the forensic examination of storage media that is formatted with the exFAT file system. This document can be used as a guide for the forensics examiner in order to provide a starting point in the search for electronic digital evidence that may be stored or hidden within this file system.

4.1.2 Disclaimer

The exFAT file system is proprietary property of Microsoft, and an implementation of the exFAT file system requires a Microsoft license to the specifications. Licensing may be found at the Microsoft Intellectual Property Licensing for exFAT page. The research in this paper provides an analysis of the exFAT file system including its structure and organization. It is not meant to implement the exFAT file system or any part of it. A static examination is performed of the contents of storage media, and does not attempt to perform any dynamic analysis by direct non-standard

© 2010 The SANS Institute Author retains full rights.

© 2010 SANS Institute, Author retains full rights.

modification to the file system itself. Any file system changes were done via standard drivers and operating system utilities.

4.1.3 Assumptions

The contents of this paper are strictly based on exFAT Version 1.00 as specified in Appendix A of the pending patent Microsoft Patent 0164440.

Unless otherwise specified, all:

- Values are unsigned
- Are stored in little-endian format
- Uses decimal notation for constants, unless specified as a hexadecimal constant
- Specifies hexadecimal constants using the prefix notation of 0x
- Specified character strings within the directory structure as 16-bit Unicode
- Character strings do not require null termination
- When describing the capacity of storage media will use power of 10 terminology
- When describing the capacity of the file system or components will use power of 2 terminology

4.1.4 Out of Scope

Some features have not been released or announced for this version. The information presented here has been limited but hopefully provides enough of the internals for a forensic examiner to get started.

The scope of this project either excludes or minimizes certain analysis that could not be done at this time. Features did not yet exist and there was a need to limit the amount of work being performed due to time constraints.

The following items are limitations or assumptions of this paper:

- Limited to version exFAT 1.00
- File system follows the standards
- exFAT file system NOT installed within a partition
- The file system is assumed NOT broken, NOT corrupt and NOT damaged

© 2010 The SANS Institute Author retains full rights.

© 2010 SANS Institute, Author retains full rights.

- Limited to the examination of removable media such as USB Flash Drives and SD, CF, SM memory cards
- No file system behavioral analysis
- No bad blocks or media failures analysis
- No file system performance analysis
- Analysis is static analysis, not dynamic
- No analysis of data in unallocated space
- No analysis of OEM region in VBR
- No analysis of Volume GUID Entries

The following were not analyzed because these features have not been implemented

- Transactional FAT (TexFAT)
- 2nd FAT
- 2nd Bitmap
- ACL

The unaddressed items listed above are left for further research for anyone wanting to follow-up with any of these specific issues, or analyze new versions of exFAT as they come out in the future.

4.2 Relevance to the Field of Digital Forensics

What will happen when there is an attempt to examine storage media formatted with the exFAT file system on a Windows system that doesn't have exFAT support? Figure 1 shows the disk properties window of an exFAT formatted disk when displayed on such a system.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

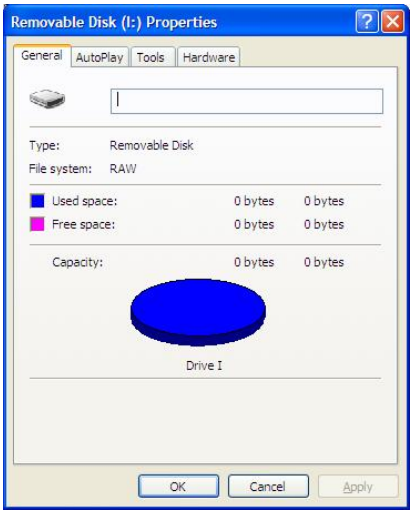


Figure 1 Disk Properties of exFAT file system using Windows XP without exFAT support

There is no information displayed and the operating system indicates that the file system is RAW. Forensics examination is usually performed using either open source tools or commercial tools. Two of the most widely used commercial forensics application tools are EnCase by Guidance Software (http://www.guidancesoftware.com/) and the Forensics Tool Kit (FTK) by Access Data. (http://www.accessdata.com/) (Carlton, 2008). These tools are used on Microsoft Windows operating systems. Currently Microsoft Windows 7 was just released in October 2009, but the two predecessor desktop operating systems are Windows XP and Windows Vista.

However the Microsoft Vista operating system has always seen a resistance of users to migrate from XP to Vista. (Carvey, 2005) (Larkin, 2007) This leaves many users in the field using these tools on Windows XP, and a large user base of these applications are law enforcement or organizations with internal forensics response teams. What does a forensics examiner on a Windows XP machine do when confronted with storage media that the operating system won't recognize? Or even when trying to perform a simple directory command as shown in Figure 2 below (Don't do this with evidence without using write blockers).

© 2010 The SANS Institute Author retains full rights.

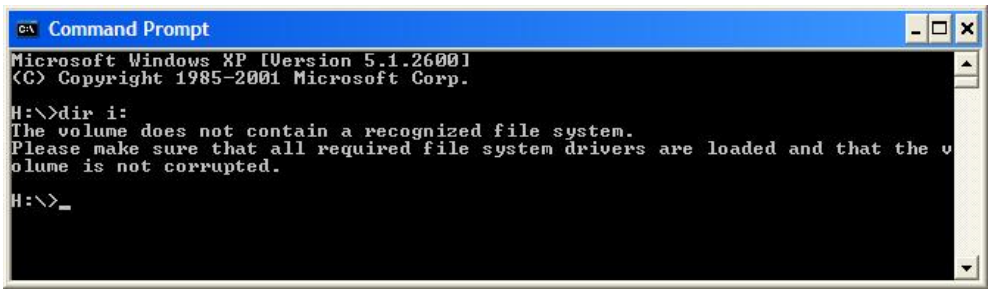


Figure 2 Dir command on Windows XP system without the exFAT drivers

Using Windows Explorer instead, opening the exFAT formatted media on a system that doesn't have exFAT support may result in this message:



Figure 3 Opening exFAT media in Windows Explorer on an XP system without the exFAT drivers

So, the drivers are then installed onto Windows XP, is that enough? Check this out:

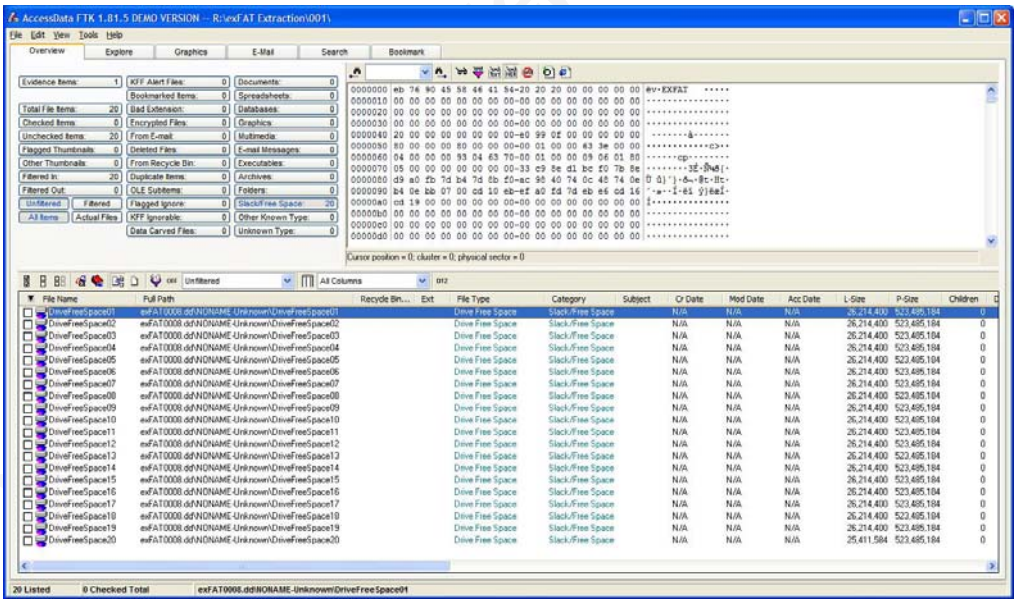


Figure 4 Screenshot of FTK Toolkit 1.81.5 Analysis of exFAT media

The output in Figure 4 displays 20 files all as free space. But because the tool doesn't understand the exFAT file structure, what results is an expensive version of a hex

© 2010 The SANS Institute Author retains full rights.

editor. Files typically have an internal signature (as shown in Figure 5) that can identify the file type. This can be used to recover files when a directory is lost. Although the files could be identified this way, there is an assumption: the file is not fragmented, and the blocks are in proper order.

```
 0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
42 4D 72 16 08 00 00 00  00 00 36 00 00 00 28 00   BMr.......6...(.

 0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
4D 5A 50 00 02 00 00 00  04 00 0F 00 FF FF 00 00   MZP.........ÿÿ..
```

Figure 5 File Signatures of a BMP (Top) and an EXE (Bottom)

Unless the forensics examiner can determine where the blocks are located, the proper sequence order of the blocks, and determine the completeness of the file (are blocks missing?) – Items recovered could be suspect.

If there is digital evidence to be found, will someone take time to look for it? If storage media is collected, and brought to a digital forensics lab, what will happen to it? If the technician at the lab inserts the media in a system and tries to image it, will they bother to continue if the operating system reports back that the media is RAW or corrupt? Will they even be able to acquire an image? Or, will the storage media just get bagged and tagged and added to the evidence pile and never processed?

Suppose the forensics examiner get past that, and manages to acquire the image and at least do some analysis by carving out pieces and analyzing them. What position does this leave the results of a forensics examination when digital evidence has been uncovered and extracted, especially if that evidence is to be used in court?

When examining the expert witness, the tools and the technology will be put through the Daubert guidelines (Daubert v. Merrell Dow Pharmaceuticals). For a file system analysis, procedures that will be scrutinized are those that are used to break one large file system image into the smaller components such as files. (Carrier, 2003)

Taking the position of prosecution and law enforcement, consider the forensics examiner on the witness stand as an expert witness. How would the expert answer questions posed from the defense about what was uncovered if the expert could not understand and describe with authority the file system? The defense may first come up with a question such as “how could you read this media when it shows up on my expert’s machine as unreadable? When the expert gets past that, then “how do you know this file

© 2010 The SANS Institute Author retains full rights.

was deleted or if other files were not mixed in with this one?" This line of questioning is intended to put doubt into the jury or the judge. This approach might not be as effective with media formatted as FAT or FAT32, as these are well known, well documented, and well understood file systems. Until exFAT has been out there a while, accepted, documented, and widely used, the forensic expert will be challenged with addressing that gap.

The fact that file systems are relevant to digital forensics should not require an argument or any discussion. Without any understanding of the file system or organization of the image being acquired it would be difficult, if not impossible, to make any sense of it. Today the world is mostly ASCII, but what if it was a disk from an IBM mainframe that used EBCDIC? Tools that search for ASCII strings won't work. What about the difference between Little-Endian vs. Big-Endian where the byte order makes a difference? All data is binary, ones and zeros, what makes data is the context of those representations.

The exFAT file system has been out for a few years already, why hasn't anyone cared and why will they care now? Many of the current file systems were constrained to 2TiB, although some could handle larger volume sizes. Disks with storage capacities at 2TB used to only be seen in servers and in data centers. Within the past couple of years, buying storage devices with 1TB and 2TB capacities with a price point of less than $200 for use in home desktops was made possible. Storage capacities of the Secure Digital SD cards were achieving 4GB, but a SDHC card was achieving up to 32GB. The SD type cards are used in many portable consumer electronics such as Personal Digital Assistants, Smart Phones, Cameras, and even GPS devices. In 2009, with the announcement of the SDXC media, with supported capacities up to 2TB, the current file systems are going to have a problem keeping up with these expanded capacities and faster I/O speeds.

DVD media today comes in 4.7GB and 8.5GB capacities. Producing a single video file that exceeds the 4GiB file limit of FAT32 is a problem. Use of NTFS can overcome this limitation but NTFS is not designed for removable media. NTFS is also a lazy write system, where NTFS will write data to storage media when it gets around to it. An abrupt removal of the storage media or even in the event of a power failure of the device can leave the file system in an inconsistent state.

© 2010 The SANS Institute Author retains full rights.

NTFS has large overhead with the many components of its file structure. A faster, more efficient file system was needed to exceed the capacities of FAT32 and not have the overhead of NTFS. Microsoft's answer was exFAT, and the file system was released with Windows CE 6.0 in November 2006.

The SDXC standard was announced at the Consumer Electronics Show (CES) in Las Vegas in January 2009. (Hissink, 2009). It is expected to actually see devices that these chips could be used in released by March 2010. Already there has been an announcement that 3 computer manufacturers will have integrated card readers. Lenovo, Hewlett Packard, and Dell have all been fingered as having Arrandale-based laptops in the works for release in early 2010 which will feature integrated SDXC readers. (Halfacree, 2009). Microsoft is driving for wider acceptance of use of the exFAT file system and has expanded its licensing program and already several media card manufacturers have bought into the standard. (Fontana, 2009)

On the software front, there was a December 3rd, 2009 announcement by Diskinternals updating their Uneraser program to support exFAT. (Yahoo News, 2009). Now there is a product on the market that will recover deleted files stored on an exFAT file system. The industry now sees exFAT as a new market for their products to address (Yahoo News (December 3rd, 2009)) because now exFAT will be more viable. When the SDXC devices start being shipped, the need for forensics applications that support the exFAT file system will accelerate. These products were probably needed earlier, but expect that as the SDXC ship dates come closer that more forensics application support for the exFAT file system will be seen.

4.3 Research Methodology

There are many proprietary and not well documented file systems in existence today. The challenge is to take a file system apart and see what makes it tick. The methodology used in this paper to do this exFAT analysis depended on examination of various Microsoft Patents, examination of previous file systems in the FAT family, Google searches, examination of information provided in Microsoft knowledge bases and MSDN, and low level examination of the file system format. Since source code is not available, this all comes down to what is called "black box" analysis (BCS SIGIST, 2001).

© 2010 The SANS Institute Author retains full rights.

The Microsoft Patents relevant to this research paper are:

- Microsoft Patent 0164440 Quick Filename Lookup Using Name Hash
- Microsoft Patent 0265400 Extensible File System
- Microsoft Patent 7613738 FAT Directory Structure for use in Transaction Safe File System

The low level analysis was performed by using a Microsoft Windows XP SP3 laptop and a Microsoft Server 2008 SP1 server (later, during the research, upgraded to SP2) and using these systems to format removable media, such as USB flash drives, Compact Flash, Secure Digital, and Smart Media with the exFAT file system. Then, using the DD tool from the Helix 2008R1 CD-ROM (http://www.e-fense.com), live acquisitions of the drive were taken for analysis. A live acquisition was required because there were issues to get the underlying Linux system to recognize the media in order to image it.

Once the image was acquired, then a copy of Winhex (http://www.x-ways.net/winhex/) was used to go through the file structures. Using a hex editor on a large file over and over again becomes very tedious. To conduct the file system structure analysis a program was written using Microsoft Visual Studio 2003 to develop a C program that would provide formatted printouts of the file system components and metadata.

As the program was being developed, an exFAT file system would be created, and files would be added, deleted, and then added again and images acquired between some of the operations to see what effect the operation had on the file structure. The output of the program was then verified to the output of various operating system utilities such a DIR, CHKDSK, DISK MANAGEMENT, and WINDOWS EXPLORER. In some cases screen shots were taken to be used in this paper and presented as figures.

Results must be verified in order to validate the analysis. Using the native tools listed above and comparing results is the best way to make sure it was done right. Even following the specifications is not enough because the implementation might not exactly follow the specifications.

© 2010 The SANS Institute Author retains full rights.

4.4 Survey of Removable Media

The reader will be given a taste of the history of removable media because the evolution of removable media and the increase of storage capacities has been a driver for a new file system that can support high capacities.

One of the earliest random access removable storage media was the floppy disk, which even pre-dates the PC. There were 8 and 12 inch variations with what would be considered today as low capacities. (History of the Floppy Disk) The last floppy was the 3.5 inch with 1.44MB capacity, although IBM did have a 2.88MB version. In attempts to exceed these limits, Imation (formerly 3M) released their LS-120 drives which took a 120MB style 3.5 inch floppy. This was an attempt to compete with Iomega, which had the Zip drives at 100MB and the JAZ drives at 1GB. The ZIP drives reached higher capacities over its lifetime, at least up to 750MB and the JAZ drives up to 2GB. Before the days of USB, the Zip drives connected via a parallel cable and the JAZ via SCSI cables. Internal IDE and SCSI versions of these drives were also available.

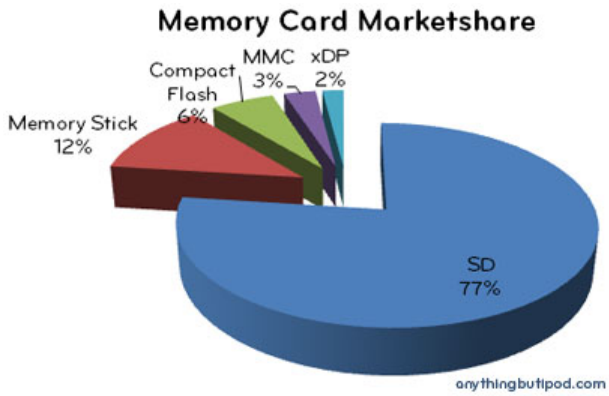
When compact disk (CD) started to become available, it provided a much larger storage media with capacities over 600MB. As programs and operating systems became larger it was more affordable to provide the distribution of these on CDROM. For example, a software product could require dozens of 1.44MB floppy disks to do an install. A 650MB CDROM could hold the data of 450 floppies. Today, some software products may be released on multiple CDROM discs or even on DVD now. (History and Capacities of CDROM) (History and Capacities of DVD) CDROM has killed the floppy, as many workstations and laptops either don't ship with floppy drives anymore or the floppy drive comes as a separate USB attachment. With the CD-R and CD-RW media, these media are writable, and provide more storage space than floppies. The DVD which is replacing the CDROM comes in 4.7GB and 8.5GB versions, but the Dual Layer (8.5GB) is not that popular yet. Another disc format replacing the DVD may be the Blu-Ray Disc, with storage capacities of 25GB and 50GB. (History and Capacities of Blue Ray Disc) The larger capacity is to support High Definition video which requires more storage because there is higher resolution and requires more digital storage.

© 2010 The SANS Institute Author retains full rights.

Another storage line for removable media is Compact Flash, Smart Media, Secure Digital, and Memory stick (Figure 6). These media types are common for use in cameras, and have been used in PDA, Cell Phones, and even GPS devices.



Figure 6 Compact Flash, SDXC, and Smart Media and SD cards



source: http://www.anythingbutipod.com/archives/2009/01/next-generation-sdxc-details.php
Figure 7 Jan 2009, Memory Card Market Share,

Compact flash has achieved 128GB capacities; although some of it may be flash and some of it actual micro disk drives. Smart media which has been discontinued achieved 128MB capacities. The SD cards, specifically the SDHC has a capacity range 4GB-32GB. The SD card market dominates the market share (see Figure 7) and if it continues to hold that share it may become the largest driver towards exFAT use.

So the common theme so far is capacities of storage media going to 32GB, with the exception of the Compact Flash which is getting to 128GB and beyond.

Today there are USB flash drives with capacities now up to 256GB on a stick. Although this is interesting, it's the compact flash and SD cards that are more common to digital still and motion cameras. And the new SDXC card, with a capacity range of 32GB-2TB could give the SD association the ability to surpass the compact flash association.

Robert Shullich rshullic@earthlink.net

© 2010 The SANS Institute Author retains full rights.

Since the new SDXC cards will support exFAT (indications that SDXC will support FAT32 were not observed), if SDXC is successful with exFAT, it will push exFAT out into the wide open. And with some laptop manufacturers announcing that they will build SDXC card readers into the laptop itself, maybe SDXC will be the new floppy. Integrated compact flash and SD card readers are not new. For example the Dell 24 inch monitor has built-in slots for these media cards. Many photo printers have card reader slots that allow the printer to print directly off the media cards and allow the connected computer direct access to the cards used in these integrated slots.

Integration into the desktop or laptop system is only the next logical step. As it becomes easier to use and access these forms of media, the higher the potential that this media may be used to store something that will eventually become digital evidence.

4.5 Survey of Microsoft File Systems

The FAT file system originated in the late 1970's with the MS DOS Operating System. The system has evolved over the years with the file systems FAT12, FAT16, and FAT32 and now, the new member of the FAT family exFAT. FAT is a simple file system organization and is ideal for removable media where quick removal of the media is required. Almost every operating system since MS DOS recognize the FAT12 and FAT16 file systems and almost every operating system since Windows 98 recognize the FAT32 system. These file systems are also used in consumer electronics such as cell phones, PDA's, and GPS devices. The FAT file system is lightweight without many features or file system overhead. Microsoft recommends FAT for flash media.

The NTFS file system was created for the enterprise and for use in Windows NT Servers and Workstations. Prior to NTFS Microsoft supported two file systems, the FAT file system and the HPFS (High Performance File System). HPFS was used in OS/2 and Warp, an operating system that was a joint venture between Microsoft and IBM. HPFS was also used in earlier versions of Windows NT. NTFS provides many features that include fault tolerance, speed, security, larger file sizes and space optimization. NTFS is not designed for removable media, because it uses a lazy write scheme and is slower to write to a disk than FAT. In a lazy write system output operations are queued and might be delayed as I/O is overlapped. Disengaging the removable media before the writes have completed could leave the file system in an inconsistent state and could become corrupted. NTFS also provides encryption and compression for files and folders. Although NTFS is only supported on a Windows NT type of system (Windows 2000, Windows XP, Windows

© 2010 The SANS Institute Author retains full rights.

Vista, Windows 7), there were some OEM drivers available to allow systems such as Windows 95 and Windows 98 access to a NTFS volume. Drivers for NTFS access may also be found on some Linux systems. Microsoft recommends NTFS for fixed disk media.

The UDF file system is used for optical media such as CD and DVD. It has high portability because it uses an ISO standard and can be read by many different file systems and used in consumer electronics. UDF has many features and limitations of NTFS. Some features, such as Alternate Data Streams (ADS) is provided by UDF but not supported in all the Microsoft UDF drivers. (Microsoft, 2004)

4.6 Getting the drivers put onto Windows XP

In order to inspect the file system using the native Windows XP operating system commands XP support of the exFAT file system must be added. This is achieved by going to the Microsoft support site and downloading the KB955704 update that adds exFAT support. Invoke the update, accept the terms, and then reboot your XP system.



Figure 8 Step 1 – Invoke Update KB955704

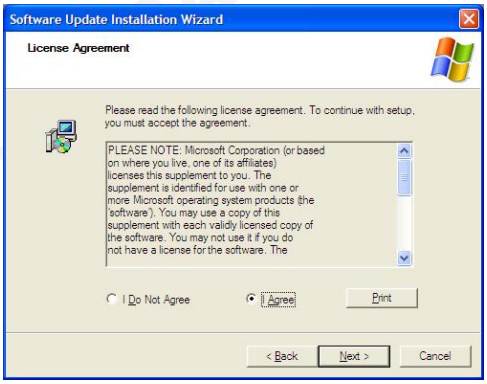


Figure 9 Step 2 – Agree to the License Agreement

© 2010 The SANS Institute Author retains full rights.

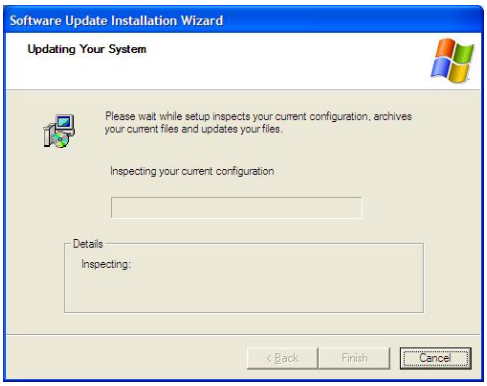


Figure 10 Step 3– KB955704 begins to update

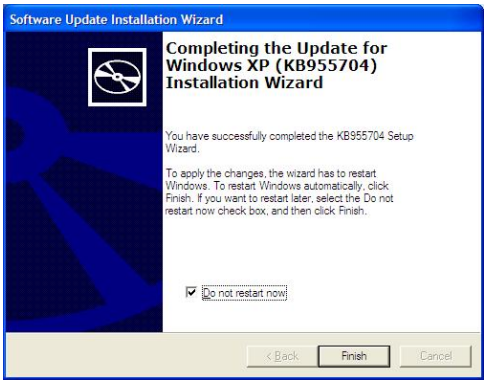


Figure 11 Step 4 – KB955704 Completed, now reboot the system

After rebooting, you should now have exFAT support. One of the easiest ways to see if the update took is to bring up a command window and do a FORMAT /? To get help. The output is shown in Figure 12.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

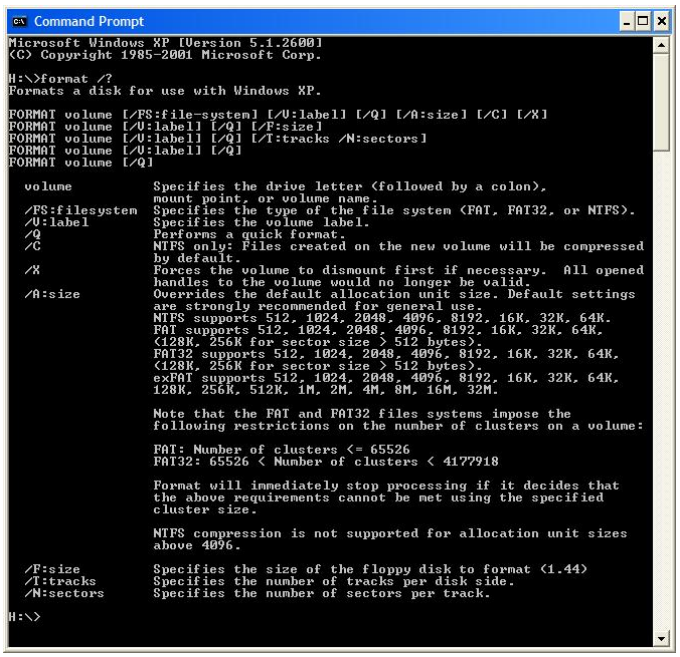


Figure 12 Format Help command on XP after KB955704

This may be a little deceiving because exFAT is not listed as a file system in the /FS: option. But under the /A: option it shows you the exFAT supported blocksizes which at least indicates that the format program was updated.

4.7 International System of Units (SI) Table

Shorthand	Longhand	Nth	Bytes
KiB	Kibibyte	2^{10}	1024
MiB	Mebibyte	2^{20}	1024 KiB
GiB	Gibibyte	2^{30}	1024 MiB
TiB	Tebibyte	2^{40}	1024 GiB
PiB	Pebibyte	2^{50}	1024 TiB
EiB	Exbibyte	2^{60}	1024 PiB
ZiB	Zebibyte	2^{70}	1024 EiB
YiB	Yobibyte	2^{80}	1024 ZiB

Table 1 Numbering Schemes

This paper will get into some very large volume and file sizes and Table 1 can be used as a reference. Since file systems will be reviewed and this paper addresses recording media, the definitions have always been confusing in the past. For example, the common terminology from the metric system was that "kilo" meant 1,000 but in computer speak a kilobyte was always 1024. When a disk manufacturer releases

© 2010 The SANS Institute Author retains full rights.

specifications of a disk drive, there would be a disclaimer that 1 Megabyte = 1,000,000 bytes. So, current terminology is now being used to differentiate between 1,000 and 1,024 by using "kilo" for 1,000 and "kibi" for 1024. This paper will address most of the sizes as a power of 2, and will be using this different terminology.

The size prefixes are explained in International System of Units (SI) which also gives more information and references other publications where this naming scheme is used. In their explanation the older prefixes were a power of 10 where these new prefixes are a power or 2. Notice that each name has "bibyte" meaning Binary Byte.

4.8 Summary of exFAT Features

- Sector sizes from 512 to 4096 bytes
- Clusters sizes to 32MiB
- Subdirectories to 256MiB
- Built for speed, less overhead than NTFS but has some of the NTFS features
- TexFAT (To be released later)
- ACL (To be released later)
- UTC Timestamp Support
- OEM Parameters Sector for device dependent parameters
- 9 sector VBR, support of larger boot program
- Potential capacity to 64ZiB
- Up to 2796202 files per subdirectory

4.9 exFAT Timeline (Key Dates)

- September 2006 – Windows CE 6.0 (HPC Factor (2009))
- March 2008 – Notable Changes in Windows Vista Service Pack 1 (Microsoft (2008))
- January 2009 – Announcement at CES of SDXC specification (Hissink, 2009)
- January 2009 – Windows XP Drivers Available (Microsoft, 2009)
- August 2009 – Tuxera Signs File System IP Agreement with Microsoft (Galli, 2009)
- March 2009 – Pretec Releases first SDXC Cards (Herrman, 2009)

© 2010 The SANS Institute Author retains full rights.

- December 2009 – Microsoft announces exFAT license program for third-parties (Microsoft Press Pass, 2009) (Johnston, 2009)
- December 2009 – SDXC laptops due soon (December 2009) (Halfacree, 2009)
- December 2009 – Diskinternals releases exFAT recovery utility (Yahoo News, 2009)

4.10 Maximum Volume and File Limitations

	FAT12	FAT16	FAT32	NTFS	UDF	exFAT
Max Volume Size	32MiB	2GiB[6]	2TiB[5]	16EiB	2TiB[4]	128PiB[1]
Max File Size	4GiB[7]	4GiB[7]	4GiB[7]	16EiB[2]	16EiB[2]	16EiB[2]
Complexity / Performance	Low	Low	Low	High	Low	Low
Fault Tolerance	No	No	No	Yes	No	Yes[3]
Object Permissions	No	No	No	Yes	No	Yes[3]
Max File Name Length	255	255	256	256	127 Unicode or 254 ASCII	255 Unicode

Comments:
[1]The maximum exFAT Volume size is specified as 2^{32} clusters by a maximum cluster size of 2^{25} (32MB) which is 2^{57}. There is a published theoretical maximum is 64ZiB which is 2^{76}, leaving a cluster size of 2^{44} (16TiB [2^{76}-2^{32})]. The specification in the patent has set an implementation limit of $2^{25}$for the cluster size. The maximum sector size is 4096 (2^{12}).
[2]The maximum file size is 2^{64}-1 which is a theoretical maximum and currently exceeds the size of the volume.
[3]This feature may be supported in a future release
[4]2TiB at 512 block size, 8TiB at 2Kib block size
[5]A maximum disk size of 8TB could be supported for a cluster size of 32KiB.
[6]4GiB for block with 64KiB clusters
7The maximum file size is 2^{32}-1 which is a theoretical maximum and currently exceeds the size of the volume.

Table 2 File System Limits

Table 2 makes an attempt to provide a comparison of some common file systems used on Microsoft systems. Information from the MSDN Library (Microsoft MSDN EE681827) was used to build part of this table, and some use the terminology of blocks while others call them clusters. exFAT will use the terms sectors (for the physical block) and clusters (for the logical block). This was a difficult table to generate, and as you can see there are many footnote exceptions. The problem is that the actual implementation may differ based on the operating system used.

If the file system is put into a partition, where a MBR record is required, the maximum volume size is also limited, usually to less than the theoretical limits of the

© 2010 The SANS Institute Author retains full rights.

volume capacity of the file system. As seen in Table 19, the field for number of sectors is a 4 byte field, limiting the number of physical sectors, to 2^{32}. A file system like exFAT which can have 2^{64} sectors would be constrained in its maximum volume size. In order to take advantage of the full capacity limits of the file system, it would need to be configured as a superfloppy where it can escape the limits of the partition MBR.

There is confusion and disagreement on the maximum size of the exFAT volume, with many theoretical limits expressed. This section will attempt to demystify some of these limits.

A Microsoft Knowledge Base article (Microsoft, 2009) states a theroetical maximum volume size of 64ZiB. As seen on other web sites, such as NTFS.COM with their file system comparison called *NTFS vs. FAT* also indicates the maximum volume size of 64ZiB. This number is the result of the following calculations: Microsoft has imposed an implementation limit of 2^{12} (4096 bytes) as a sector size limit. In the VBR, as seen in Table 3 is an 8 byte number than can specify up to 2^{64} sectors. This can result in a volume space of 2^{76} bytes, which is 64ZiB. This sounds good on paper, but there is one slight catch, since the FAT cell entries are 32 bits in size and can address at most 2^{32} clusters, this would require a cluster size of 2^{44} bytes, or 16TiB. Think about that, 16 tebibytes for ONE cluster. With the exception of high end server file systems, you will rarely see an entire file system being that large, and this is just one block.

The current exFAT implementation's maximum is smaller, and is 128PiB. Here is how this value is calculated: Microsoft has limited the maximum cluster size to 2^{25} bytes (32MiB). This number is reached by multiplying the sector size by the number of sectors per cluster. The sector size may be defined between 2^9 (512 bytes) and 2^{12} (4096 bytes) and the product of these 2 values cannot exceed 2^{25} bytes. Next, the FAT entries are examined which are 4 bytes and can track 2^{32} of these clusters. This calculates to a maximum volume space of 2^{57} bytes (128PiB).

Microsoft in their recent licensing announcement states "support from 32GB to 256TB" (Microsoft Press Pass, 2009). This new stated limit is 2^{48}. The origins of this new limit is currently unknown, but if you take the maximum sector size which is 2^{12} and block it one sector per cluster, at 2^{32} clusters, you will get 2^{48}. Analysis of the exFAT file system was performed on different storage media 500MB or less, so 32GB is not a lower

© 2010 The SANS Institute Author retains full rights.

limit either. Unlike FAT32 that requires a minimum number of clusters to be configured, it appears that exFAT does not have that restriction.

Microsoft has put a practical limit on the sector and cluster sizes. The maximum theoretical limits on the cluster size without these limits are a maximum of 4^{255}.bytes per cluster, which is a really large number.

When examining the maximum file size, the storage location within the directory entries of the exFAT file system is an eight byte non-signed integer which can hold a value up to 2^{64}-1. This value exceeds the maximum volume size based on the current specifications, and as currently implemented the maximum file size is constrained by the configured size of the file system.

5 exFAT Internals

5.1 Volume Structure

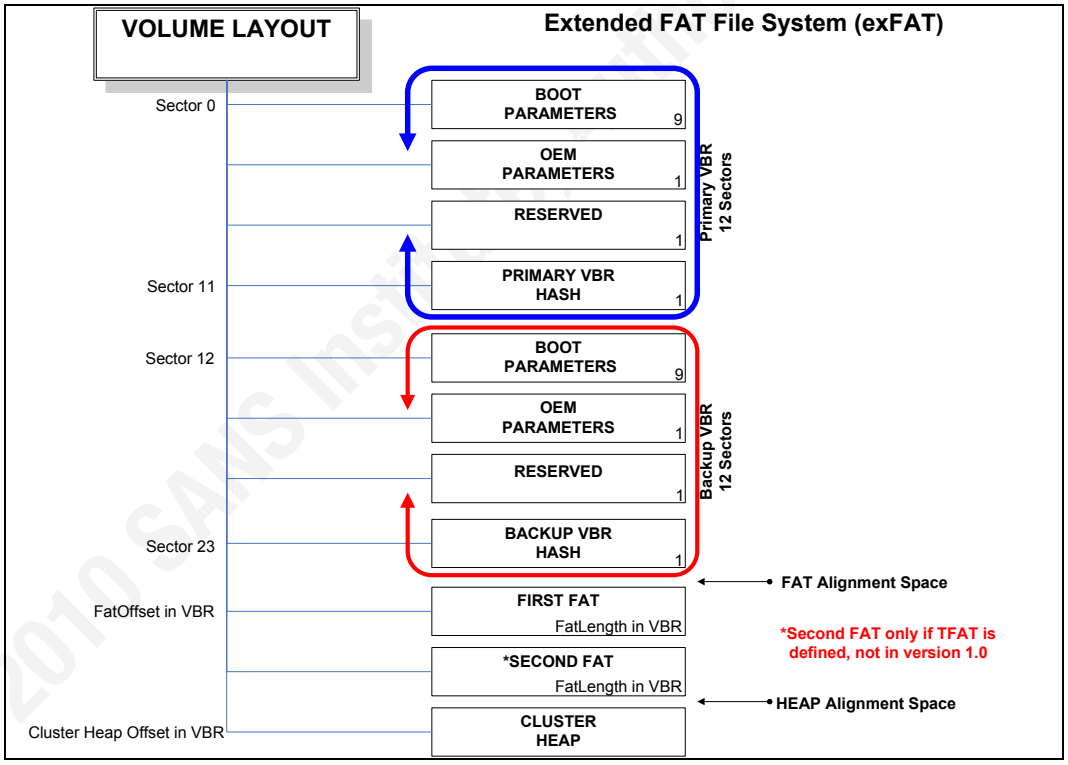


Figure 13 Extended FAT File System (exFAT) Volume Layout

© 2010 The SANS Institute Author retains full rights.

The exFAT specification defines the volume layout as regions, and defines four regions:

- The Main Boot Region
- The Backup Boot Region
- The FAT Region
- The Data Region

There are also sub regions which will be explained in later sections. A volume layout is shown in Figure 13.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

5.2 Volume Boot Record (VBR)

Field Name	Offset (byte)	Size (byte)	Description/Value
Jump Boot	0	3	0xEB7690
File System Name	3	8	"EXFAT "
Must Be Zero	11	53	Must be 0x00
Partition Offset	64	8	Sector Address
Volume Length	72	8	Size of total volume in sectors
FAT Offset	80	4	Sector address of 1st FAT
FAT Length	84	4	Size of FAT in Sectors
Cluster Heap offset	88	4	Sector address of the Data Region
Cluster Count	92	4	Number of clusters in the Cluster Heap
Root Directory First Cluster	96	4	Cluster address of the Root Directory
Volume Serial Number	100	4	Volume Serial Number
File System Revision	104	2	VV.MM (01.00 for this release)
Volume Flags	106	2	Field / Offset bits / Size bits / Description Active FAT / 0 / 1 / 0 – 1st, 1 – 2nd Volume Dirty / 1 / 1 / 0 – Clean, 1 - Dirty Media Failure / 2 / 1 / 0 – No Failures, 1 – Failures Reported Clear to Zero / 3 / 1 / No Meaning Reserved / 4 / 12 /
Bytes Per Sector	108	1	This is a power of 2. Range: min of 2^9 = 512 byte cluster size, and a max of 2^{12} = 4096.
Sectors Per Cluster	109	1	This is a power of 2. Range: Min of 2^1=512. The maximum Cluster size is 32 MiB, so the Values in Bytes per Sector + Sectors Per Cluster cannot exceed 25.
Number of FATS	110	1	This number is either 1 or 2, and is only 2 if TexFAT is in use.
Drive Select	111	1	Used by INT 13
Percent In Use	112	1	Percentage of Heap in use
Reserved	113	7	
Boot Code	120	390	The Boot Program
Boot Signature	510	2	0xAA55
Excess	512		If the sector is larger than 512 bytes, extra padding may exist beyond the signature
Comments: Volume size is minimum of 1MB and maximum size is 2^{64-1} sectors			

Table 3 Layout for Main and Backup Boot Sector Structure

The Volume Boot Record (VBR), as shown in Table 3, is the first critical file system collection of metadata needed by the forensics examiner. This collection of sectors

© 2010 The SANS Institute Author retains full rights.

defines the limits and locations of the exFAT regions, and has a pointer to the Root Directory.

The Main Boot Region of the VBR is composed of five sub-regions of a total of 12 sectors:

- The Main Boot Sector (MBS)
- The Main Extended Boot Sectors (MEBS)
- The OEM Parameters
- A reserved sector
- The Checksum Sector

The Backup Boot Region is a repeat of the 12 sectors found in the Main Boot Region, and together, both regions total 24 sectors. Since the concept of the term cluster only applies to the contents of the Cluster Heap, the VBR will always be expressed as sectors.

The MBS does not differ conceptually from the partition Master Boot Sectors or Volume Boot Records of previous FAT file systems. It contains Boot Code, the BIOS Parameters Block (BPB), and a signature. The purpose of the BPB is to describe the physical layout of the file system volume. The common signature (as shown in Figure 14) used in this sector is 0xAA55.

```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
000001F0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA   ..............Uª
```

Figure 14 Winhex Display of VBR Signature

The Boot Code is located in the first 3 bytes of the MBS and also consumes 390 bytes at offset 120. The first 3 bytes is a Jump Boot sequence which bypasses the BPB and jumps to the Boot Code. Since any executable sequence of computer instruction may be stored in the boot code, this may be of interest to the forensics examiner should customized boot code be stored. It would be in this area of the sector that a Boot Sector Virus would modify and implant itself.

© 2010 The SANS Institute Author retains full rights.

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

       0   EB 76 90 45 58 46 41 54  20 20 20 00 00 00 00 00   ëv EXFAT   .....
      16   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
      32   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
      48   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
      64   3F 00 00 00 00 00 00 00  C1 F3 01 00 00 00 00 00   ?.......Áó......
      80   80 00 00 00 80 00 00 00  00 01 00 00 58 3E 00 00   €...€.......X>..
      96   05 00 00 00 EC 99 D1 C4  00 01 00 00 09 03 01 80   ....ì™ÑÄ.......€
     112   5C 00 00 00 00 00 00 00                           \.......
```

Figure 15 Winhex of the first 120 bytes of a MBS

The File System Name, also referred to as the OEM label, is an 8 byte ASCII field containing the name of the file system. This makes identification of the file system easier, and as shown in Figure 15 the name is "EXFAT" and is padded with training blanks. If this file system is created on a fixed hard drive in a partition, you cannot rely on the partition type within the MBR to determine the file system type because the partition code for exFAT is 0x07 and is shared with other file systems (see Table 20). The next field, Must Be Zero, defines 53 bytes of 0x00 in a location that the older FAT file systems used to define their BPB. This reduces the risk of the legacy FAT implementations of accidently mounting an exFAT volume by mistake.

The Volume Length is a count of the total number of sectors on the volume. This number needs to be larger than a 32 bit number, so it is defined as 2^{64}. Suppose for example the maximum sized Cluster Heap was defined, which is currently limited to 2^{32}-11 clusters. If the cluster ratio is set to 1 sector per cluster (1:1), then a 32 bit number is required to hold the volume length. If the sector to cluster ratio was 1:16, then a 36 bit number would be required. If the current maximum as per the specification were used, and assuming a sector size of 512 bytes, an additional 16 bits need to be added, requiring a 48 bit number. This is based on a 25 bit maximum (32MiB cluster size) and 9 of those bits are used to define 512 bytes for the sector size.

Four fields are used to describe the FAT. The FAT offset is used to define the sector offset of the FAT region and points to the 1st FAT. If the number of FATS is 2, then the 2nd FAT will immediately follow the 1st FAT, starting on a sector boundary. The number of FATS will always be 1 because TexFAT is not implemented and the 2nd FAT only exists in a TexFAT environment. If the implementation does not verify this value, then the file system could be modified to increase this number and imbed fake FATS in the volume in order to create additional slack space to hide data. The FAT length is the

© 2010 The SANS Institute Author retains full rights.

length of the FAT in sectors. In the Volume Flags there is a flag for the Active FAT. This only applies in a TexFAT environment, when number of FATS is equal to 2. This flag indicates which of the two FATS is active.

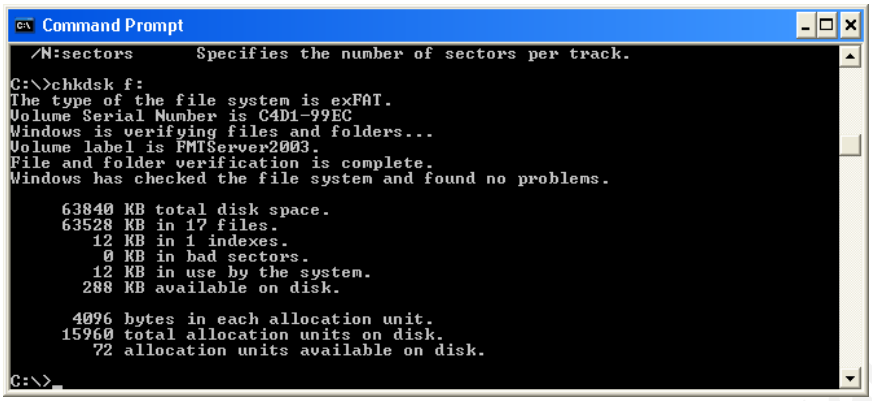


Figure 16 Chkdsk of an exFAT formatted disk

The final region, the Cluster Heap, is the data portion of the volume structure and holds the directories and files. The Cluster Heap is allocated in cluster units and the Cluster Count defines how many allocation units are defined. The Cluster Offset identifies the sector address of where the Cluster Heap begins. Once inside the Cluster Heap, the addressing units are in clusters. In Figure 16 the Cluster Count is shown as *total allocation units on disk*, and in this example shows 15,960.

In comparison, a FAT32 file system requires a minimum of 65,526 clusters making FAT32 unusable for small disks formats. exFAT does not have that restriction and smaller media may be used. In testing, a 32MB compact flash card was formatted as an exFAT file system.

A key value in this sector for the forensics examiner is the Root Directory First Cluster. The details of the Root Directory are described in section 6.1, and this value points to the first cluster of the Root Directory which resides in the Cluster Heap. The VBR defines the structure of the volume, but the Root Directory defines the contents within the Cluster Heap. All the metadata about files, subdirectories, the volume label, etc reside in this directory.

Two critical fields are the bytes per sector and sectors per cluster. One thing that is special about these fields is that the values contained are exponents. For example Figure 15 shows that the bytes per sector are 9 and the sectors per cluster are 8. This is 2^9

© 2010 The SANS Institute Author retains full rights.

bytes per sector (512) and 2^3 sectors per cluster (8) resulting in a cluster size of 4096 bytes. The maximum aggregate sum of these two exponents is 25, for a maximum cluster size of 32MiB. The maximum value for the bytes per sector field is 12 (2^{12} = 4096 bytes).

At offset location 104 is the file system revision number, which appears in Figure 15 and is 0x0100 and translates to version 01.00.

The boot signature of the MBS is always at offset location 510. If the sector size is defined as greater than 512 bytes, the signature will still be located at this location, and the remainder of the sector will be undefined and not used.

Field Name	Offset (byte)	Size (byte)	Description/Value
Extended Boot Code	0	508-4092	Additional Boot Code
Extended Boot Signature	508-4092	4	0xAA550000
Comments: Signature actually stored as 0x000055AA			

Table 4 Layout for Extended Boot Sector Structure

The Main Extended Boot Region takes up the next 8 sectors, even when not used. This allows a larger boot program by providing additional sectors for boot code. Unlike the MBS, the MEBS, when extended to larger than 512 bytes, allows usage of the entire sector for boot code and the record signature is moved to the last four bytes. If a sector size of 4096 bytes was used, the boot signature would be at offset 4092. If a MEBS sector is not in use, the boot code should all be 0x00, followed by the boot signature.

Field Name	Offset (byte)	Size (byte)	Description/Value
Parameters[0]	0	48	Parameters
Parameters[1]	48	48	Parameters
Parameters[2]	96	48	Parameters
Parameters[3]	244	48	Parameters
Parameters[4]	192	48	Parameters
Parameters[5]	240	48	Parameters
Parameters[6]	288	48	Parameters
Parameters[7]	336	48	Parameters
Parameters[8]	384	48	Parameters
Parameters[9]	432	48	Parameters
Reserved	480	32-3616	Rest of sector Reserved

Table 5 Layout for OEM Parameter Structure

The next sector in the VBR (sector 9) is the OEM parameters record. Since this record really doesn't exist yet (it is all zeros in the file systems that were generated), there

© 2010 The SANS Institute Author retains full rights.

is little analysis that can be done at this time. The patent specifies this table as 10 fields of 48 bytes, the first 16 bytes of each field is the GUID and the remaining 32 bytes are the parameters, but no additional definition is provided.

The entries are not sorted, and it is possible that the first 9 are empty and the last has data, so the specification states that all 10 entries should be searched. This sector is filled out by the media manufacturer at the factory and a format operation is not supposed to erase this sector with the exception of a secure wipe of the media.

Examination of Microsoft MSDN AA914663 provides a definition of the 32 byte parameter field, as shown in Figure 17:

```
struct
{
   GUID OemParameterType; //Value is OEM_FLASH_PARAMETER_GUID
   UINT32 EraseBlockSize; //Erase block size in bytes
   UINT32 PageSize;
   UINT32 NumberOfSpareBlocks;
   UINT32 tRandomAccess; //Random Access Time in nanoseconds
   UINT32 tProgram; //Program time in nanoseconds
   UINT32 tReadCycle; // Serial read cycle time in nanoseconds
   UINT32 tWriteCycle; // Write Cycle time in nanoseconds
   UCHAR Reserved[4];
}
FlashParameters;
```

Figure 17 OEM Parameters Type Definition

Sector 10 is reserved, and is not currently defined. Sector 11 is a checksum sector, where every 4 byte integer is a 32 bit repeating checksum value of the previous 11 sectors. If anyone wanted to tamper with the VBR by changing values in the BPB or the boot code, like a boot sector virus infecting the VBR, then the checksum would have to be recalculated and sector 11 would need to be updated. The last 3 sectors of this 12 sector VBR (sectors 9, 10 and 11) do not contain signatures, the signatures are only used for sectors containing boot code and are in the first 9 sectors.

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
    5632   7D 0A 4E 29 7D 0A 4E 29  7D 0A 4E 29 7D 0A 4E 29   }.N)}.N)}.N)}.N)
    5648   7D 0A 4E 29 7D 0A 4E 29  7D 0A 4E 29 7D 0A 4E 29   }.N)}.N)}.N)}.N)
```

Figure 18 Winhex dump of part of a VBR checksum sector

Figure 18 shows a partial dump of the checksum sector, the checksum is 0x294E0A7D and repeats in every 4 bytes of the entire sector. For a sector size of 512

© 2010 The SANS Institute Author retains full rights.

bytes, it would repeat 128 times. Figure 19 shows the Microsoft Visual C function that was used to compute and verify the checksum value.

```
UINT32 VBRChecksum(const unsigned char octets[], long NumberOfBytes)
{
  UINT32 Checksum = 0;
  long Index;

  for (Index = 0; Index < NumberOfBytes; Index++)
        {
                if (Index == 106 || Index == 107 || Index == 112)
                {
                        continue;
                }
                Checksum = ((Checksum <<31) | (Checksum>> 1)) + (UINT32) octets[Index];
        }
  return Checksum;
```

Figure 19 Code snippet of VBR checksum calculation function in C

For comparison, the FAT32 VBR is within a reserved 32 sector region, with a primary VBR of 3 sectors at sectors 0, 1 and 2 and then a backup VBR located at sectors 6, 7 and 8. (Mueller, 2003) In a FAT32 VBR, executable boot code can reside in the 1st and 3rd sectors, where an exFAT VBR can have 9 sectors containing executable code.

Figure 49 shows a formatted dump using a Winhex template to display the 1st sector VBR. This template doesn't currently exist because it was developed as part of this research, but the source code for the template is provided in Figure 48.

5.3 File Allocation Table (FAT)

Field Name	Offset (byte)	Size (byte)	Description/Value
FAT Entry [0]	0	4	Media Type 0xFFFFFF8 Hard Drive
FAT Entry [1]	4	4	Constant 0xFFFFFFFF
FAT Entry [2]	8	4	First Cluster
Last FAT Entry	(Cluster Count +1) * 4	4	Last Cluster
Free Space	(Cluster Count +2) * 4	Remainder of Sector	What is left over of the last sector
Comments: The First cluster is cluster 2, there is no cluster 0 or 1. If there were 10 clusters (Cluster Count) the clusters would be numbered from 2 to 11, and the entire FAT would be 12 entries of 32 bits each.			

Table 6 Layout for the File Allocation Table (FAT)

exFAT is in the FAT family of file systems along with FAT12, FAT16 and FAT32. In explaining the FAT file system, Carrier, 2005 on page 260 explains the two purposes of the FAT, one being to determine the allocation status of a cluster and the other is to find the next allocated cluster in a file or directory cluster chain. In the exFAT file system these responsibilities change.

© 2010 The SANS Institute Author retains full rights.

In exFAT, the FAT is no longer used for allocation status. Like NTFS, exFAT will use a bitmap to keep track of the cluster allocation status. As far as where the next cluster resides, the FAT in the exFAT file system will work similar to previous FAT file systems when the file is fragmented. If the file or directory becomes fragmented then the FAT will need to be used to track the location of the clusters.

Theoretically, this change has the potential of speeding up I/O operations. When writing on a FAT32 file system the FAT must be accessed when each cluster is allocated or read. In exFAT this operation is flipping a bit in the Allocation Bitmap. As long as the file remains not fragmented, the FAT does not need to be updated. Even if there is data already in the FAT, those corresponding cells don't even need to be zeroed because there is a flag in the Stream Extension Directory Entry (Section 6.9) that indicates that the FAT is invalid. Read operations of a non-fragmented file stored in exFAT would not require access to the FAT or the Allocation Bitmap and reduces overall file system I/O overhead.

The FAT uses a singly linked list to track the location of clusters. A singly linked list is an object with a key and a next pointer, but does not have a previous pointer like found in a doubly linked list. (Cormen, Leiserson, Rivest & Stein, 2001). In the case of a FAT, an array is used and the cell location is the key. The contents at that cell are the next pointer.

Byte	Capacity	Media Size and Type
F0	2.88 MiB	3.5-inch, 2-sided, 36-sector
F0	1.44 MiB	3.5-inch, 2-sided, 18-sector
F9	720 KiB	3.5-inch, 2-sided, 9-sector
F9	1.2 MiB	5.25-inch, 2-sided, 15-sector
FD	360 KiB	5.25-inch, 2-sided, 9-sector
FF	320 KiB	5.25-inch, 2-sided, 8-sector
FC	180 KiB	5.25-inch, 1-sided, 9-sector
FE	160 KiB	5.25-inch, 1-sided, 8-sector
F8		Fixed disk

Table 7 Media Descriptor Definitions as used in legacy FAT file systems

The first two cell entries of the FAT table are predefined. The first entry is the media type, and is set to 0xF8 which signifies a fixed disk. FAT12 and FAT16 systems are capable of supporting floppy disks, and other media values were available (see Table

© 2010 The SANS Institute Author retains full rights.

7). exFAT does not support floppy disks. Figure 20 shows the results of trying to format a 3½ inch floppy disk using a Server 2008 system with the exFAT support installed.

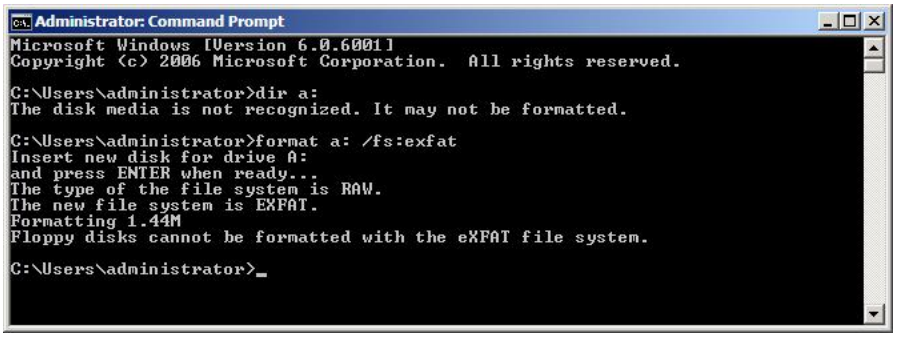


Figure 20 Attempt to format a 1.44 floppy disk with an exFAT file system

```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F

00020000   F8 FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   øÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00020010   FF FF FF FF 00 00 00 00  00 00 00 00 00 00 00 00   ÿÿÿÿ............
00020020   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00020030   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
```

Figure 21 Winhex display of 16 FAT cells

The Winhex display in Figure 21 shows the first 16 locations of a specific exFAT file system. The first location has 0xFFFFFFF8 which is the media descriptor. The second location is 0xFFFFFFFF. Both of these relative locations in an exFAT file system should always have those same 2 values in them.

The first cluster in an exFAT file system is at index 2, which works out nice since the first two cells are reserved and do not represent any clusters. In this same example, clusters 2, 3 and 4 have EOF markers, i.e. 0xFFFFFFFF. In the file system that is being used for this example, cluster 2 is the first cluster of the Allocation Bitmap, cluster 3 is the first cluster of the UP-Case Table, and Cluster 4 is the first cluster of the Root Directory. All three of these areas are created as part of the format operation to initially create the file system and are not part of normal file system operations.

Although the UP-Case Table and Allocation Bitmap sizes should be static and those areas should not be fragmented, the Root Directory can grow and could fragment. Section 5.2 provides the definition of the VBR where the Root Directory first cluster pointer resides, it should be noted that there is no equivalent of a No FAT Chain flag, and as will be seen in sections 6.3 and 6.4, the directory entries for the Allocation Bitmap and UP-Case Table do not have this flag as well. Without an indication of whether the FAT is

© 2010 The SANS Institute Author retains full rights.

used to chain clusters, the assumption is that they are chained and the FAT for these 3 areas will operate as in legacy FAT implementations.

```
Printing Simulated chkdsk totals

    131072 bytes in each allocation unit.
      497 Total allocation units on disk.
       61 Allocation units available on disk.
      436 Allocation units in use.

Analyzing 1st FAT
FF (End Of Chains): 3     F7 (Bad Clusters): 0     Cell Contains Zero: 494     NonZero (Remaining Non-Zero Cells):     0
```

Figure 22 Program simulated Chkdsk totals

As part of analyzing the FAT structure, the program reads all the FAT cell entries (excluding the first two reserved cells) and provides a count of them as shown in Figure 22. As shown in Figure 21, there are 3 EOF markers. That particular 64MB USB drive was 87 percent full with 3 very large files on it, so that only 61 clusters were still free. Using 128KiB cluster sizes, there were only 497 clusters created in that file system, and when the program counted both zero and non-zero cells, the contents of 494 FAT entries contained zeroes. The other 3 non-zero entries were those 3 EOF markers.

What this shows is that even though this particular USB drive had 436 clusters allocated and in use, the FAT table wasn't used to record anything except for 3 of the clusters. This verifies that the FAT isn't used to represent all file cluster chains. This theoretically should have less of a performance impact on reading a file compared to when using one of the FAT predecessors. If a FAT chain is not generated for a contiguous set of file blocks then the FAT does not need to be consulted. This can reduce I/O to read the FAT entries and may simplify I/O operations where multiple blocks can be read at once.

There are a few special values that relate to the FAT:

- 0x00000000 – No significant meaning
- 0x00000001 – Not a valid cell value
- 0xFFFFFFF6 – Largest Value
- 0xFFFFFFF7 – Bad Block
- 0xFFFFFFF8 – Media Descriptor
- 0xFFFFFFF9-0xFFFFFFFE – Not Defined
- 0xFFFFFFFF – End of File (EOF)

© 2010 The SANS Institute Author retains full rights.

Typically the cells may have zero which would indicate that there is no chain. In exFAT, if there is no chain, then a bit in the secondary flags of the directory record would indicate that the FAT chain is invalid. Knowing whether a chain exists in the first place is marked in the directory entry.

Since the cluster index begins at 2, there is no cluster 0 or cluster 1. So there should be no FAT entries with a value of 1. The actual valid values of clusters will be in the range of 2 to the Cluster Count + 1. For example, if there were 10 clusters, the range would be from 2 thru 11.

The largest value for Cluster Count + 1 is 0xFFFFFFF6. This limits the number of entries in the FAT table to 2^{32}-11. This is also the maximum number of clusters that can be tracked by a 32 bit FAT table in the Cluster Heap. 0xFFFFFFF7 is used to mark bad clusters and 0xFFFFFFFF is used for the FAT Chain EOF marker, and this is consistent with the prior versions of the FAT family. If a bad cluster is marked by a FAT bad cluster marker, then the media failure flag in the VBR should also be set.

The maximum number of FAT tables for exFAT is 2, but since Transactional exFAT (TexFAT) is not supported in version 1.0, there will only be one FAT defined. The address of the first FAT, the size of the FAT table, and the number of FAT tables is specified in the VBR. If the number of FAT is set to 2, then the 2nd FAT will begin on the next sector address following the first FAT and both FAT tables will be the same size. When there are 2 FAT tables, the active FAT table is indicated by a flag in the VBR. (See section 5.2)

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

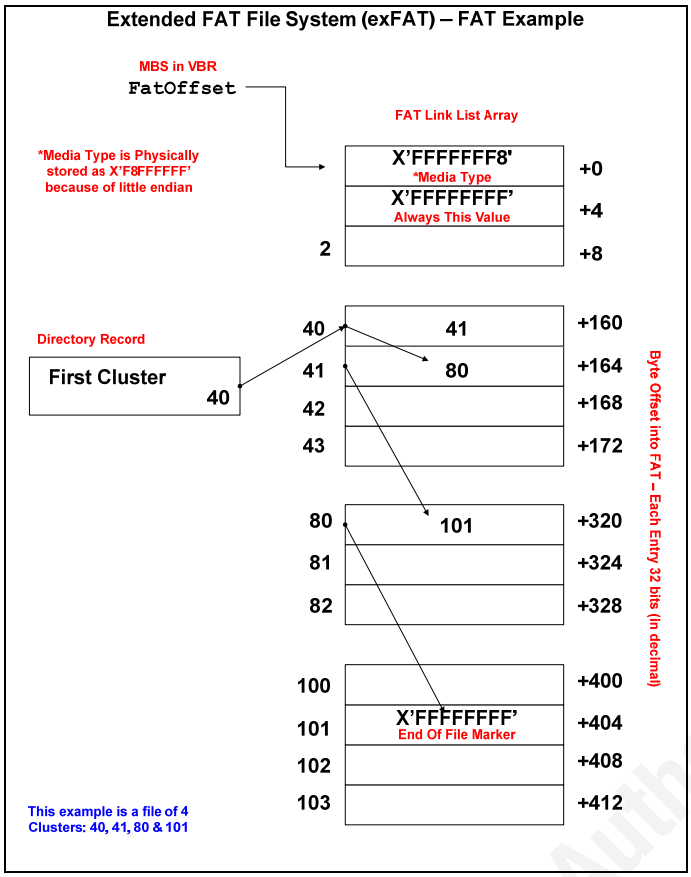


Figure 23 Extended FAT File System (exFAT) Example

Using Figure 23, examination of a FAT chain will be illustrated. There are two main pointers in play, the first being the sector address of the FAT table as specified in the Main Boot Sector of the VBR. This example assumes a single FAT. The FAT Offset points to the first entry in the first FAT table, which is the media descriptor.

The second pointer is the first cluster pointer located in the Stream Extensions Directory Entry in the directory. In this example the first cluster in the chain of data blocks is at cluster 40.

After processing the data in cluster 40, the next cluster needs to be accessed. The Stream Extension Directory Entry is examined and it was determined that the FAT chain is valid, so the contents in cell 40 of the FAT table is examined. Cell 40 is at byte offset 160 of the FAT table because each cell entry is 32 bits. The cell location contains 41, the next cluster in the chain.

© 2010 The SANS Institute Author retains full rights.

The process then reads the data in cluster 41, processes it, and extracts the contents in cell 41 of the FAT table. This time the contents is 80, so the process reads cluster 80, processes it, and reads cell 80 of the FAT table.

At cell 80 the contents is 101, the process reads cluster 101, processes it, and then examines cell 101 of the FAT table and sees the EOF marker, which flags the end of chain and no more clusters remain. In theory the FAT might not even need to be consulted for the EOF marker because the process may have already stopped reading if the data length was reached.

The walk of the FAT chain is now complete.

5.4 Allocation Bitmap Table

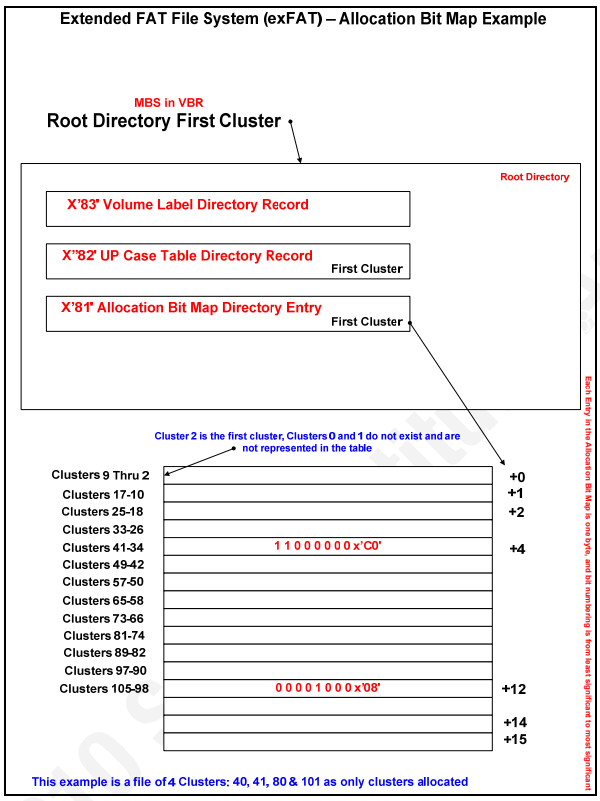


Figure 24 Extended FAT File System (exFAT) Allocation Bitmap Example

The maximum number of Allocation Bitmap tables for exFAT is 2, but since TexFAT is not supported, there will only be one Allocation Bitmap table.

The FAT table lives outside the Cluster Heap, but the Allocation Bitmap table lives inside the Cluster Heap and resides within a set of clusters. The Allocation Bitmap table appears to be built first and has been assigned to cluster 2 (the first cluster in the

© 2010 The SANS Institute Author retains full rights.

Cluster Heap). This is not written in stone, so the Allocation Bitmap table can actually be anywhere in the Cluster Heap, whether built there by the format program or moved somewhere else afterwards.

An Allocation Bitmap Directory Entry will point to the Allocation Bitmap table with its first cluster field. This entry is explained in section 6.3.

The Allocation Bitmap table is broken down into multiple 8 bit bytes where each 8 bit byte represents the allocation status of 8 clusters. The first cluster, cluster 2, is represented in the first byte by the first bit (bit 0) in the Allocation Bitmap table. When performing the calculation to find the relative position for the cluster in the table, first subtract 2 from the cluster number, divide by 8, and the resulting integer is the byte offset into the Allocation Bitmap table. For the remaining bits, the clusters will be mapped from the least significant bit (bit 0) through the most significant bit (bit 7).

Expanding on the example from Figure 23, Figure 24 will show a corresponding Allocation Bitmap where clusters 40 and 41 are allocated. The bit positions go from right to left, so in this example will show left to right as clusters 41 to 34 and the first 2 bits set to 1, indicating that they are allocated.

Bits that are zero are unallocated (free). Note that the first byte of the table represents clusters 9 through 2, where byte 0, bit 0, is cluster 2 – the first cluster.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

5.5 Time Stamp Format

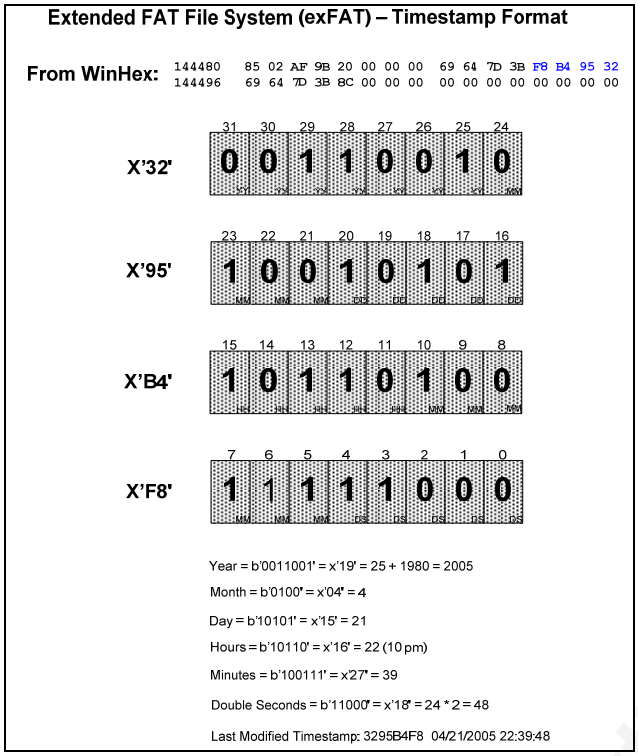


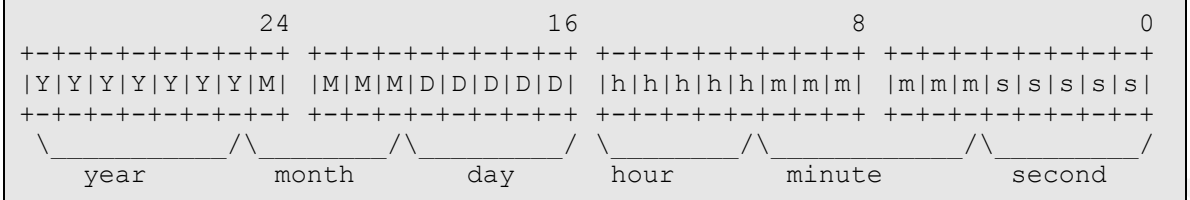
Figure 25 Extended FAT File System (exFAT) Timestamp Format

When an investigator or forensics analyst wants to develop a timeline of activity on a system, one of the most useful pieces of information is the file times. (Carvey, 2005). This process is called a MAC time analysis. Understanding the 3 main timestamps and their behavior when used for file creation, access, and modification is important to achieve this goal. The analysis of these timestamps can provide valuable insight into the history of the file and the extent of the user's knowledge of the files existence and contents (Casey, 2002). Section 6.8 will show where the timestamps are located and how to extract them. This section will explain the underlying format in order to help the forensics examiner understand how to convert the timestamp to a human readable date and time.

A breakdown of the file system timestamp format is provided by Carrier (Carrier, 2005). Another example is provided in Figure 25. MSDN provides a mapping and calls this the DOS date/time format (Figure 26).

© 2010 The SANS Institute Author retains full rights.

The DOS date/time format is a bitmask:



The year is stored as an offset from 1980. Seconds are stored in two-second increments. (So if the "second" value is 15, it actually represents 30 seconds.)

Source: http://blogs.msdn.com/oldnewthing/archive/2003/09/05/54806.aspx

Figure 26 The DOS Date/Time format

The DOS date/time format has not changed in exFAT, and is the same as used in earlier FAT file systems. The exFAT file system provides support for UTC timestamps which has an advantage when data is collected from different time zones, and can be important when the forensics examiner has to correlate data and logs taken from several different systems that may have been located in different time zones. This analysis hasn't established if there is a special single UTC timestamp format, but it was determined that the exFAT file system actually uses the aggregate of three different fields to make up what can be called the UTC timestamp. These fields are not stored together as one single set of fields for each type of date/time timestamp. This is different than the NTFS UTC timestamps that use a 64-bit number in 100 nanosecond intervals with an epoch of January 1, 1601 (UTC).

The first field is the DOS date/time value. This is actually 2 separate fields by itself since the date is in one 16 bit word and the time is in the other 16 bits. Since both of these components are stored together, this will be treated as one 32 bit (4 bytes) date/time stamp. Microsoft has also specified this as a 4 byte single field, and not as two separate fields.

The next field is a one byte field for 10ms units, and ranges from 0-199. This actually provides the "odd" seconds. Since in the DOS timestamp format the seconds are really "double seconds" the seconds will always be even. When the 10ms portion is then factored in, between 0-1990 ms, or between 0-1.99 seconds is being added. So when the contents of this field are 100 or more, the seconds will become odd when combined. This field only exists for the Create and Last Modified timestamps, and it appears that

© 2010 The SANS Institute Author retains full rights.

Windows XP maintains both these fields while Server 2008 only maintain the create 10ms field. This appears to be an inconsistency in the cross platform implementation, and requires more black box behavioral analysis to map the different scenarios.

The third field which appears for all three components (create, modify, access) is the time zone offset, and is one byte. These fields contain non-zero values when UTC support is present. Windows XP with exFAT support installed has UTC support, but Vista SP1 and Server 2008 SP1 did not. Examination of these fields will provide information on whether the operating system that created or updated the corresponding date/time timestamp field had UTC support, and will provide insight into the timestamp contents. Initial tests using Server 2008 SP1 did not produce these dates. After applying SP2 to the Server 2008 system the time zone offsets began to appear.

A search to find any documentation on the format of the time zone offsets did not produce any results. Trial and error was used by changing the clock settings on the Windows XP machine and then observing changes in these fields to see how they were affected. Table 23 was generated based on those observations and with some extrapolation provides a translation of these offsets. They appear to be in the range of 128-255 in 15 minute increments that appear to provide a range of ± 16 hours. A formula was developed (by Jeff Hamm) that shows the time zone offset to be a 7 bit signed integer. The purpose of the high order bit has not been determined.

The location of these fields also conflict with the layout as appearing in the specification released in the Patent. The create time zone offset overlays the field defined for last access 10ms, and the other two time zone offset values overlay 2 bytes of a reserved area. The File Directory Entry layout in Table 15 has been modified to reflect what was observed based on the implementations behavior and does not match the layout provided in the specification.

The DOS Timestamp will always be written with the local machine time. It could have been implemented in one of two ways, where the UTC time could have been entered into the DOS Timestamp, but Microsoft apparently didn't go that route. So regardless of whether the host system has UTC support or not, the same date/time information is recorded, it is the local time. Then, with UTC support, the time zone offset is recorded for the corresponding timestamp.

© 2010 The SANS Institute Author retains full rights.

5.6 Cluster Heap

The Cluster Heap is the data area of the file system volume. The Root Directory, files, subdirectories, the UP-Case Table, and the Allocation Bitmap reside in this area on the storage media. Allocation status – allocated or unallocated clusters – is tracked by the Allocation Bitmap, and when clusters must be chained to combine multiple clusters into a larger non-contiguous file, these chains are tracked within the FAT file structure. The FAT itself it stored outside of the Cluster Heap.

5.7 Transactional FAT

Transactional FAT or Transaction safe FAT, and also known as TexFAT for the exFAT file system is not supported in this version, and there is little documentation currently available on this feature. This is a limitation for this research as there is no empirical study that can be performed at this time. Information provided here is based on the theory of what to expect and looking at what Microsoft may have done in the Windows CE version of exFAT.

Microsoft has a patent (Microsoft Patent 7613738 (November 3, 2009)) called "FAT Directory Structure for use in Transaction Safe File System" that provides some idea of how this feature should work.

In Figure 42 is an example a 18 MB file using 140 clusters, each sized at 128KiB. This is 256 sectors per cluster, or 35,840 sectors that have to be written. Supposed the file system was fragmented in such a way to force the file to be fragmented, then 140 FAT entries would have to be accessed. In any case, 140 bits in the allocation table would have to also be modified when the file is written or deleted. Now, lets take a theoretical example of a 8.5GB avi video file using the same blocking factor to be written to a very large storage medium. That would come out to 64,850 clusters or over 16.6 million sectors. Even with very fast devices, it is not just the time to write out such a large file, but the multiple different operations required to complete such an operation. Updates are required to the directory, Allocation Bitmap, and FAT table. What happens if it breaks in the middle? The objective of Transaction Safe FAT is to make all those updates atomic.

The desirable properties of transactions is the ACID test. (Elmasri & Navathe, 1994) These properties include the Atomicity, Consistency, Isolation and Durability of that transaction. The atomic principle is effectively an "all or nothing" result.

© 2010 The SANS Institute Author retains full rights.

The way Transaction Safe FAT is supposed to work is that there will be 2 copies of the FAT and 2 copies of the Allocation Bitmap, and the FAT and Allocation Bitmap will be paired. The metadata would be frozen from updates so that updates to the file system would be isolated, and a working pair of the FAT/Bitmap would be updated and in flux. Should the transaction fail, such as the storage media suddenly being abruptly yanked out of the storage device or the occurrence of a power failure, the state and consistency of the file system would not be impacted. Once the transaction is successfully completed, the FAT/Bitmap pair is flipped to the other set.

The patent refers to the use of placeholders. Suppose a subdirectory is being updated. The subdirectory is pointed to by a parent directory, and that pointer is a first cluster field in the Stream Extensions Directory Entry of the File Entry Set that defines the subdirectory. A placeholder cluster is obtained, the data is copied over to the placeholder, and then the placeholder is updated. Once the update is complete, the cluster address pointer in the parent Stream Extension Directory Entry is then updated to point to the placeholder, and the old cluster can then be released and returned to the Cluster Heap.

Microsoft MSDN CC907928 discusses limitations in the Windows CE version of TexFAT. It is not known at this time whether these limitations will carry over to the desktop as well. In the Windows CE version, file names are limited to 247 characters due to the way the Root Directory will be updated. A shadow copy of the Root Directory is maintained for the updates that require transaction safety. The Root Directory in Windows CE is in a fixed location and can't be moved.

6 exFAT Directory Structure

To explain what is not in exFAT, let's examine something that the legacy FAT file systems did have. Figure 27 shows a display of a FAT32 subdirectory:

```
Offset       0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  1048576   2E 20 20 20 20 20 20 20  20 20 20 10 00 AC 8B 6B   .           ..¬‹k
  1048592   87 3B 87 3B 00 00 8C 6B  87 3B 04 00 00 00 00 00   ‡;‡;..Œk‡;......
  1048608   2E 2E 20 20 20 20 20 20  20 20 20 10 00 AC 8B 6B   ..          ..¬‹k
  1048624   87 3B 87 3B 00 00 8C 6B  87 3B 00 00 00 00 00 00   ‡;‡;..Œk‡;......
```

Figure 27 Winhex display of FAT 32 Subdirectory for special pointers

© 2010 The SANS Institute Author retains full rights.

Since Winhex supports FAT and FAT32 with its factory supplied templates, let's run the FAT directory template on these two special directory entries.

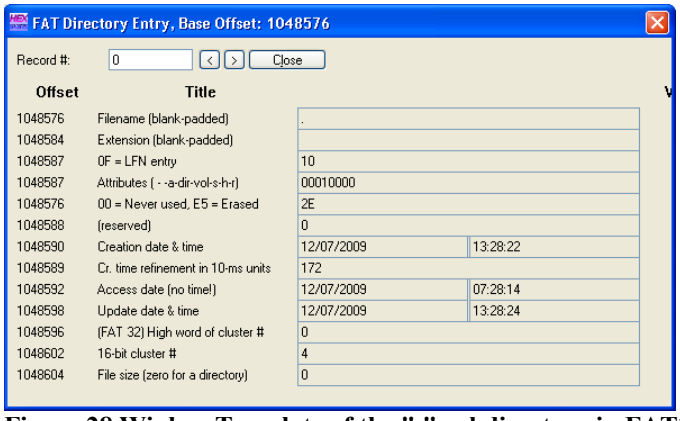


Figure 28 Winhex Template of the "." subdirectory in FAT32

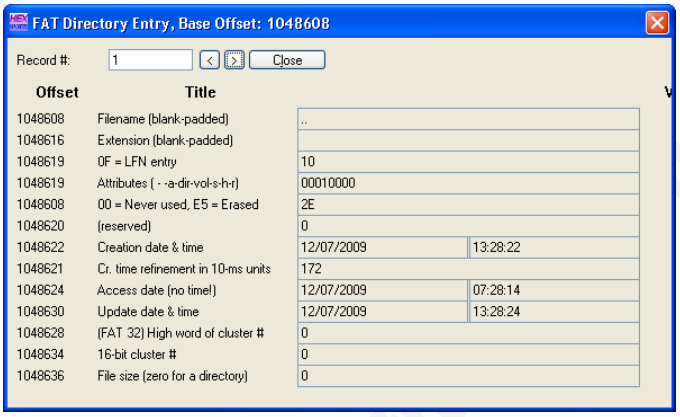


Figure 29 Winhex Template of the ".." subdirectory in FAT32

Figure 28 and Figure 29 displays two special subdirectory entries that exist in the FAT and FAT32 subdirectories. These are actually physical entries in those directories and are the first two entries of each subdirectory, and exist even when the directory is empty. They do not exist in the Root Directory. Those entries have special definitions: "." means this directory and ".." means containing directory. The exFAT specifications indicate that these special filenames shall not be physically recorded in the directory. Now when the directory is listed they may be listed as if they did exist, so in exFAT these two special directories have shifted from a physical concept to a conceptual concept – the reverse of the legacy FAT file systems.

Microsoft classifies the entries in the directory as either critical or benign. Critical entries are required for the file system to operate properly, benign entries are optional.

© 2010 The SANS Institute Author retains full rights.

With the exception of file and subdirectory definitions, all critical entries must be in the Root Directory.

For the forensics examiner, the benign entries may have significant importance. When an exFAT file system goes through the mount process it is expected that the mount will only succeed if the critical entries are in order. All defined critical entries must be known to the file system and if any unknown critical entries are found then the file system should not be mounted. However, the file system will ignore the benign entries.

This is where it can become interesting. This would allow critical entries to be changed to benign entries, or even new benign entries to be created and effectively create a new file system within a file system. If someone wanted to create files and hide them, benign entries could be created pointing to the clusters where the hidden data resides and the file system would ignore those entries. If the Allocation Bitmap was updated to prevent destruction of the data, there would be allocated space to files that did not show up in any directory listing. The only way to uncover this type of hidden data is to go deep into the file system with a byte by byte inspection.

The directory entries are also broken down into Primary and Secondary entries. In exFAT the only secondary entries are found in a file or subdirectory definition. The Primary and Secondary entries of a specific definition, when grouped together, are called a Directory Entry Set, which is actually an array of directory entries.

Type Field	Offset	Size
In Use	7	1
Category	6	1
Importance	5	1
Code	0	5

Table 8 Breakdown of the Entry Type

Table 8 provides the layout of the entry type, a one byte field that identifies each entry in the directory. Since 0x00 is an end of directory marker, 0x80 is not defined. The identification of primary/secondary and critical/benign entries come from this value.

- In Use: 0 – Not n Use, 1- In Use
- Category: 0 – Primary entry, 1 – Secondary entry
- Importance: 0 – Critical entry, 1 – Benign entry
- Code: Identifies the entry

© 2010 The SANS Institute Author retains full rights.

6.1 Root Directory

The Root Directory is used to define files, sub-directories, the volume label, the location of the UP-Case Table, and the location of the Allocation Bitmap. Other entries such as TexFAT and ACL may also reside in the directory, but these entries do not exist because support has not been implemented. They do exist in the Windows CE version of the exFAT support.

The directory entries are 32 bytes in length, each beginning with a type code (Entry Type) to identify the purpose and status of the entry. Multiple entries are used to define a file or a subdirectory, with a minimum of 3 and a maximum of 19 entries. The directory entry set making up a file is an ordered array of entries, containing 1 primary and multiple secondary entries, and do not contain sequence numbers or other identifiers to keep the entries in order.

For File Entry Sets, the first bit is used to indicate if the entry is in use. In some cases, if this bit is set to off, then the entry is part of a deleted set. This will vary based on the purpose of the entry.

10 different entry types are defined in exFAT, and in the next sections the details will be provided on each, where known. A subdirectory can hold up to 2,796,202 ($2^{23}/3$) files. This is based on a maximum data length of a subdirectory being limited to 256MiB. Such a limitation wasn't indicated for the Root Directory.

6.2 Volume Label Directory Entry

<table>
<tr><th>Field Name</th><th>Offset (byte)</th><th>Size (byte)</th><th>Description/Value</th></tr>
<tr><td>Entry Type</td><td>0</td><td>1</td><td>0x83
<table>
<tr><td>Type Field</td><td>Offset</td><td>Size</td><td>Value</td></tr>
<tr><td>In Use</td><td>7</td><td>1</td><td>1</td></tr>
<tr><td>Category</td><td>6</td><td>1</td><td>0</td></tr>
<tr><td>Importance</td><td>5</td><td>1</td><td>0</td></tr>
<tr><td>Code</td><td>0</td><td>5</td><td>00011</td></tr>
</table></td></tr>
<tr><td>Character Count</td><td>1</td><td>1</td><td>Number of characters in label</td></tr>
<tr><td>Volume Label</td><td>2</td><td>22</td><td>Volume Label in Unicode</td></tr>
<tr><td>Reserved</td><td>24</td><td>8</td><td></td></tr>
<tr><td colspan="4">Comments: If the Entry Type is 0x03 then there is no volume label</td></tr>
</table>

Table 9 Layout for Volume Label Directory Entry

© 2010 The SANS Institute Author retains full rights.

The Volume Label Directory Entry defines the volume label. This is a 0x83 entry, and the length of the volume label is a maximum of 11 characters in length and is expressed as a 16 bit Unicode string. A character count is provided to indicate the length of the volume label as the string is not null terminated.

If the media is formatted without a volume label, then a 0x03 directory entry will appear instead, which indicates that there is no volume label. Here the "InUse" bit would be set to off, but it does not indicate a deleted volume label, just that one was not assigned.

6.3 Allocation Bitmap Directory Entry

<table>
<tr><th>Field Name</th><th>Offset (byte)</th><th>Size (byte)</th><th>Description/Value</th></tr>
<tr><td>Entry Type</td><td>0</td><td>1</td><td>0x81
<table><tr><td>Type Field</td><td>Offset</td><td>Size</td><td>Value</td></tr><tr><td>In Use</td><td>7</td><td>1</td><td>1</td></tr><tr><td>Category</td><td>6</td><td>1</td><td>0</td></tr><tr><td>Importance</td><td>5</td><td>1</td><td>0</td></tr><tr><td>Code</td><td>0</td><td>5</td><td>00001</td></tr></table></td></tr>
<tr><td>Bit Map Flags</td><td>1</td><td>1</td><td><table><tr><td>Bit</td><td>Size</td><td>Value</td><td>Purpose</td></tr><tr><td>7-1</td><td></td><td></td><td>Reserved</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1st Bitmap</td></tr><tr><td>0</td><td>1</td><td>1</td><td>2nd Bitmap</td></tr></table></td></tr>
<tr><td>Reserved</td><td>2</td><td>18</td><td></td></tr>
<tr><td>First Cluster</td><td>20</td><td>4</td><td>Cluster Address of First Data Block</td></tr>
<tr><td>Data Length</td><td>24</td><td>8</td><td>Length of the Data</td></tr>
<tr><td colspan="4">Comments: There will be at least 1 of these entries. The number of entries is based on the Number of Fats specified in the VBR/MBS.</td></tr>
</table>

Table 10 Layout for Allocation Bitmap Directory Entry

The Allocation Bitmap Directory Entry defines the location of the Allocation Bitmaps. There will be either 1 or 2 of this type 0x81 entry, depending on the number of FATs. 2 FATs only exist when TexFAT is being used. When there are 2 FATs, a bit in the Bit Map Flags will indicate which FAT is associated with this directory entry.

The first cluster field points to the start of the Allocation Bitmap. Although the Allocation Bitmap may usually be in the first cluster (cluster 2) it can theoretically be placed anywhere in the Cluster Heap, and probably can be moved to a different set of clusters. The forensics examiner should not depend on the Allocation Bitmap being the

© 2010 The SANS Institute Author retains full rights.

first cluster in the Cluster Heap, as someone who may manipulate the Cluster Heap storage could move the Allocation Bitmap to fool an investigator.

The data length field holds the length of the Allocation Bitmap in bytes. To determine what this value should be, take the number of clusters in the Cluster Heap (Cluster Count in the VBR), and divide by 8 – rounding up to the next byte integer. This will calculate how many bytes are required for each Allocation Bitmap; the Allocation Bitmap is one bit per cluster defined in the Cluster Heap.

6.4 UP-Case Table Directory Entry

<table>
<tr><th>Field Name</th><th>Offset (byte)</th><th>Size (byte)</th><th>Description/Value</th></tr>
<tr><td>Entry Type</td><td>0</td><td>1</td><td>0x82
<table><tr><td>Type Field</td><td>Offset</td><td>Size</td><td>Value</td></tr><tr><td>In Use</td><td>7</td><td>1</td><td>1</td></tr><tr><td>Category</td><td>6</td><td>1</td><td>0</td></tr><tr><td>Importance</td><td>5</td><td>1</td><td>0</td></tr><tr><td>Code</td><td>0</td><td>5</td><td>00010</td></tr></table></td></tr>
<tr><td>Reserved1</td><td>1</td><td>3</td><td></td></tr>
<tr><td>Table Checksum</td><td>4</td><td>4</td><td></td></tr>
<tr><td>Reserved2</td><td>8</td><td>12</td><td></td></tr>
<tr><td>First Cluster</td><td>20</td><td>4</td><td>Cluster Address of First Data Block</td></tr>
<tr><td>Data Length</td><td>24</td><td>8</td><td>Length of the Data</td></tr>
<tr><td colspan="4">Comments:</td></tr>
</table>

Table 11 Layout for UP-Case Table Directory Entry

The UP-Case Table is used to convert the filename to upper case for certain operations, such as comparing the filename to a search string. The case of the filenames is preserved when stored in the directory, but certain operations are case insensitive.

The first cluster field points to the beginning of the UP-Case Table, and the data length holds the length of the table. There is a Table Checksum value which is a checksum of the table and must be checked prior to using the table. A routine that can be used calculate the checksum is shown in Figure 30.

© 2010 The SANS Institute Author retains full rights.

```
UINT32 UPCaseChecksum(const unsigned char octets[], long NumberOfBytes)
{
  UINT32 Checksum = 0;
  long Index;
  for (Index = 0; Index < NumberOfBytes; Index++)
       {
              Checksum = ((Checksum <<31) | (Checksum>> 1)) + (UINT32) octets[Index];
       }
  return Checksum;
```

Figure 30 Checksum routine for the UP-Case Table

The UP-Case Table is small, at less than 6,000 characters. Now imagine a cluster size of 128KiB being used for the file system. That provides over 100KiB of file slack space for hiding things. Now imagine the maximum cluster size of 32MiB. This is a cluster location that does not display when executing a DIR command, will probably never be moved or relocated during a disk defragmentation, and probably will not be modified or overwritten by the file system. The checksum is only done against the UP-Case Table itself and not on the slack space and makes this a prime target space for the sophisticated criminal to hide things.

6.5 Volume GUID Directory Entry

Field Name	Offset (byte)	Size (byte)	Description/Value
Entry Type	0	1	0xA0 Type Field: In Use — Offset 7, Size 1, Value 1 Type Field: Category — Offset 6, Size 1, Value 0 Type Field: Importance — Offset 5, Size 1, Value 1 Type Field: Code — Offset 0, Size 5, Value 0
Secondary Count	1	1	Always Zero
Set Checksum	1	2	
General Primary Flags	4	2	Field: Allocation Possible — Offset 0, Size 1, Value 0 – No Field: No FAT Chain — Offset 1, Size 1, Value 0 – Valid 1 - Invalid Field: Custom — Offset 2, Size 14
Volume GUID	6	16	
Reserved	22	10	
Comments: There is either no GUID entry, or a maximum of 1 This is a benign primary entry			

Table 12 Layout for Volume GUID Directory Entry

This entry is defined as a benign primary entry. The specification provided a definition of a Volume GUID Directory Entry. There may be either 1 or none of these

© 2010 The SANS Institute Author retains full rights.

occurring in the file system. None of the tests performed during this research produced such an entry, and with lack of a GUID entry to analyze, there isn't much more that can be provided at this time.

6.6 TexFAT Padding Directory Entry

Field Name	Offset (byte)	Size (byte)	Description/Value
Entry Type	0	1	0xA1
Reserved	1	31	
Comments: The patent does not provide the specifications for this entry This is a benign primary entry			

Type Field	Offset	Size	Value
In Use	7	1	1
Category	6	1	0
Importance	5	1	1
Code	0	5	1

Table 13 Layout for TexFAT Padding Directory Entry

TexFAT is not supported in version 1.00, but the specification indicated the existence of such an entry without actually defining the fields. This is also classified as a benign primary entry.

6.7 Windows CE Access Control Table Directory Entry

Field Name	Offset (byte)	Size (byte)	Description/Value
Entry Type	0	1	0xE2
Reserved	1	31	
Comments: The patent does not provide the specifications for this entry This is a benign secondary entry			

Type Field	Offset	Size	Value
In Use	7	1	1
Category	6	1	1
Importance	5	1	1
Code	0	5	2

Table 14 Layout for Windows CE Access Control Table Directory Entry

ACL is not supported in version 1.00, but the specification indicated the existence of such an entry without actually defining the fields. Since the entry is labeled "Windows CE", this may be a holdover of the Windows CE implementation. This is also classified as a benign secondary entry. Access control is typically on a file by file basis. If these

© 2010 The SANS Institute Author retains full rights.

entries do come into support in the future, they probably will become secondary entries of the File Entry Set.

6.8 File Directory Entry

Field Name	Offset (byte)	Size (byte)	Description/Value
Entry Type	0	1	0x85
Secondary Count	1	1	
Set Checksum	2	2	
File Attributes	4	2	
Reserved1	6	2	
Create	8	4	DOS Timestamp Format
Last Modified	12	4	DOS Timestamp Format
Last Accessed	16	4	DOS Timestamp Format
Create 10ms	20	1	10ms increments between 0-199
Last Modified 10ms	21	1	10ms increments between 0-199
Create TZ Offset[1]	22	1	Time zone difference to UTC in 15 min increments
Last Modified TZ[1] Offset	23	1	Time zone difference to UTC in 15 min increments
Last Accessed TZ[1] Offset	24	1	Time zone difference to UTC in 15 min increments
Reserved2	25	7	

Type Field	Offset	Size	Value
In Use	7	1	1
Category	6	1	0
Importance	5	1	0
Code	0	5	00101

Attribute	Offset	Size	Mask
Reserved2	6	10	
Archive	5	1	0x20
Directory	4	1	0x10
Reserved1	3	1	
System	2	1	0x04
Hidden	1	1	0x02
Read-Only	0	1	0x01

Comments: **If the In Use bit is zero (0x05) then this is probably a deleted file, it will also occur when a file is renamed and the number of file name extension directory entries changes.**
[1]These fields are not defined in the specification provided in the patent document, they were observed during the analysis of the implementation.

Table 15 Layout for File Directory Entry

A File Directory Entry defines a file or subdirectory. It does not stand alone. The File Entry Set should contain from 3 to 19 32-byte directory entries. The File Directory Entry (0x85) starts the File Entry Set, followed by a Stream Extension Directory Entry

© 2010 The SANS Institute Author retains full rights.

(0xC0) and then from 1 to 17 of the File Name Extension Directory Entry (0xC1). These entries must be in sequence without gaps.

The secondary count is a one byte unsigned integer that will range from 2 to 18, and indicate how many entries follow the primary directory entry. In the case of the 0x85 entry, this count does not include the 0x85 entry itself. This value can actually go to 255 as a maximum, but in the current implementation, 18 will be the maximum. The reason that the 0x85 entry is not included is that this entry is a primary entry and the other entries are secondary entries, and this is a count of the secondary entries contained in the entry set.

File Attributes are very similar to those used in FAT/FAT32 and should have the same value definitions. The attribute for a volume label does not exist in exFAT because the volume label is defined in a 0x81 entry. Since Long File Name and 8.3 filename support does not exist in exFAT, those attributes are no longer defined as well.

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  143456   85 02 32 50 20 00 00 00  66 64 7D 3B 73 85 32 35   ….2P ...fd};s…25
  143472   66 64 7D 3B C3 00 00 00  00 00 00 00 00 00 00 00   fd};Ã...........
  143488   C0 03 00 0B 9B 10 00 00  C0 E8 03 00 00 00 00 00   À...›...Àè......
  143504   00 00 00 00 06 00 00 00  C0 E8 03 00 00 00 00 00   ........Àè......
  143520   C1 00 77 00 69 00 6E 00  68 00 65 00 6C 00 70 00   Á.w.i.n.h.e.l.p.
  143536   2E 00 65 00 78 00 65 00  00 00 00 00 00 00 00 00   ..e.x.e.........
```

Figure 31 File Entry Set created by Server 2008 SP1

Figure 31 shows a File Entry Set sequence of the x85, xC0 and xC1 entries. The three timestamps are: Create (0x3B7D6466) Last Modified (0x35328573) Last Accessed (0x3B7D6466) and this entry was created using a Windows Server 2008 SP1 machine. Now examine what happens when you display the properties of this fie:

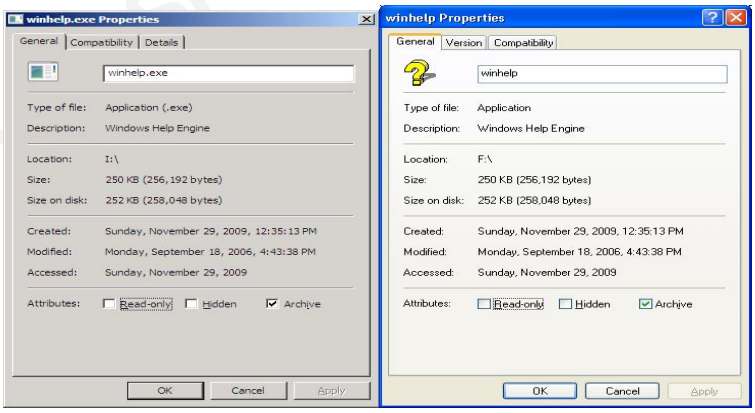


Figure 32 Display of File Properties for exFAT created on Server 2008

Robert Shullich rshullic@earthlink.net

© 2010 The SANS Institute Author retains full rights.

For accessed, the date is provided but no time is given. The left was performed using Windows Server 2008 SP1, and the right was performed on a Windows XP SP3 machine, the results are both the same. Although with exFAT the last access time is stored, not all commands will display the time value. Figure 33 shows the properties of a NTFS file and displays both the time and date, so the capability to display the accessed time is there but not provided for exFAT files.

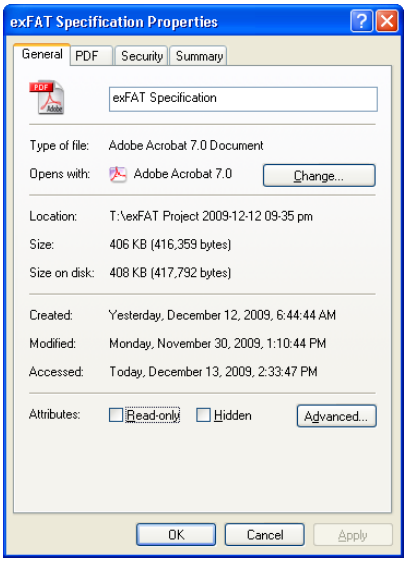


Figure 33 Display of File Properties for NTFS on Windows XP

The exFAT specification defines 3 additional fields of a single byte non-signed integer to add a 10ms increment to each time, this value ranges (in decimal) from 0 to 199. Only 2 of these 3 fields were implemented. Since the seconds recorded in the DOS time value is double seconds (every two seconds) the 10ms increment adds in the odd missing second as well as refines the time to within 10 milliseconds.

Now look at the display in Figure 34 at the same file using Windows Explorer:

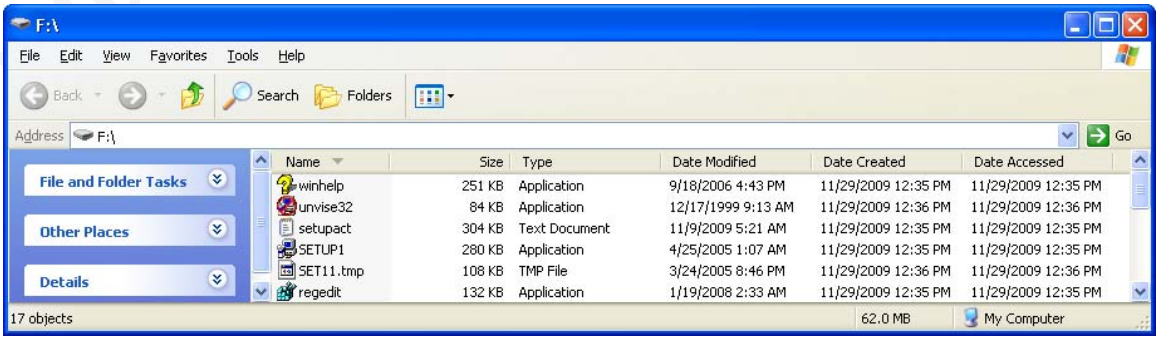


Figure 34 Display of dates using Windows Explorer

© 2010 The SANS Institute Author retains full rights.

The date accessed shows a time value when the properties window did not.

In performing some of the black box testing on the Windows XP system, the MAC times exhibited unexpected behavior. Opening a file on an exFAT file and displaying data in the file, or copying the file to another physical disk did not affect the last accessed timestamp on the source file. Performing the same operations on a NTFS file did change the last accessed timestamp on each operation. It is expected that when a file is accessed, i.e. one of its data clusters are read, that the last accessed timestamp would be changed, and observation showed it was not updated at all. If the file was modified, then the last accessed timestamp did change. Also, in this scenario, the modified 10ms increment was set to zero. Although the analysis was only performed for a few files, this indicates a potentially bigger problem, especially if an investigation depends on these timestamps being consistent and properly updated as expected.

Let's look at a theoretical situation: suppose the forensics examiner is investigating a case of child porn, and the investigation target makes a claim that the pictures (let's say here that they are JPEG files in a JPG format) got downloaded but the target claims that they never looked at them and didn't even realize that they were there. The forensics examiner is dependent on the different metadata of the file system to confirm or refute such a claim. However, if the file system does not update the last accessed timestamp when any program opens the JPG file, then how can such a claim be validated?

In Figure 31 look at the next 3 bytes (offset 143476) after the time stamps, they are 0xC3, 0x00, 0x00 and refer to the 10ms additions to Create, Modified, Accessed, in that order. The 10ms increment for Modified and Accessed are zero.

```
Offset       0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  7929856   85 04 6C 53 20 00 00 00  55 0B 76 3B 39 65 27 33   ….lS ...U.v;9e'3
  7929872   55 0B 76 3B 45 00 EC EC  EC 00 00 00 00 00 00 00   U.v;E.ìì.......
```

Figure 35 Winhex of 0x85 entry created on Windows XP SP3

Figure 35 shows the Winhex dump of a 0x85 entry for a different USB stick and for a different file, but was created on Windows XP SP3 with the KB for exFAT support applied. Observe two occurrences in the display: First, the 10ms increments for Create, Modified and Access are 0x45, 0x00 and 0xEC. Second, the other two bytes of 0xEC are

© 2010 The SANS Institute Author retains full rights.

sitting in a reserved area which is undefined. For all files being created in exFAT by the Windows XP system, the 3 byte sequence of 0xECECEC is always being written in offset 22 of the 0x85 entry. This value may be different when created on a different XP system. As explained in section 5.5, these are time zone offsets and are part of the UTC timestamp. (In this example 0xEC will convert to EST or GMT-5, the time zone in use on the systems used in this research).

One of the key procedures of a forensics examiner is to create a timeline of the events that occurred within a system. In order to do this, there needs to be an understanding of the timestamp metadata stored in the file system as well as knowledge of the tools that display such data (MAC Time Analysis). The behavior of what is written in the timestamp storage locations has some variance depending on which operating system is used. Even on the same operating system the different tools that display that metadata may behave differently.

A checksum is computed across all entries in the entry set. The size of the entry set in entries will be the secondary count + 1; the size of the entry set in bytes will be the size * 32, with a minimum entry set of 3 entries being 96 bytes.

```
UINT16 EntrySetChecksum(const unsigned char octets[], long NumberOfBytes)
{
  UINT16 Checksum = 0;
  long Index;

  for (Index = 0; Index < NumberOfBytes; Index++)
        {
                if (Index == 2 || Index == 3)
                {
                        continue;
                }
                Checksum = ((Checksum <<15) | (Checksum>> 1)) + (UINT16) octets[Index];
        }
  return Checksum;
}
```

Figure 36 File Entry Set Checksum Calculation in C

Figure 36 shows the code that can be used to calculate the checksum value for the File Entry Set. The checksum itself is located at offsets 2 & 3, so this field is excluded during the calculation.

If the 0x85 entry is not in use, with the high bit set off in the entry type, then this directory entry will actually be shown as a 0x05 entry. This most likely indicates that the file was deleted. All the remaining entries that are part of the same File Entry Set will also have their high order bits set to off, resulting in the 0xC0 becoming a 0x40 and each

© 2010 The SANS Institute Author retains full rights.

0xC1 becoming a 0x41. Although these bits are changed to zero, the checksum of the File Entry Set is not recalculated or altered.

The next four figures will be used to take a deeper dive into the File Directory Entry:

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  524544   85 04 EF 91 20 00 00 00  50 62 86 3B D3 62 BA 3A   ….ï‘ ...Pb†;Ób°:
  524560   50 62 86 3B 11 00 EC EC  EC 00 00 00 00 00 00 00   Pb†;..ìì........
```

Figure 37 Winhex Dump of a 0x85 image before deletion

```
Seeking Relative Byte Location: 524544, For Directory Index: 9, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 85 Directory Entry Record
Checksum:                91EF
Calculated Checksum is:  91EF Size Directory Set (bytes):  160
Secondary Count          004
File Attributes:         0020 Archive
Create Timestamp:        3B866250  12/06/2009 12:18:32
Last Modified Timestamp: 3ABA62D3  05/26/2009 12:22:38
Last Accessed Timestamp: 3B866250  12/06/2009 12:18:32
 10 ms Offset Create           11  17
 10 ms Offset Modified         00   0
 Time Zone Create              EC 236 Value of tz is: GMT -05:00
 Time Zone Modified            EC 236 Value of tz is: GMT -05:00
 Time Zone Last Accessed       EC 236 Value of tz is: GMT -05:00
```

Figure 38 Formatted translation of a 0x85 image before deletion

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  524544   05 04 EF 91 20 00 00 00  50 62 86 3B D3 62 BA 3A   ..ï‘ ...Pb†;Ób°:
  524560   50 62 86 3B 11 00 EC EC  EC 00 00 00 00 00 00 00   Pb†;..ìì........
```

Figure 39 Winhex dump of a 0x85 image after deletion

```
Seeking Relative Byte Location: 524544, For Directory Index: 9, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 5 Directory Entry Record (Deleted)
Checksum:                91EF
Calculated Checksum is:  89EF Size Directory Set (bytes):  160
Secondary Count          004
File Attributes:         0020 Archive
Create Timestamp:        3B866250  12/06/2009 12:18:32
Last Modified Timestamp: 3ABA62D3  05/26/2009 12:22:38
Last Accessed Timestamp: 3B866250  12/06/2009 12:18:32
 10 ms Offset Create           11  17
 10 ms Offset Modified         00   0
 Time Zone Create              EC 236 Value of tz is: GMT -05:00
 Time Zone Modified            EC 236 Value of tz is: GMT -05:00
 Time Zone Last Accessed       EC 236 Value of tz is: GMT -05:00
```

Figure 40 Formatted translation of a 0x85 image after deletion

This example was performed using a Windows XP system, so the 3 time zone offsets (0xEC) appear in the entry. Now this value may be different on other Windows XP systems, for example Jeff Hamm of Paradigm Solutions who presented this topic at the Techno Digital Investigation conference in Oct 2009, showed the values of 0xF0 in

© 2010 The SANS Institute Author retains full rights.

his hex dumps (Hamm 2009), and he believes that these may be the time zone values. He believed that these values were a time zone offset in minutes. Further research using the black box analysis shows that these are in fact a time zone offset, but not in minutes, but coded for 15 minute intervals. Those values that appeared in a reserved area might have been ignored during my analysis if Jeff hadn't given me a clue of what they might be.

Before deleting the file, the file directory entry has an entry type of 0x85, and a file entry set checksum value of 0x91EF and the program recalculates the checksum and it matches with 0x91EF.

The file is then deleted, and an after set of images of the directory entries are produced. The file entry type is now set to 0x05 because the entry is no longer in use and the high order bit has been set to zero. The checksum stored in the file directory entry remains at 0x91EF, but when the File Entry Set checksum was recalculated, a value of 0x89EF was obtained. These checksums do not match, and this is because the "InUse" bit was turned off in all the entry types of the File Entry Set. But the checksum was not recalculated and updated as part of the deletion process.

When a File Entry Set goes to not "InUse" it doesn't always mean the file was deleted. When a file is renamed, and the file name length requires a different number of file name extension entries, a complete new File Entry Set is created. It is possible to determine some of the file renaming actions performed on the file system because the directory entry with the old name may still be in the directory and intact. It also appears that the file system may be more apt at times to add a new directory entry into unused space in the directory before overwriting existing inactive entries providing a potential longer life for the artifacts.

A test was performed where a file was renamed but did not require a change in the size of the File Entry Set. A filename of less than 15 characters was renamed to a smaller file name. The original File Entry Set was modified, and did not result in a deleted File Entry Set followed by a new File Entry Set.

The size of the File Entry Set in Figure 38 is 5 entries, and since the size of each directory entry is 32 bytes, the complete File Entry Set is 160 bytes. The File Entry Set contains 5 entries because the secondary count is 4, and this does not include the file directory entry itself, so one is added to it.

© 2010 The SANS Institute Author retains full rights.

6.9 Stream Extension Directory Entry

<table>
<tr><th>Field Name</th><th>Offset (byte)</th><th>Size (byte)</th><th>Description/Value</th></tr>
<tr><td>Entry Type</td><td>0</td><td>1</td><td>0xC0
<table><tr><td>Type Field</td><td>Offset</td><td>Size</td><td>Value</td></tr><tr><td>In Use</td><td>7</td><td>1</td><td>1</td></tr><tr><td>Category</td><td>6</td><td>1</td><td>1</td></tr><tr><td>Importance</td><td>5</td><td>1</td><td>0</td></tr><tr><td>Code</td><td>0</td><td>5</td><td>0</td></tr></table></td></tr>
<tr><td>General Secondary Flags</td><td>1</td><td>1</td><td><table><tr><td>Field</td><td>Offset</td><td>Size</td><td>Value</td></tr><tr><td>Allocation Possible</td><td>0</td><td>1</td><td>0 – No
1 – Yes</td></tr><tr><td>No FAT Chain</td><td>1</td><td>1</td><td>0 – Valid
1 - Invalid</td></tr><tr><td>Custom</td><td>2</td><td>14</td><td></td></tr></table></td></tr>
<tr><td>Reserved1</td><td>2</td><td>1</td><td></td></tr>
<tr><td>Name Length</td><td>3</td><td>1</td><td></td></tr>
<tr><td>Name hash</td><td>4</td><td>2</td><td>Used for directory searches</td></tr>
<tr><td>Reserved2</td><td>6</td><td>2</td><td></td></tr>
<tr><td>Valid Data Length</td><td>8</td><td>8</td><td></td></tr>
<tr><td>Reserved3</td><td>16</td><td>4</td><td></td></tr>
<tr><td>First Cluster</td><td>20</td><td>4</td><td>Cluster Address of First Data Block</td></tr>
<tr><td>Data Length</td><td>24</td><td>8</td><td>Length of the Data
If this is a directory, then the maximum value for this field is 256M</td></tr>
<tr><td colspan="4">Comments: 1 Entry Per File
If the In Use bit is zero (0x40) then this is probably part of a deleted file set</td></tr>
</table>

Table 16 Layout for Stream Extension Directory Entry

The Stream Extension Directory Entry provides information on the location and size of the file. It also provides a hash of the file name that can be used to speed up directory searches. The address of the first cluster points to the first cluster of the data file.

If the data length is zero, then there might not be any cluster allocated, and the first cluster address may also be zero. Since the first cluster of the Cluster Heap always begins at index 2 (cluster 0 and cluster 1 are not defined), a zero as the first cluster can never be the address of a real cluster. If there is no cluster allocated to this file, then the secondary flags should indicate that the FAT chain is also invalid.

If the file is deleted then the first bit that indicates that the entry would be in use will actually now be set to zero, and the resulting entry type will be 0x40.

The next figures will take a deep dive at the Stream Extensions Directory Entry:

© 2010 The SANS Institute Author retains full rights.

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  524576   C0 03 00 20 DC CD 00 00  7D 18 17 01 00 00 00 00   À.. ÜÍ..}.......
  524592   00 00 00 00 94 00 00 00  7D 18 17 01 00 00 00 00   ....”...}.......
```

Figure 41 Winhex display before deletion image of a 0xC0 director entry

```
Seeking Relative Byte Location: 524576, For Directory Index: 10, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: C0 Directory Entry Record, Stream Extension
Secondary Flags:  03
     Flag Bit 0: Allocation Possible
     Flag Bit 1: FAT Chain Invalid
Length of UniCode Filename is: 32
Name Hash Value is:            DCCD
Stream Extension First Cluster     148 Byte Location: 19398656
Cluster     148 is Allocated
Stream Extension Data Length         18290813 Bytes Slack:   71805 Clusters Used:     140
Stream Extension Valid Data Length   18290813 Bytes Slack:   71805 Clusters Used:     140
```

Figure 42 Translation of before deletion image of a 0xC0 director entry

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  524576   40 03 00 20 DC CD 00 00  7D 18 17 01 00 00 00 00   @.. ÜÍ..}.......
  524592   00 00 00 00 94 00 00 00  7D 18 17 01 00 00 00 00   ....”...}.......
```

Figure 43 Winhex display after deletion image of a 0xC0 director entry

```
Seeking Relative Byte Location: 524576, For Directory Index: 10, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 40 Directory Entry Record, Stream Extension (Deleted)
Secondary Flags:  03
     Flag Bit 0: Allocation Possible
     Flag Bit 1: FAT Chain Invalid
Length of UniCode Filename is: 32
Name Hash Value is:            DCCD
Stream Extension First Cluster     148 Byte Location: 19398656
Cluster     148 is Not Allocated
Stream Extension Data Length         18290813 Bytes Slack:   71805 Clusters Used:     140
Stream Extension Valid Data Length   18290813 Bytes Slack:   71805 Clusters Used:     140
```

Figure 44 Translation of after deletion image of a 0xC0 director entry

The figures above are for an audio file of 18,290,813 bytes written on an exFAT file system formatted with a 128KiB cluster size. This file will require 140 clusters to hold the actual file data. Since the file will not fit exactly in the 140 cluster allocation, the last cluster will not be full. In this case, there is file slack space of 71,805 bytes which is enough to fit another file that can be hidden in that unused space. When the analysis is performed before file deletion, the FAT and Allocation Bitmaps are inspected. The program determines based on the Allocation Bitmap, that the first cluster is allocated. In order to determine if all 140 clusters are allocated, the Allocation Bitmap must be inspected for all 140 clusters, and if there is a FAT chain, the FAT has to be traversed to determine the identity and order of each of those clusters.

© 2010 The SANS Institute Author retains full rights.

If the No FAT Chain flag is set to 1 (invalid) as in this example, then the file clusters are resident in contiguous clusters and indicates that there is no file fragmentation. To determine if all the clusters are allocated, clusters 148 (the first cluster) through cluster 287 would have to have their corresponding Allocation Bitmap settings inspected.

Now the file in this example is then deleted, and the entry type in the Streams Extensions Directory Entry was changed from a 0xC0 to a 0x40 (as seen in Figure 43), all other fields remain unchanged. When the allocation status of cluster 148 is checked, the Allocation Bitmap shows that the cluster is no longer allocated. If the cluster was shown instead to be allocated, then the most likely cause would be that the cluster was reused for another file indicating that the data may have been overwritten.

Walking the FAT chains (when required) and verifying the Allocation Bitmap would be essential to the forensics examiner. If any of the clusters of a deleted file are shown to be allocated – short of a corrupted file system – it would indicate that the cluster was reassigned to another file and part of the deleted file was overlaid by another. There are many scenarios of cluster allocations and how files may get overwritten by other files, and these scenarios are beyond the scope of this paper. But when recovering deleted files, these possible scenarios need to be understood by the forensics examiner or tool that is performing the recovery of such deleted files.

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

6.10 File Name Extension Directory Entry

<table>
<tr><th>Field Name</th><th>Offset (byte)</th><th>Size (byte)</th><th>Description/Value</th></tr>
<tr><td>Entry Type</td><td>0</td><td>1</td><td>0xC1

<table>
<tr><td>Type Field</td><td>Offset</td><td>Size</td><td>Value</td></tr>
<tr><td>In Use</td><td>7</td><td>1</td><td>1</td></tr>
<tr><td>Category</td><td>6</td><td>1</td><td>1</td></tr>
<tr><td>Importance</td><td>5</td><td>1</td><td>0</td></tr>
<tr><td>Code</td><td>0</td><td>5</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>General Secondary Flags</td><td>1</td><td>1</td><td>
<table>
<tr><td>Field</td><td>Offset</td><td>Size</td><td>Value</td></tr>
<tr><td>Allocation Possible</td><td>0</td><td>1</td><td>0 – No
1 – Yes</td></tr>
<tr><td>No FAT Chain</td><td>1</td><td>1</td><td>0 – Valid
1 - Invalid</td></tr>
<tr><td>Custom</td><td>2</td><td>14</td><td></td></tr>
</table>
</td></tr>
<tr><td>File Name</td><td>2</td><td>30</td><td>Unicode part of filename is 15 characters, for a maximum of 255
Special filenames of “.” And “..” have special meanings of “this directory” and “containing directory” and shall not be recorded.</td></tr>
<tr><td colspan="4">Comments: There can be from 1 to 17 of these entries, for a maximum of 17x15 character long filenames (255 characters). The representation is 16 bit Unicode, 2 characters per directory entry. The filename character string is not null terminated.
If the In Use bit is zero (0x40) then this is probably part of a deleted file set</td></tr>
</table>

Table 17 Layout for File Name Extension Directory Entry

The 3rd entry type of a File Entry Set is the File Name Extension Directory Entry, which has an entry type of 0xC1, and with the exception of the previous two directory entries this entry may repeat multiple times, right now up to 17 times.

The File Name Extension Directory Entry is simple, it has the entry type, secondary flags, and the remaining 30 bytes are used for a segment of the filename. The general secondary flags indicate that allocation of clusters is not possible. To understand this, a typical directory entry has a standard general format that offset 20 is a 4 byte first cluster address and offset 24 is an 8 byte length value. By setting the flag that allocation is not possible, this really indicates that offset 20 does not hold a cluster address field. This allows that field to be redefined as something else and used for different data.

The file name is Unicode (16 bit) characters, and does not use null termination of the string. The actual length of the string is provided as a one byte unsigned integer in the Stream Extensions Directory Entry as Name Length. Size limitations present a value

© 2010 The SANS Institute Author retains full rights.

range of 0-255 and the file name length being currently supported is a maximum of 255 characters.

Since a File Name Extension Directory Entry can hold 15 characters per entry, up to 17 entries may be required to hold longer filenames. This is where the calculation for a file set is a maximum of 19 directory entries, a filename of 255 characters would require all 17 0xC1 entries to hold the filename. The strictest order is required because if the order is not maintained then the wrong filename would be interpreted by the file system.

In Carrier's book (Carrier, 2005) he describes file name structure of the FAT file system and the Long File Name (LFN) Entries that supplement the legacy 8.3 filename support. There are two different entry types in the legacy FAT directory for a file. These types of entries do not exist in the exFAT file system. There are no 8.3 entries within the exFAT file system, and unlike in legacy FAT where the end of the filename comes first in the directory entry order (the entries are in reverse order), the entries in exFAT are in order where the end of the filename will appear last.

Now a deeper dive is taken into the File Name Extension Directory Entries:

The full filename in the example below is: "cryptography_cryp-203-32kbps.mp3"

```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

  524608   C1 00 63 00 72 00 79 00  70 00 74 00 6F 00 67 00   Á.c.r.y.p.t.o.g.
  524624   72 00 61 00 70 00 68 00  79 00 5F 00 63 00 72 00   r.a.p.h.y. .c.r.
  524640   C1 00 79 00 70 00 2D 00  32 00 30 00 33 00 2D 00   Á.y.p.-.2.0.3.-.
  524656   33 00 32 00 6B 00 62 00  70 00 73 00 2E 00 6D 00   3.2.k.b.p.s...m.
  524672   C1 00 70 00 33 00 00 00  00 00 00 00 00 00 00 00   Á.p.3...........
  524688   00                                                 .
```

Figure 45 Winhex display of File Name Extension Directory Entry

© 2010 The SANS Institute Author retains full rights.

```
Seeking Relative Byte Location: 524608, For Directory Index: 11, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: C1 Directory Entry Record, File Name Extension
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Invalid
cryptography_cr
Seeking Relative Byte Location: 524640, For Directory Index: 12, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: C1 Directory Entry Record, File Name Extension
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Invalid
yp-203-32kbps.m
Seeking Relative Byte Location: 524672, For Directory Index: 13, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: C1 Directory Entry Record, File Name Extension
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Invalid
p3
```

Figure 46 Translation of File Name Extension Directory Entry before deletion

```
Seeking Relative Byte Location: 524608, For Directory Index: 11, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 41 Directory Entry Record, File Name Extension (Deleted)
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Valid
cryptography_cr
Seeking Relative Byte Location: 524640, For Directory Index: 12, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 41 Directory Entry Record, File Name Extension (Deleted)
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Valid
yp-203-32kbps.m
Seeking Relative Byte Location: 524672, For Directory Index: 13, Reading 1 byte, Bytes
read: 1
Root Entry Type Read is: 41 Directory Entry Record, File Name Extension (Deleted)
Secondary Flags:  00
     Flag Bit 0: Allocation Not Possible
     Flag Bit 1: FAT Chain Valid
p3
```

Figure 47 Translation of File Name Extension Directory Entry after deletion

From the File Name Extension Directory Entry (0xC0) the name length is 32 Unicode characters (each character takes 2 bytes) which will require 3 File Name Extension Directory Entries, with 2 characters in the 3rd entry. This is verified in the Winhex and program outputs in Figure 45 and Figure 46. When the file was deleted, and the "InUse" bit changed to a zero, and as shown the entry types were changed to 0x41 as displayed in Figure 47.

© 2010 The SANS Institute Author retains full rights.

7 Areas for Future Research

- Hacking the File System – Breaking the file system by changing values and observing how the file system reacts. If you change the hash values what message would you get? Would the file system even complain? What if you mark some entries of an entry set as InUse and other not InUse?
- Moving things around – What if you move the Allocation Bitmap to a different cluster, would the file system still operate properly? What if you fragment the Allocation Bitmap and put the clusters in a different order, will the file system abide by the FAT chain?
- Putting non-standard data in the directory entries – Definitely needed for forensics examination. Suppose you build a directory of 255 directory entries and put executable code or pictures in the directory entry. 255x30 = 7,650 bytes to save bytes in it. Will the file system complain? What will it do?
- Deleting part of a file set – What happens if you leave part of the directory set as InUse but delete other parts of the entries, what will the file system do?
- What happens if you change critical directory entries to benign entries?
- What happens if you create new benign entries? Will the file system mount?
- OEM Parameters – Since this record type was not encountered in testing, analysis could not be performed. When someone actually creates these entries they should be evaluated.
- TexFAT – When Microsoft releases it, it needs analysis.
- ACL – When Microsoft releases it, it needs analysis.
- In a partition – an exFAT file system should be generated in a disk partition to see what the partition code will actually be set to since the current documentation says 0x07. What does a listing of the MBR show using native Windows commands? Do they say exFAT?
- Analysis under Windows 7, does Windows 7 do anything differently?
- MAC Analysis. How and when does Windows 7 update its timestamps?

© 2010 The SANS Institute Author retains full rights.

8 Summary

The Extended FAT File System (exFAT) is a new and not yet widely used file system. It has been out for a few years and it will gain acceptance and momentum with the release of storage devices that will support the new SDXC standard. Forensics investigators and the maker of forensics tools need to be ready and prepared for an influx of acquired evidence that requires analysis of this new file structure.

The time for addressing this new file system specification is here today. The SDXC media standard was announced in January 2009. In late 2009, devices that can use this new media have been announced and they will be available in early 2010. Already there is media formatted with the exFAT file system out there and containing potential digital evidence that is being collected but with no tools to analyze them.

9 Acknowledgements

I want to thank Jeff Hamm for his assistance, including providing me with his initial research that gave me a head start for this paper, as well as collaboration on the timestamp offsets. I also want to thank X-Ways, the makers of Winhex, who provided a license for the product so I can develop templates for the research.

10 Author Information

Robert Shullich is a Graduate student in the Forensics Computing program at John Jay College of Criminal Justice, CUNY. He holds a BS and MS in Computer Science from the College of Staten Island, CUNY, MBA from Baruch College, CUNY, and a MS in Telecommunications Networking from Brooklyn Polytechnic University. He serves on the SANS Advisory Board, the IANS Technical Advisory Committee and the IDC US Events Advisory Board. With over 35 years in IT including disciplines of Mainframe Operations, Systems Programming, Program Application Development, LAN Administration, Networking, and Information Security, he holds many professional computer certifications including: CPP, CISSP, CISSP-ISSMP, CISSP-ISSAP, SSCP,

© 2010 The SANS Institute Author retains full rights.

CISA, CISM, CGEIT, CEH, CIPP, SCP/SCNA, GSEC, GSNA, GREM, GCFW, GCIH, GCFA, GAWN, MCSE+Security and Security+.

11 References

BCS SIGIST (2001). *Standard for Software Component Testing* (April 27, 2001). Retrieved December 11, 2009 from: http://www.testingstandards.co.uk/Component Testing.pdf

Carlton, Gregory H (2008). *An Evaluation of Windows-Based Computer Forensics Application Software Running on a Macintosh*, Journal of Digital Forensics, Security and Law, 3(3).

Carrier, Brian (2003). *Open Source Digital Forensics Tools: The Legal Argument*. Retrieved December 4, 2009 from: http://www.digital-evidence.org/papers/opensrc_legal.pdf

Carrier, Brian (2005). *File system forensic analysis.* Upper Saddle River, NJ: Pearson Education, Inc.

Carvey, Harlan (2005). *Windows forensics and incident recovery*. Boston, MA: Pearson Education Inc.

Casey, Eoghan (2002). *Handbook of Computer Crime Investigation*. London: Academic Press

Casey, Eoghan (2004). *Digital evidence and computer crime: forensic science, computers, and the internet* (2nd ed.). London: Academic Press.

Cormen, Thomas, Leiserson, Charles, Rivest, Ronlad & Stein, Clifford (2001). I*ntroduction to Algorithms* (2nd ed.). MIT Press

Daubert v. Merrell Dow Pharmaceuticals. *Daubert v. Merrell Dow Pharmaceuticals (92-102), 509 U.S. 579 (1993)*. Retrieved December 4, 2009 from: http://supct.law.cornell.edu/supct/html/92-102.ZS.html

Elmasri, Ramez, & Navathe, Sham (1994). *Fundamentals of database systems*. Addison Wesley Publishing Company.

Fontana, John (2009). *Microsoft expands exFAT multimedia file system licensing.* Network World (December 10, 2009). Retrieved December 15, 2009 from http://www.networkworld.com/news/2009/121009-microsoft-exfat-multimedia-file-system.html?fsrc=netflash-rss

© 2010 The SANS Institute Author retains full rights.

Galli, Peter (2009). *Tuxera Signs File System IP Agreement with Microsoft* (August 26, 2009) Retrieved December 15, 2009 from http://port25.technet.com/archive/2009/08/26/tuxera-signs-file-system-covenant-with-microsoft.aspx

Griffith, E. (2008). *OS Wars: The Battle for Your Desktop*, PC Magazine, Vol. 27, No. 4, March 1, 2008. Retrieved December 15, 2009 from http://www.pcmag.com/article2/0,2817,2273486,00.asp

Halfacree, Gareth (2009). *SDXC laptops due soon* (December 1, 2009) Retrieved December 14, 2009 from http://www.bit-tech.net/news/hardware/2009/12/01/sdxc-laptops-due-soon

Hamm Jeff (2009). *Extended FAT File System*. Presented at Techno Forensics Conference October 2009 at NIST, Retrieved January 6, 2010 from http://paradigmsolutions.files.wordpress.com/2009/12/exfat-excerpt-1-4.pdf

Herrman, John (2009). *First SDXC Card Is The World's Fastest, Only Holds 32GB*. (March 6, 2009). Retrieved November 20, 2009 from http://gizmodo.com/5165352/first-sdxc-card-is-the-worlds-fastest-only-holds-32gb

Hissink , Dennis (2009). *CES Show Report: SDXC flash memory cards* (January 7, 2009) Retrieved November 20, 2009 from http://www.ces-show.com/

History of the Floppy Disk. Wikipedia. Retrieved November 20, 2009 from http://en.wikipedia.org/wiki/Floppy_disk

History and Capacities of CDROM. Wikipedia. Retrieved November 20, 2009 from http://en.wikipedia.org/wiki/Cdrom

History and Capacities of DVD. Wikipedia. Retrieved November 20, 2009 from http://en.wikipedia.org/wiki/DVD

History and Capacities of Blue Ray Disc. Wikipedia. Retrieved November 20, 2009 from http://en.wikipedia.org/wiki/Blue_ray

HPC Factor (2009). *The History of Windows CE: Windows CE 6.0 & into the future....* Retrieved October 9, 2009 from http://www.hpcfactor.com/support/windowsce/wce6.asp

International System of Units (SI). Retrieved November 10, 2009 from http://physics.nist.gov/cuu/Units/binary.html

Johnston, Stuart (2009). Microsoft Licenses exFAT to Third Parties. (December 10, 2009). Internetnews.com. Retrieved December 15, 2009

© 2010 The SANS Institute Author retains full rights.

http://www.internetnews.com/software/article.php/3852686/Microsoft+Licenses+exFAT+to+Third+Parties.htm

Larkin, Eric (2007). *Vista Resistance: Why XP Is Still So Strong*, September 26, 2007. Retrieved December 15, 2009 http://www.pcworld.com/article/137635/vista_resistance_why_xp_is_still_so_strong.htm

Microsoft Intellectual Property Licensing for exFAT. Retrieved December 10, 2009 from http://www.microsoft.com/iplicensing/productDetail.aspx?productTitle=exFAT File System Licensing Program

Microsoft MSDN AA914663. *OEM Parameter Definition with exFAT*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/aa914663.aspx

Microsoft MSDN EE681827. *File System Functionality Comparison*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/ee681827(VS.85).aspx

Microsoft MSDN CC907928. *TexFAT File Naming Limitations*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/cc907928.aspx

Microsoft Patent 0164440 (June 25, 2009). *Quick Filename Lookup Using Name Hash. Pub No. US 2009/0164440 A1* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat20090164440.pdf

Microsoft Patent 0265400 (October 22, 2009). *Extensible File System. Pub No. US 2009/0265400 A1* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat20090265400.pdf

Microsoft Patent 7613738 (November 3, 2009). *FAT Directory Structure for use in Transaction Safe File System. Pub No. US 7613738 B2* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat7613738.pdf

Microsoft Press Pass (2009). *Microsoft's Latest Flash Memory Technology Now Available for License*. (December 10, 2009). Retrieved December 10, 2009 from http://www.microsoft.com/presspass/press/2009/dec09/12-10msflashtechpr.mspx

Microsoft (2004). *Local File Systems for Windows.* (May 5, 2004) Retrieved December 10, 2009 from http://www.microsoft.com/whdc/device/storage/LocFileSys.mspx

© 2010 The SANS Institute Author retains full rights.

Microsoft (2008). *Microsoft Notable Changes in Windows Vista Service Pack 1* (March 2008). Retrieved December 10, 2009 from http://technet.microsoft.com/en-us/library/cc709618(WS.10).aspx

Microsoft (September 2009). *Description of the exFAT file system driver update package. Q955704*. Retrieved November 10, 2009 from http://support.microsoft.com/kb/955704

Mueller, Scott (2003). *Upgrading and Repairing PCs*. Que.

NTFS.COM. *NTFS vs. FAT*. Retrieved December 10, 2009 from http://www.ntfs.com/ntfs_vs_fat.htm

SD Card Association. *SDXC*. Retrieved December 10, 2009 from http://www.sdcard.org/developers/tech/sdxc

US Department Of Justice (2004). *Forensic Examination of Digital Evidence: A Guide for Law Enforcement*. Retrieved December 10, 2009 from http://www.ncjrs.gov/pdffiles1/nij/199408.pdf

Yahoo News (December 3rd, 2009). *The World's First Data Recovery for exFAT Drives!* Retrieved December 8, 2009 from http://news.yahoo.com/s/prweb/20091203/bsprweb/prweb3275634

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

12 Appendix

Table of Where Things Are

Reverse Engineering the Microsoft Extended FAT File System (exFAT) 1
GIAC (GCFA) Gold Certification 1
ABSTRACT 1
1 Introduction 2
2 Definitions 3
3 Prior Work 4
4 Setting a Foundation 4
4.1 Purpose, Disclaimer and Scope 4
4.1.1 Purpose 4
4.1.2 Disclaimer 4
4.1.3 Assumptions 5
4.1.4 Out of Scope 5
4.2 Relevance to the Field of Digital Forensics 6
4.3 Research Methodology 11
4.4 Survey of Removable Media 13
4.5 Survey of Microsoft File Systems 15
4.6 Getting the drivers put onto Windows XP 16
4.7 International System of Units (SI) Table 18
4.8 Summary of exFAT Features 19
4.9 exFAT Timeline (Key Dates) 19
4.10 Maximum Volume and File Limitations 20
5 exFAT Internals 22
5.1 Volume Structure 22
5.2 Volume Boot Record (VBR) 24
5.3 File Allocation Table (FAT) 30
5.4 Allocation Bitmap Table 36
5.5 Time Stamp Format 38
5.6 Cluster Heap 41
5.7 Transactional FAT 41
6 exFAT Directory Structure 42
6.1 Root Directory 45
6.2 Volume Label Directory Entry 45
6.3 Allocation Bitmap Directory Entry 46
6.4 UP-Case Table Directory Entry 47
6.5 Volume GUID Directory Entry 48
6.6 TexFAT Padding Directory Entry 49
6.7 Windows CE Access Control Table Directory Entry 49
6.8 File Directory Entry 50
6.9 Stream Extension Directory Entry 57

Robert Shullich rshullic@earthlink.net

© 2010 The SANS Institute Author retains full rights.

6.10 File Name Extension Directory Entry 60
7 Areas for Future Research 63
8 Summary 64
9 Acknowledgements 64
10 Author Information 64
11 References 65
12 Appendix 69
Table of Where Things Are 69
List of Tables 70
List of Figures 71
Table of Authorities 72
12.1 Glossary 75
12.2 Partition Master Boot Record Partition Layout 77
12.3 List of selected Partition Codes 78
12.4 SDXC Formats 79
12.5 Disassembly of the VBR 80
12.6 Time Zone Offset Table 81
12.7 Winhex Sample VBR Template 83
12.8 Winhex Sample VBR Template Output 84

List of Tables

Table 1 Numbering Schemes 18
Table 2 File System Limits 20
Table 3 Layout for Main and Backup Boot Sector Structure 24
Table 4 Layout for Extended Boot Sector Structure 28
Table 5 Layout for OEM Parameter Structure 28
Table 6 Layout for the File Allocation Table (FAT) 30
Table 7 Media Descriptor Definitions as used in legacy FAT file systems 31
Table 8 Breakdown of the Entry Type 44
Table 9 Layout for Volume Label Directory Entry 45
Table 10 Layout for Allocation Bitmap Directory Entry 46
Table 11 Layout for UP-Case Table Directory Entry 47
Table 12 Layout for Volume GUID Directory Entry 48
Table 13 Layout for TexFAT Padding Directory Entry 49
Table 14 Layout for Windows CE Access Control Table Directory Entry 49
Table 15 Layout for File Directory Entry 50
Table 16 Layout for Stream Extension Directory Entry 57
Table 17 Layout for File Name Extension Directory Entry 60
Table 18 Acronym Table 76
Table 19 Layout of one 16-byte partition record 77
Table 20 Some Partition Type Definitions 78
Table 21 SDXC Formats 79
Table 22 Disassembly of 1st VBR Boot Sector 80
Table 23 Time Zone Index Offset Table 82

© 2010 The SANS Institute Author retains full rights.

List of Figures

Figure 1 Disk Properties of exFAT file system using Windows XP without exFAT support 7
Figure 2 Dir command on Windows XP system without the exFAT drivers 8
Figure 3 Opening exFAT media in Windows Explorer on an XP system without the exFAT drivers 8
Figure 4 Screenshot of FTK Toolkit 1.81.5 Analysis of exFAT media 8
Figure 5 File Signatures of a BMP (Top) and an EXE (Bottom) 9
Figure 6 Compact Flash, SDXC, and Smart Media and SD cards 14
Figure 7 Jan 2009, Memory Card Market Share, ... 14
Figure 8 Step 1 – Invoke Update KB955704 16
Figure 9 Step 2 – Agree to the License Agreement 16
Figure 10 Step 3– KB955704 begins to update 17
Figure 11 Step 4 – KB955704 Completed, now reboot the system 17
Figure 12 Format Help command on XP after KB955704 18
Figure 13 Extended FAT File System (exFAT) Volume Layout 22
Figure 14 Winhex Display of VBR Signature 25
Figure 15 Winhex of the first 120 bytes of a MBS 26
Figure 16 Chkdsk of an exFAT formatted disk 27
Figure 17 OEM Parameters Type Definition 29
Figure 18 Winhex dump of part of a VBR checksum sector 29
Figure 19 Code snippet of VBR checksum calculation function in C 30
Figure 20 Attempt to format a 1.44 floppy disk with an exFAT file system 32
Figure 21 Winhex display of 16 FAT cells 32
Figure 22 Program simulated Chkdsk totals 33
Figure 23 Extended FAT File System (exFAT) Example 35
Figure 24 Extended FAT File System (exFAT) Allocation Bitmap Example 36
Figure 25 Extended FAT File System (exFAT) Timestamp Format 38
Figure 26 The DOS Date/Time format 39
Figure 27 Winhex display of FAT 32 Subdirectory for special pointers 42
Figure 28 Winhex Template of the "." subdirectory in FAT32 43
Figure 29 Winhex Template of the ".." subdirectory in FAT32 43
Figure 30 Checksum routine for the UP-Case Table 48
Figure 31 File Entry Set created by Server 2008 SP1 51
Figure 32 Display of File Properties for exFAT created on Server 2008 51
Figure 33 Display of File Properties for NTFS on Windows XP 52
Figure 34 Display of dates using Windows Explorer 52
Figure 35 Winhex of 0x85 entry created on Windows XP SP3 53
Figure 36 File Entry Set Checksum Calculation in C 54
Figure 37 Winhex Dump of a 0x85 image before deletion 55
Figure 38 Formatted translation of a 0x85 image before deletion 55
Figure 39 Winhex dump of a 0x85 image after deletion 55
Figure 40 Formatted translation of a 0x85 image after deletion 55
Figure 41 Winhex display before deletion image of a 0xC0 director entry 58

Robert Shullich rshullic@earthlink.net

© 2010 The SANS Institute Author retains full rights.

Figure 42 Translation of before deletion image of a 0xC0 director entry 58
Figure 43 Winhex display after deletion image of a 0xC0 director entry 58
Figure 44 Translation of after deletion image of a 0xC0 director entry 58
Figure 45 Winhex display of File Name Extension Directory Entry..... 61
Figure 46 Translation of File Name Extension Directory Entry before deletion 62
Figure 47 Translation of File Name Extension Directory Entry after deletion 62
Figure 48 Sample Winhex Template that I developed as part of this research..... 83
Figure 49 Output of Sample Winhex Template that were developed as part of this research 84

Table of Authorities

Cases

BCS SIGIST (2001). *Standard for Software Component Testing* (April 27, 2001). Retrieved December 11, 2009 from http://www.testingstandards.co.uk/Component Testing.pdf 15, 69

Carlton, Gregory H (2008). *An Evaluation of Windows-Based Computer Forensics Application Software Running on a Macintosh*, Journal of Digital Forensics, Security and Law, 3(3). 11, 69

Carrier, Brian (2003). *Open Source Digital Forensics Tools: The Legal Argument*. Retrieved December 4, 2009 from: http://www.digital-evidence.org/papers/opensrc_legal.pdf..... 13, 69

Carrier, Brian (2005). *File system forensic analysis.* Upper Saddle River, NJ: Pearson Education, Inc.. ..... 34, 65, 69

Carvey, Harlan (2005). *Windows forensics and incident recovery*. Boston, MA: Pearson Education Inc. 11, 42, 69

Casey, Eoghan (2002). *Handbook of Computer Crime Investigation*. London: Academic Press 42, 69

Casey, Eoghan (2004). *Digital evidence and computer crime: forensic science, computers, and the internet* (2nd ed.). London: Academic Press. 7, 69

Cormen, Thomas, Leiserson, Charles, Rivest, Ronlad & Stein, Clifford (2001). Introduction to Algorithms (2nd ed.). MIT Press..... 35, 69

Daubert v. Merrell Dow Pharmaceuticals. *Daubert v. Merrell Dow Pharmaceuticals (92-102), 509 U.S. 579 (1993)*. Retrieved December 4, 2009 from: http://supct.law.cornell.edu/supct/html/92-102.ZS.html 13, 69

Elmasri, Ramez, & Navathe, Sham (1994). *Fundamentals of database systems*. Addison Wesley Publishing Company. 45, 69

Fontana, John (2009). *Microsoft expands exFAT multimedia file system licensing.* Network World (December 10, 2009). Retrieved December 15, 2009 from: http://www.networkworld.com/news/2009/121009-microsoft-exfat-multimedia-file-system.html?fsrc=netflash-rss..... 15, 69

Galli, Peter (2009). *Tuxera Signs File System IP Agreement with Microsoft* (August 26, 2009) Retrieved December 15, 2009 from http://port25.technet.com/archive/2009/08/26/tuxera-signs-file-system-covenant-with-microsoft.aspx: 23, 70

© 2010 The SANS Institute Author retains full rights.

Halfacree, Gareth (2009). *SDXC laptops due soon* (December 1, 2009) Retrieved December 14, 2009 from http://www.bit-tech.net/news/hardware/2009/12/01/sdxc-laptops-due-soon: 15, 24
Hamm Jeff (2009). *Extended FAT File System*. Presented at Techno Forensics Conference October 2009 at NIST, Retrieved January 6, 2010 from http://paradigmsolutions.files.wordpress.com/2009/12/exfat-excerpt-1-4.pdf. 60, 70
Herrman, John (2009). *First SDXC Card Is The World's Fastest, Only Holds 32GB*. (March 6, 2009). Retrieved November 20, 2009 from http://gizmodo.com/5165352/first-sdxc-card-is-the-worlds-fastest-only-holds-32gb: 23, 70
Hissink , Dennis (2009). *CES Show Report: SDXC flash memory cards* (January 7,2009) Retrieved November 20, 2009 from: http://www.ces-show.com/ 6, 15, 23, 70
HPC Factor (2009). *The History of Windows CE: Windows CE 6.0 & into the future...*. Retrieved October 9, 2009 from: http://www.hpcfactor.com/support/windowsce/wce6.asp 23, 70
International System of Units (SI). Retrieved November 10, 2009 from http://physics.nist.gov/cuu/Units/binary.html 22, 23, 70
Johnston, Stuart (2009). Microsoft Licenses exFAT to Third Parties. (December 10, 2009). Internetnews.com. Retrieved December 15, 2009 from http://www.internetnews.com/software/article.php/3852686/Microsoft+Licenses+exFAT+to+Third+Parties.htm 24, 71
Larkin, Eric (2007). *Vista Resistance: Why XP Is Still So Strong*, September 26, 2007. Retrieved December 15, 2009 http://www.pcworld.com/article/137635/vista_resistance_why_xp_is_still_so_strong.htm 11, 71
Mueller, Scott (2003). *Upgrading and Repairing PCs*. Que. 34, 72
NTFS.COM. *NTFS vs. FAT*. Retrieved December 10, 2009 from http://www.ntfs.com/ntfs_vs_fat.htm 25, 72
SD Card Association. *SDXC*. Retrieved December 10, 2009 from http://www.sdcard.org/developers/tech/sdxc 72
US Department Of Justice (2004). *Forensic Examination of Digital Evidence: A Guide for Law Enforcement*. Retrieved December 10, 2009 from http://www.ncjrs.gov/pdffiles1/nij/199408.pdf 6, 7, 72
Yahoo News (December 3rd, 2009). *The World's First Data Recovery for exFAT Drives!* Retrieved December 8, 2009 from http://news.yahoo.com/s/prweb/20091203/bs_prweb/prweb3275634 ..... 15, 24, 72

Patents

Microsoft Patent 0164440 (June 25, 2009). *Quick Filename Lookup Using Name Hash. Pub No. US 2009/0164440 A1* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat20090164440.pdf 6, 9, 16, 71
Microsoft Patent 0265400 (October 22, 2009). *Extensible File System. Pub No. US 2009/0265400 A1* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat20090265400.pdf 16, 71
Microsoft Patent 7613738 (November 3, 2009). *FAT Directory Structure for use in Transaction Safe File System. Pub No. US 7613738 B2* Retrieved December 10, 2009 from http://www.pat2pdf.org/patents/pat7613738.pdf 16, 71

© 2010 The SANS Institute Author retains full rights.

10

History and Capacities of Blue Ray Disc. Wikipedia. Retrieved November 20, 2009 from: http://en.wikipedia.org/wiki/Blue_ray 17, 70

History and Capacities of CDROM. Wikipedia. Retrieved November 20, 2009 from: http://en.wikipedia.org/wiki/Cdrom..... 17, 70

History and Capacities of DVD. Wikipedia. Retrieved November 20, 2009 from: http://en.wikipedia.org/wiki/DVD 17, 70

History of the Floppy Disk. Wikipedia. Retrieved December 10, 2009 from http://en.wikipedia.org/wiki/Floppy_disk..... 17, 70

13

Microsoft (2004). *Local File Systems for Windows.* (May 5, 2004) Retrieved December 10, 2009 from http://www.microsoft.com/whdc/device/storage/LocFileSys.mspx 20, 71

Microsoft (2008). *Microsoft Notable Changes in Windows Vista Service Pack 1* (March 2008). Retrieved December 10, 2009 from http://technet.microsoft.com/en-us/library/cc709618(WS.10).aspx..... 23, 72

Microsoft (September 2009). *Description of the exFAT file system driver update package. Q955704*. Retrieved November 10, 2009 from http://support.microsoft.com/kb/955704 ..... 20, 23, 72

Microsoft Intellectual Property Licensing for exFAT. Retrieved December 10, 2009 from http://www.microsoft.com/iplicensing/productDetail.aspx?productTitle=exFAT File System Licensing Program 8, 71

Microsoft MSDN AA914663. *OEM Parameter Definition with exFAT*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/aa914663.aspx ... 33, 71

Microsoft MSDN CC907928. *TexFAT File Naming Limitations*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/cc907928.aspx..... 46, 71

Microsoft MSDN EE681827. *File System Functionality Comparison*. Retrieved December 10, 2009 from http://msdn.microsoft.com/en-us/library/ee681827(VS.85).aspx 24, 71

Microsoft Press Pass (2009). *Microsoft's Latest Flash Memory Technology Now Available for License*. (December 10, 2009). Retrieved December 10, 2009 from http://www.microsoft.com/presspass/press/2009/dec09/12-10msflashtechpr.mspx ... 24, 25, 71

© 2010 The SANS Institute Author retains full rights.

12.1 Glossary

Acronym	Full Text
ACID	Atomicity, Consistency, Isolation and Durability
ACL	Access Control List
ASCII	American Standard Code for Information Exchange
AVI	Audio Video Interface
BIOS	Basic Input Output System
BMBS	Backup Main Boot Sector
BMEBS	Backup Main Extended Boot sector
BPB	BIOS Parameter Block
CD	Compact Disc
CF	Compact Flash (Media Card used in Cameras)
CPU	Central processing Unit
CR-R	Compact Disc – Read only
CR-RW	Compact Disc – Read/Write
DOS	Disk Operating System
DVD	Digital Versatile Disc or Digital Video Disc
DVD-R	DVD - Read Only
DVD-RW	DVD – Read/Write
EB	Exabytes (1000^6)
EBCDIC	Extended Binary Coded Decimal Interchange Code
EOF	End Of File
exFAT	Extensible File Allocation Table
FAT	File Allocation Table
FAT12	File Allocation Table, 12-bit cluster indices
FAT16	File Allocation Table, 16-bit cluster indices
FAT32	File Allocation Table, 32-bit cluster indices
FAT64	File Allocation Table, Nickname for exFAT
FTK	Forensics Tool Kit
GB	Gigabytes (1000^3)
GMT	Greenwich Mean Time
GPS	Global Positioning Satellite
GPT	GUID Partition Table
GUID	Globally Unique Identifier
HPFS	High Performance File System
INT	Interrupt
JPEG	Joint Photographic Experts Group
KB	Kilobytes (1000)
KB	Knowledge Base
LFN	Long File Name
MAC	Modified Date, Accessed Date, Create Date
MB	Megabytes (1000^2)
MBR	Master Boot Record
MBS	Main Boot Sector
MEBS	Main Extended Boot Sector
MS	Milliseconds
NIST	National Institute of Standards and Technology

© 2010 The SANS Institute Author retains full rights.

NT	New Technology (Windows NT)
NTFS	NT File System
OEM	Original Equipment Manufacture
PB	Petabytes (1000^5)
PDA	Personal Digital Assistants
SD	Secure Digital (Media Card used in Cameras, PDA, GPS and other devices)
SDHC	Secure Digital High Capacity
SDXC	Secure Digital eXtended Capacity media, might just use XC.
SM	Smart Media (Media Card used in earlier digital cameras)
TB	Terabytes (1000^4)
TexFAT	Transaction-safe exFAT
UDF	Universal Disk Format
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VBR	Volume Boot Record
Windows CE	Windows Consumer Electronics
XC	eXtended Capacity
YB	Yottabytes (1000^8)
ZB	Zetabytes (1000^7)

Table 18 Acronym Table

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

12.2 Partition Master Boot Record Partition Layout

Offset (hex)	Field length (bytes)	Description
0x00	1	status 0x80 = bootable (*active*) 0x00 = non-bootable, other = invalid[
0x01	3	CHS address of first block in partition The format is described in the next 3 bytes.
0x01	1	head
0x02	1	sector is in bits 5–0; bits 9–8 of cylinder are in bits 7–6
0x03	1	bits 7–0 of cylinder
0x04	1	partition type[
0x05	3	CHS address of last block in partition. The format is described in the next 3 bytes.
0x05	1	head
0x06	1	sector is in bits 5–0; bits 9–8 of cylinder are in bits 7–6
0x07	1	bits 7–0 of cylinder
0x08	4	LBA of first sector in the partition
0x0C	4	number of blocks in partition, in Little-Endian format
Source: http://en.wikipedia.org/wiki/Master_boot_record		

Table 19 Layout of one 16-byte partition record

© 2010 SANS Institute

© 2010 The SANS Institute Author retains full rights.

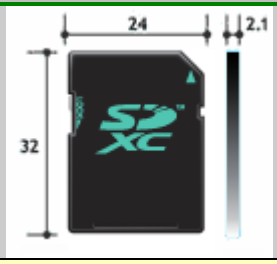
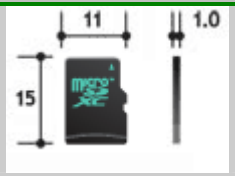
12.3 List of selected Partition Codes

Type	Description
0x00	Empty
0x01	FAT12
0x04	FAT16, 16~32MB
0x05	Microsoft Extended Partition
0x06	FAT16, 32MB~2GB
0x07	OS/2 IFS (e.g., HPFS)
0x07	exFAT
0x07	Advanced Unix
0x07	Windows NT NTFS
0x08	AIX boot partition
0x0a	OS/2 Boot Manager
0x0b	WIN95 OSR2 FAT32
0x0c	WIN95 OSR2 FAT32, LBA-mapped
0x0e	WIN95: DOS 16-bit FAT, LBA-mapped
0x0f	WIN95: Extended partition, LBA-mapped
0x82	Solaris x86
0x82	Linux Swap
0x83	Linux native partition
0x85	Linux Extended
0xa5	BSD/386, 386BSD, NetBSD, FreeBSD
0xa6	OpenBSD
0xa8	Mac OS-X
0xee	EFI GPT Disk
0xfb	VMware File System
0xfc	VMware Swap partition
Source: http://www.win.tue.nl/~aeb/partitions/partition_types-1.html	

Table 20 Some Partition Type Definitions

© 2010 The SANS Institute Author retains full rights.

12.4 SDXC Formats

	SDXC	microSDXC
Size		
Area	768 mm^2	165 mm^2
Card Volume	1,613 mm^3	165 mm^3
Thickness	2.1 mm	1.0 mm
Weight	Approx. 2g	Approx. 0.5g
Number of pins	9 pins	8 pins
File System	exFAT	exFAT
Operating Voltage	2.7V - 3.6V	2.7V - 3.6V
Write-protect Switch	YES	NO
Copyright protection	CPRM	CPRM
Compatibility	-	Yes (with adapter)
Capacity	Over 32 GB - 2 TB	Over 32 GB - 2 TB

Source: http://www.sdcard.org/developers/tech/sdxc

Table 21 SDXC Formats

© 2010 SANS Institute

© 2010 The SANS Institute Author retains full rights.

12.5 Disassembly of the VBR

```
                jmp     short BootCode
                db  90h
aExfat          db 'EXFAT   '           ; OEM Label
                db 35h dup(0)           ; Must Be Zero
                db 38h dup(0)           ; Rest of Non-Boot Code of sector
BootCode:       xor     cx, cx         ; Beginning of Boot Code
                mov     ss, cx
                mov     sp, 7BF0h
                mov     ds, cx
                mov     al, byte_7DFB
Offset84:       mov     ah, 7Dh
                mov     si, ax
offset88:       lodsb
                cbw
                inc     ax
                jz      offset99
                dec     ax
                jz      offset9E
                mov     ah, 0Eh
                mov     bx, 7
                int     10h   ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                              ; AL = character, BH = display page (alpha modes)
                              ; BL = foreground color (graphics modes)
                jmp     short Offset88
offset99:       mov     al, byte 7DFD
                jmp     short offset84
offset9e:       int     16h             ; KEYBOARD -
                int     19h             ; DISK BOOT
                                        ; causes reboot       of disk system
                db 5Eh dup(0)
aRemoveDisksOrO       db 0Dh,0Ah
                db 'Remove disks or other media.',0FFh,0Dh,0Ah
                db 'Disk error',0FFh,0Dh,0Ah
                db 'Press any key to restart',0Dh,0Ah
                dw 3Bh dup(0)
                db 3Dh dup(0FFh)
                db    0
                db  1Fh
                db  2Ch
                db 55h, 0AAh    ; Signature for 1st VBR Sector - Main Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 2nd VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 3rd VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 4th VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 5th VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 6th VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 7th VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 8th VBR Sector - Main Extended Boot Sector
                db 1FEh dup(0)
                db 55h, 0AAh    ; Signature for 9th VBR Sector - Main Extended Boot Sector
                db 200h dup(0)  ; OEM Parms Sector
                db 200h dup(0)  ; Reserved Sector
                dd 40h  dup(0B0EB2FFEh)       ; CheckSum Sector
```

Table 22 Disassembly of 1st VBR Boot Sector

© 2010 The SANS Institute Author retains full rights.

12.6 Time Zone Offset Table

Index	Offset	Time Zone Description
252	-01	Azores Standard Time (GMT-01:00) Cape Verde Standard Time (GMT-01:00)
248	-02	Mid-Atlantic Standard Time (GMT-02:00)
244	-03	E. South America Standard Time (GMT-03:00) S.A. Eastern Standard Time (GMT-03:00) Greenland Standard Time (GMT-03:00)
242	-03:30	Newfoundland Standard Time (GMT-03:30)
240	-04	Atlantic Standard Time (GMT-04:00) S.A. Western Standard Time (GMT-04:00) Pacific S.A. Standard Time (GMT-04:00)
236	-05	Eastern Standard Time (GMT-05:00) U.S. Eastern Standard Time (GMT-05:00) S.A. Pacific Standard Time (GMT-05:00)
232	-06	Central Standard Time (GMT-06:00) Canada Central Standard Time (GMT-06:00) Mexico Standard Time (GMT-06:00) Central America Standard Time (GMT-06:00)
228	-07	Mountain Standard Time (GMT-07:00) Mexico Standard Time 2 (GMT-07:00) U.S. Mountain Standard Time (GMT-07:00)
224	-08	Pacific Standard Time (GMT-08:00)
220	-09	Alaskan Standard Time (GMT-09:00)
216	-10	Hawaiian Standard Time (GMT-10:00)
212	-11	Samoa Standard Time (GMT-11:00)
208	-12	Dateline Standard Time (GMT-12:00)
180	+13	Tonga Standard Time (GMT+13:00)
176	+12	New Zealand Standard Time (GMT+12:00) Fiji Islands Standard Time (GMT+12:00)
172	+11	Central Pacific Standard Time (GMT+11:00)
168	+10	West Pacific Standard Time (GMT+10:00) Vladivostok Standard Time (GMT+10:00) Tasmania Standard Time (GMT+10:00) E. Australia Standard Time (GMT+10:00) A.U.S. Eastern Standard Time (GMT+10:00)
166	+09:30	A.U.S. Central Standard Time (GMT+09:30) Cen. Australia Standard Time (GMT+09:30)
164	+09	Yakutsk Standard Time (GMT+09:00) Tokyo Standard Time (GMT+09:00) Korea Standard Time (GMT+09:00)
160	+08	North Asia East Standard Time (GMT+08:00) W. Australia Standard Time (GMT+08:00) Taipei Standard Time (GMT+08:00) Singapore Standard Time (GMT+08:00) China Standard Time (GMT+08:00)
156	+07	North Asia Standard Time (GMT+07:00) S.E. Asia Standard Time (GMT+07:00)
154	+06:30	Myanmar Standard Time (GMT+06:30)
152	+06	N. Central Asia Standard Time (GMT+06:00) Sri Lanka Standard Time (GMT+06:00)

© 2010 The SANS Institute Author retains full rights.

		Central Asia Standard Time (GMT+06:00)
151	+05:45	Nepal Standard Time (GMT+05:45)
150	+05:30	India Standard Time (GMT+05:30)
148	+05	West Asia Standard Time (GMT+05:00) Ekaterinburg Standard Time (GMT+05:00)
146	+04:30	Afghanistan Standard Time (GMT+04:30)
144	+04	Caucasus Standard Time (GMT+04:00) Arabian Standard Time (GMT+04:00)
142	+03:30	Iran Standard Time (GMT+03:30)
140	+03	Arabic Standard Time (GMT+03:00) E. Africa Standard Time (GMT+03:00) Arab Standard Time (GMT+03:00) Russian Standard Time (GMT+03:00)
136	+02	South Africa Standard Time (GMT+02:00) Israel Standard Time (GMT+02:00) GTB Standard Time (GMT+02:00) FLE Standard Time (GMT+02:00) Egypt Standard Time (GMT+02:00) E. Europe Standard Time (GMT+02:00)
132	+01	W. Central Africa Standard Time (GMT+01:00) W. Europe Standard Time (GMT+01:00) Romance Standard Time (GMT+01:00) Central European Standard Time (GMT+01:00) Central Europe Standard Time (GMT+01:00)
128	+00	Greenwich Standard Time (GMT)

Table 23 Time Zone Index Offset Table

© 2010 SANS Institute, Author retains full rights.

© 2010 The SANS Institute Author retains full rights.

12.7 Winhex Sample VBR Template

```
template "Boot Sector exFAT"

// Template by Robert Shullich
// John Jay College of Criminal Justice

// To be applied to the first VBR sector of a exFAT-formatted logical drive.
// This template assumes a DD acquired image

description "BIOS parameter block (BPB) and more"
applies_to file
sector-aligned



begin
        read-only hex 3 "JMP instruction"
        char[8] "OEM"

        goto     0x0040

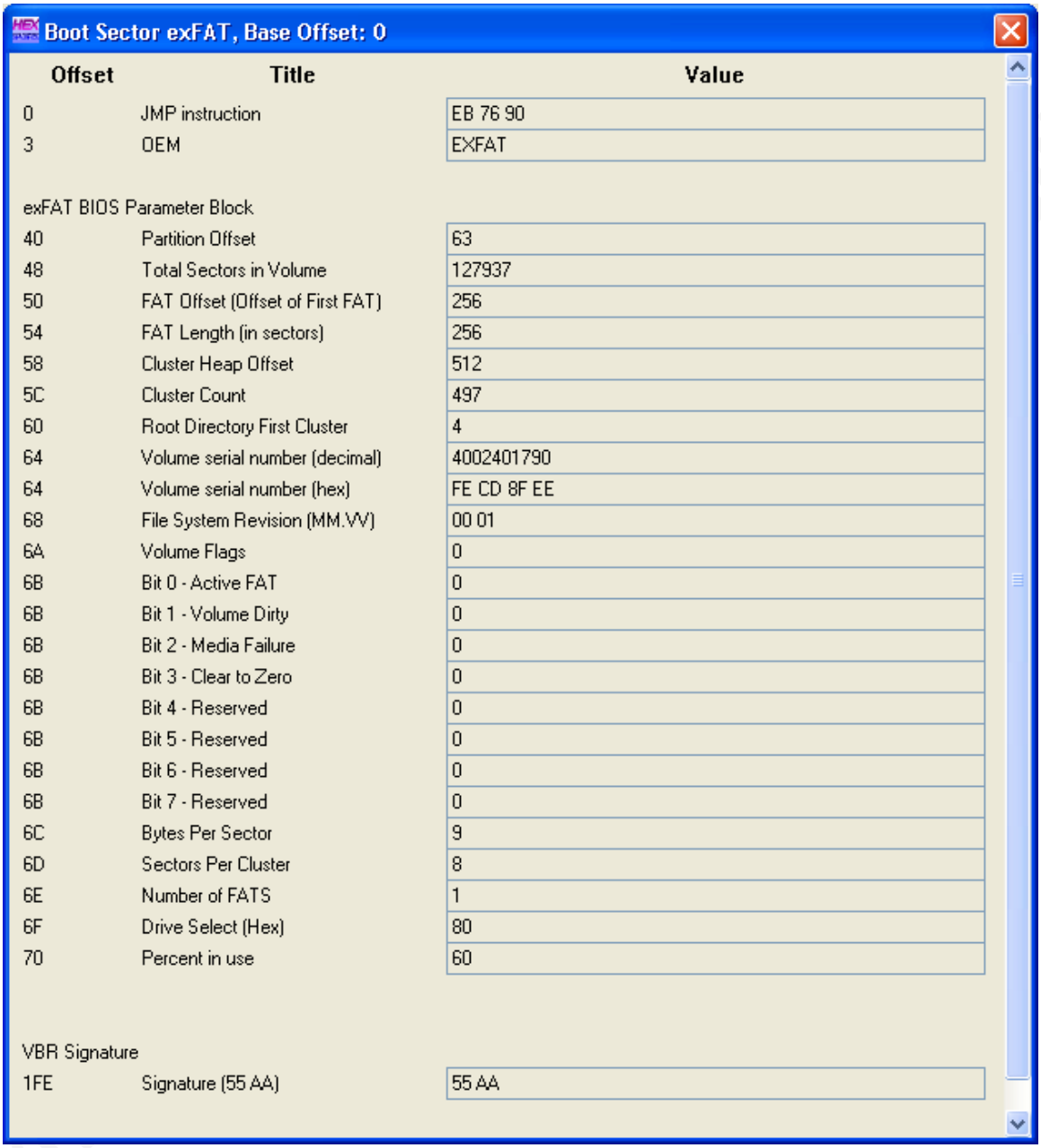
        section                         "exFAT BIOS Parameter Block"
        int64                           "Partition Offset"
        int64                           "Total Sectors in Volume"
        uint32                          "FAT Offset (Offset of First FAT)"
        uint32                          "FAT Length (in sectors)"
        uint32                          "Cluster Heap Offset"
        uint32                          "Cluster Count"
        uint32                          "Root Directory First Cluster"
        uint32                          "Volume serial number (decimal)"
        move -4
        hex 4                           "Volume serial number (hex)"
        hex 2                           "File System Revision (MM.VV)"
        uint16                          "Volume Flags"
        move -1
        uint_flex "0" "Bit 0 - Active FAT"
        move -4
        uint flex "1" "Bit 1 - Volume Dirty"
        move -4
        uint_flex "2" "Bit 2 - Media Failure"
        move -4
        uint flex "3" "Bit 3 - Clear to Zero"
        move -4
        uint flex "4" "Bit 4 - Reserved"
        move -4
        uint flex "5" "Bit 5 - Reserved"
        move -4
        uint_flex "6" "Bit 6 - Reserved"
        move -4
        uint flex "7" "Bit 7 - Reserved"
        move -3
        uint8                           "Bytes Per Sector"
        uint8                           "Sectors Per Cluster"
        uint8                           "Number of FATS"
        hex 1                           "Drive Select (Hex)"
        uint8                           "Percent in use"
        endsection

        section                         "VBR Signature"
        goto           0x1FE
        read-only hex 2 "Signature (55 AA)"
        endsection
end
```

Figure 48 Sample Winhex Template that I developed as part of this research

© 2010 The SANS Institute Author retains full rights.

12.8 Winhex Sample VBR Template Output



Boot Sector exFAT, Base Offset: 0

Offset	Title	Value
0	JMP instruction	EB 76 90
3	OEM	EXFAT
exFAT BIOS Parameter Block		
40	Partition Offset	63
48	Total Sectors in Volume	127937
50	FAT Offset (Offset of First FAT)	256
54	FAT Length (in sectors)	256
58	Cluster Heap Offset	512
5C	Cluster Count	497
60	Root Directory First Cluster	4
64	Volume serial number (decimal)	4002401790
64	Volume serial number (hex)	FE CD 8F EE
68	File System Revision (MM.VV)	00 01
6A	Volume Flags	0
6B	Bit 0 - Active FAT	0
6B	Bit 1 - Volume Dirty	0
6B	Bit 2 - Media Failure	0
6B	Bit 3 - Clear to Zero	0
6B	Bit 4 - Reserved	0
6B	Bit 5 - Reserved	0
6B	Bit 6 - Reserved	0
6B	Bit 7 - Reserved	0
6C	Bytes Per Sector	9
6D	Sectors Per Cluster	8
6E	Number of FATS	1
6F	Drive Select (Hex)	80
70	Percent in use	60
VBR Signature		
1FE	Signature (55 AA)	55 AA

Figure 49 Output of Sample Winhex Template that were developed as part of this research

© 2010 The SANS Institute Author retains full rights.