

SANS

www.sans.org

FORENSICS 518
MAC FORENSIC
ANALYSIS

518.1

Mac Essentials
and the
HFS+ File System

The right security training for your staff, at the right time, in the right location.

Copyright © 2015, The SANS Institute. All rights reserved. The entire contents of this publication are the property of the SANS Institute.

IMPORTANT-READ CAREFULLY:

This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

SANS acknowledges that any and all software and/or tools presented in this courseware are the sole property of their respective trademark/registered/copyright owners.

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.



FOR518 - Section 1

Mac Essentials & the HFS+ File System

The **SANS** Institute



Sarah Edwards
oompa@csh.rit.edu
@iamevl twin



@sansforensics

<http://computer-forensics.sans.org>

© SANS.
All Rights Reserved

Mac Forensic Analysis

Author: Sarah Edwards
oompa@csh.rit.edu
<http://twitter.com/iamevl twin>
<http://twitter.com/sansforensics>

SANS DFIR
DIGITAL FORENSICS & INCIDENT RESPONSE

Website
digital-forensics.sans.org

SIFT Workstation
dfir.to/SANS-SIFT

Join The SANS DFIR Community

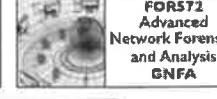
-  Blog: dfir.to/DFIRBlog
-  Twitter: [@sansforensics](https://twitter.com/sansforensics)
-  Facebook: [sansforensics](https://facebook.com/sansforensics)
-  Google+: [gplus.to/sansforensics](https://plus.google.com/sansforensics)
-  Mailing list: dfir.to/MAIL-LIST
-  YouTube: dfir.to/DFIRCast

DFIR CURRICULUM

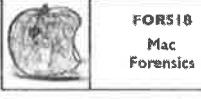
CORE

	FOR408 Windows Forensics GCFE		SEC504 Hacker Techniques, Exploits, and Incident Handling GCIH
--	---	---	---

IN-DEPTH INCIDENT RESPONSE

	FOR508 Advanced Incident Response GCFA		FOR572 Advanced Network Forensics and Analysis GNFA
	FOR610 REM: Malware Analysis GREM		

SPECIALIZATION

	FOR518 Mac Forensics		FOR520 Memory Forensics In-Depth
	MGT535 Incident Response Team Management		FOR585 Advanced Smartphone Forensics

This page intentionally left blank.



DIGITAL FORENSICS & INCIDENT RESPONSE

DFIR CURRICULUM

CORE

FOR408 Windows Forensics 	SEC504 Hacker Techniques, Exploits, and Incident Handling
---	---

IN-DEPTH INCIDENT RESPONSE

FOR508 Advanced Incident Response 	FOR572 Advanced Network Forensics and Analysis
--	--

Join The SANS DFIR Community



Blog: dfir.to/DFIRBlog

Twitter: @sansforensics



Facebook: [sansforensics](https://www.facebook.com/sansforensics)



Google+: [gplus.to/sansforensics](https://plus.google.com/u/0/+sansforensics)



Mailing list: dfir.to/MAIL-LIST



YouTube: [dfir.to/DFIRcast](https://www.youtube.com/DFIRcast)

SPECIALIZATION

FOR518 Mac Forensics 	FOR526 Memory Forensics In-Depth
MGT535 Incident Response Team Management 	FOR585 Advanced Smartphone Forensics



Exercise 1.0 – Exercise Setup

WELCOME to FOR518 - Mac Forensic Analysis!
Students, please start this exercise before class.

Please let your instructor know if you had issues with the
pre-class setup (Exercise 0) or this exercise (1.0).

© SANS,
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Course Agenda

Section 1 – Mac Essentials & the HFS+ File System

Section 2 – User Domain File Analysis

Section 3 – System & Local Domain File Analysis

Section 4 – Advanced Analysis Topics

Section 5 – iOS Analysis

Section 6 – Mac Forensic Challenge

© SANS.
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.



Mac Essentials & the HFS+ File System

The SANS Institute
Sarah Edwards

© SANS,
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Section 1

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

This page intentionally left blank.



COMPUTER FORENSICS
and INCIDENT RESPONSE



Section 1 - Part 1 Mac Fundamentals

© SANS
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Apple & Mac History

- Apple Computer was established on April 1, 1976 in Cupertino, CA by:
 - Steve Jobs
 - Steve Wozniak
 - Ronald Wayne
- Apple I (\$666.66)
- First Macintosh released in 1984 (\$2,500)
- Mac OS X released in 2001



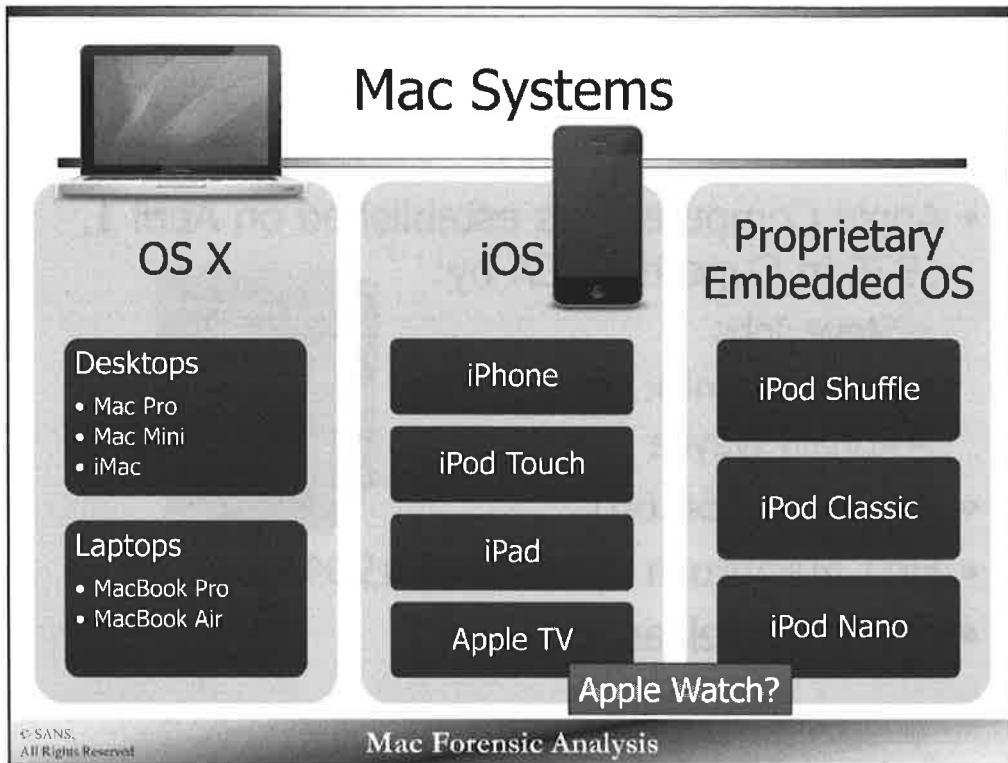
© SANS.
All Rights Reserved

Mac Forensic Analysis

Apple Computer was established on April 1, 1976 (incorporated January 3, 1977) by Steve Jobs, Steve Wozniak and Ronald Wayne.

The first computer they sold was the Apple I for \$666.66 in 1976. In 1984 the first Macintosh was introduced to the Super Bowl audience with the famous “1984” commercial. The Macintosh was touted as the first commercially successful personal computer with a graphical user interface. It was also \$2,500!

Macintosh computers use the Mac Operating System (Mac OS). The older Macs used the “Classic” Mac OS (System 1-9) while the newer Macs use Mac OS X (Ten). Mac OS X introduced a Unix-based operating system.

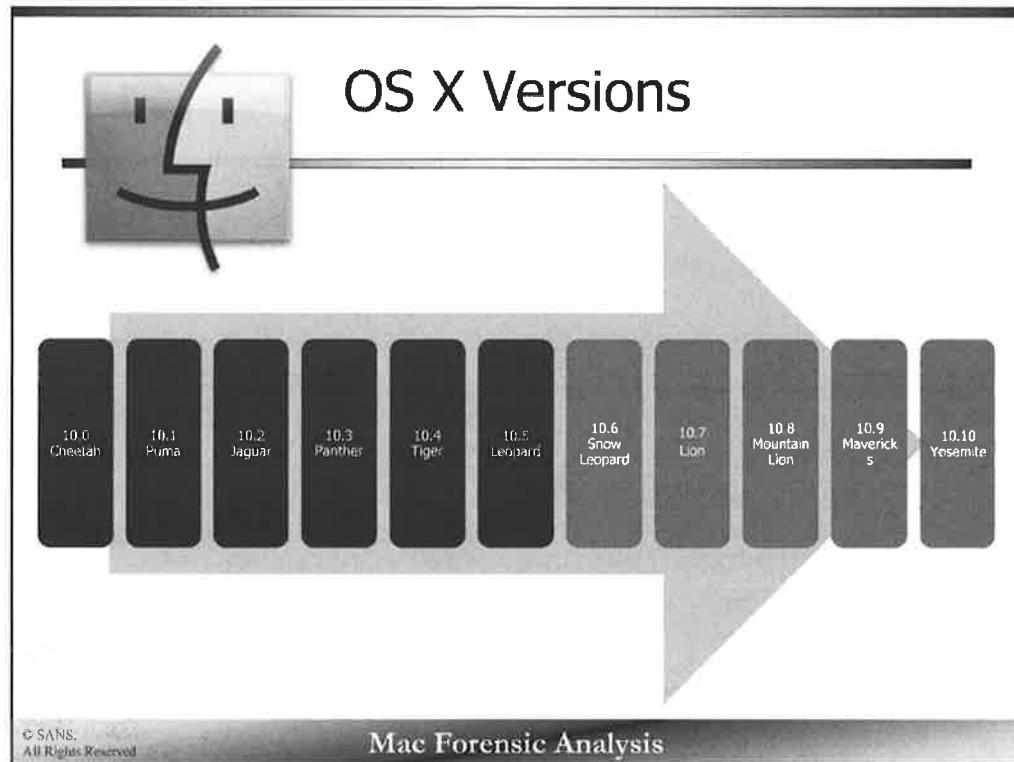


Modern Mac Systems are comprised of three types: OS X, iOS and a proprietary embedded operating system.

OS X is featured on the desktop and laptop product lines.

iOS is featured on mobile devices such as the iPhone, iPad and the Apple TV. iOS is based upon OS X, and could be considered OS X “lite”. It has many of the same characteristics and file system intricacies but does not contain all the software and features of OS X.

The iPod Shuffle, Nano, and Classic contain a proprietary embedded operating system.



There have been eleven versions of OS X, 10.0 – 10.10. The latest, Yosemite was released in October of 2014.

OS X introduced the Unix-based operating system with an easy-to-use GUI for the casual user.

This class will feature some items since 10.6, unless otherwise noted. Each version of the operating system changes slightly and can differ in various ways. Where appropriate the version of the operating system will be listed, but be aware that changes may vary and testing the various operating systems may help you with specific questions.

CPU Architecture

PowerPC

- 1994 - 2006
- 2011 - Support discontinued with 10.6
- Open Firmware

Intel x86

- 2006 - Hardware Transition
- 2009 - Software (Intel-only) with 10.6
- Extensible Firmware Interface (EFI)

© SANS.
All Rights Reserved

Mac Forensic Analysis

After the introduction of Unix core components to the operating system, Apple Computer made the transition from PowerPC to Intel x86 architecture.

PowerPC (Performance Optimization With Enhanced RISC – Performance Computing) was used from 1994 to 2006. PowerPC was no longer supported with OS X Snow Leopard (10.6). This architecture used the Open Firmware interface, similar to a PC's BIOS. The last version of OS X that was PowerPC-only was OS X Panther (10.3). This interface was accessible using the CMD+Option+O+F.

The Intel x86 transition started in 2006. OS X Tiger (10.4) was the first available OS X version that one could use on x86 architecture. Intel-based Macs use the Extensible Firmware Interface (EFI), also similar to a PCs BIOS.

Open Firmware and EFI interface between the hardware and the operating system to provide drivers and a pre-OS environment. EFI has additional advantages such as the ability to boot from USB, and larger disks.

Mac Analysis in a Windows World



Acquisition Tools

Analysis Tools

File Systems

Native/Proprietary Software

Hardware

File Formats

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Windows platform is by far the most popular analysis platform in the digital forensics community.

Many of the tools used for acquisition are the same as those used in acquiring Windows systems. A hard drive is a hard drive. Live acquisitions, however, pose a different challenge, the same techniques can be used but tools specific to Mac OS X will be needed to acquire memory, create an image, and to gather volatile data.

Analysis tools may also be the same as those used for Windows systems (i.e.: Encase, FTK, X-Ways). Many of the most popular forensic analysis tools can be used with varying success depending on their Hierarchical File System Plus (HFS+) file system interpretation and Mac-specific file capabilities. Not all tools understand what a binary property list file is. The Mac OS, as with many operating systems, uses different software and file formats.

The HFS+ file system can bring up certain challenges to forensic tools and analysis techniques. For example, HFS+ cannot be read or written to natively on Windows systems, therefore mounting an HFS+ drive in Windows requires additional software or the use of a Mac.

Timestamp Formats

HFS+ Dates or Mac OS	<ul style="list-style-type: none">• 32bit• 1/1/1904 00:00:00 UTC• Example: 3487322096 (0xCFDC4FF0)
Unix Epoch	<ul style="list-style-type: none">• 32bit• 1/1/1970 00:00:00 UTC• Example: 1404477296 (0x53B69F70)
Mac Epoch or Mac Absolute or Cocoa or Webkit	<ul style="list-style-type: none">• 32bit• 1/1/2001 00:00:00 UTC• Example: 426170096 (0x1966D6F0)

© SANS,
All Rights Reserved

Mac Forensic Analysis

You will encounter a variety of timestamps on a Mac system. While Unix epoch is relatively well known, the number of seconds from midnight on 1/1/1970, the HFS+ file system uses a different epoch date, midnight on 1/1/1904. Another epoch time, sometimes called Mac Absolute Time starts at midnight on 1/1/2001.

Throughout the class the various timestamps will be highlighted.

Convert timestamps using `date -ur <epochtime>`:

- Mac Epoch – add 978307200
- HFS+ Date – subtract 2082844800

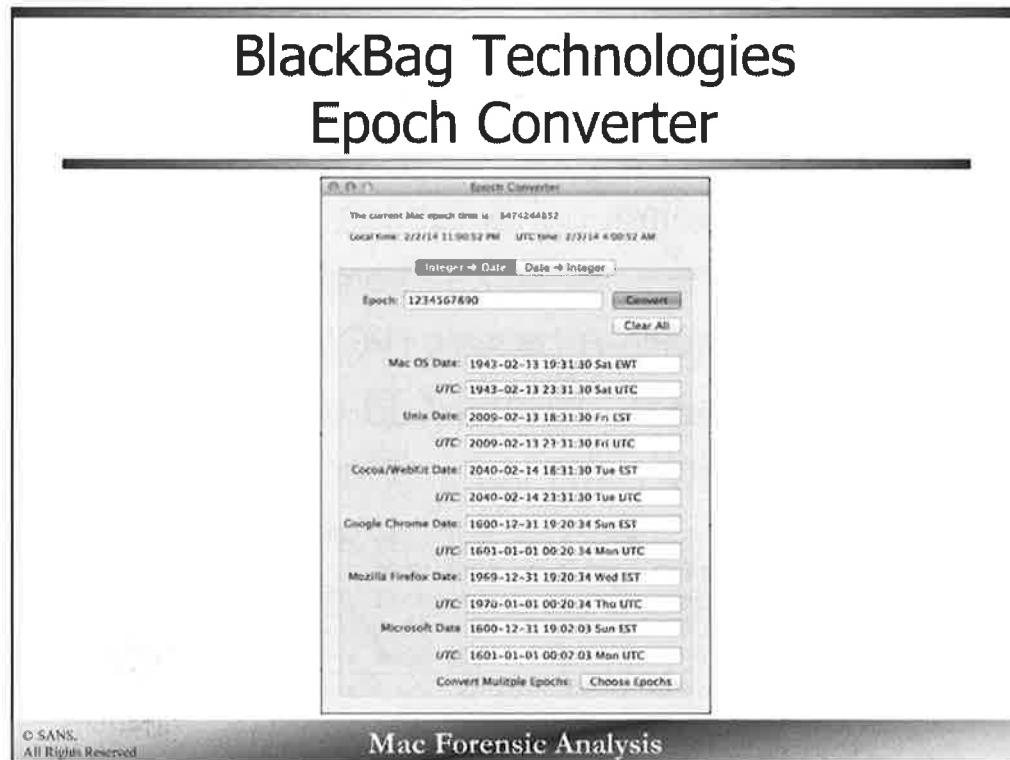
For example:

```
$ date -ur 1234567890
Fri Feb 13 23:31:30 UTC 2009
```

Reference:

Mac Developer Library – NSDate Class Reference

[https://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/nsdate_Class/Reference/Reference.html]



BlackBag Technologies has a wonderful GUI-based timestamp converter. It has the ability to convert from timestamp integer to date and vice versa.

This utility supports many timestamp formats in local system time and UTC:

- Mac OS (HFS+)
- Unix Epoch
- Cocoa/WebKit
- Google Chrome
- Mozilla Firefox
- Microsoft

This application is similar to Dcode by Digital Detective for Windows-based systems.

SQLite Databases

- SQL-based relational database
- File Signature – “SQLite format 3”
- SQLite Database Browsers (PC/Mac)
- SQLite Manager - Firefox Add-on (PC/Mac)
- Command Line: sqlite3 (Mac)

```
00000000: 5351 4c69 7465 2066 6f72 6d61 7420 3300  SQLite format 3.  
00000010: 1000 0101 0040 2020 0000 086f 0080 00b2  .....@ ...o....  
00000020: 0000 0000 0000 0000 0000 0002 0000 0001  .....  
00000030: 0000 0000 0000 0000 0001 0000 0000 0000  .....  
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  .....
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

Software on Mac systems use SQLite databases to store various types of information such as settings and Internet histories. SQLite is a SQL-based relational database. These databases can have multiple tables with multiple types of keys and values.

In the screenshot above, the signature used for SQLite databases is “SQLite format 3”. An easy way to carve for these files would be to use that signature as a starting point.

SQLite files are non-system specific and can be viewed using different tools for Mac, Windows or Linux systems. You may want to use SQLite Database Browser. This application was recently updated and now handles newer SQLite files that it was not able to do before.

Another utility to use the Firefox Add-On “SQLite Manager”. This add-on can be used with any system that Firefox can be installed on. [<https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/>]

SQLite Database

Firefox SQLite Manager Plugin

The screenshot shows the Firefox SQLite Manager Plugin interface. On the left, a tree view lists the database structure: Master Table (1), Tables (5), Views (0), Indexes (6), and Triggers (0). The 'Tables' section is expanded, showing five tables: cfurl_cache_blob_data, cfurl_cache_receiver_data, cfurl_cache_response, cfurl_cache_schema_version, and sqlite_sequence. The 'cfurl_cache_response' table is selected and its data is displayed in a grid on the right. The grid has columns: entry_ID, version, hash_value, storage_policy, request_key, time_stamp, and partition. The data consists of 14 rows, with row 8012 currently selected.

cfurl_cache_response						
entry_ID	version	hash_value	storage_policy	request_key	time_stamp	partition
8001	0	-76386465...	0	http://secure-u...	2014-05-29 01...	linkedin.com
8002	0	602016332...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8003	0	514109032...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8004	0	300130991...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8005	0	-34902579...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8006	0	236992091...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8007	0	-827754063	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8008	0	847787110...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8009	0	-11653670...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8010	0	715792937...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8011	0	347864710...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8012	0	-9158089...	0	http://m.c.lnkd...	2014-05-29 01...	linkedin.com
8013	0	429496940...	0	http://ad.double...	2014-05-29 01...	linkedin.com
8014	0	-42793857...	0	http://www.goo...	2014-05-29 01...	linkedin.com

Mac Forensic Analysis

This is an example of the Safari Cache .db SQLite database. The screenshot shown is using the Firefox SQLite Manager Plugin.

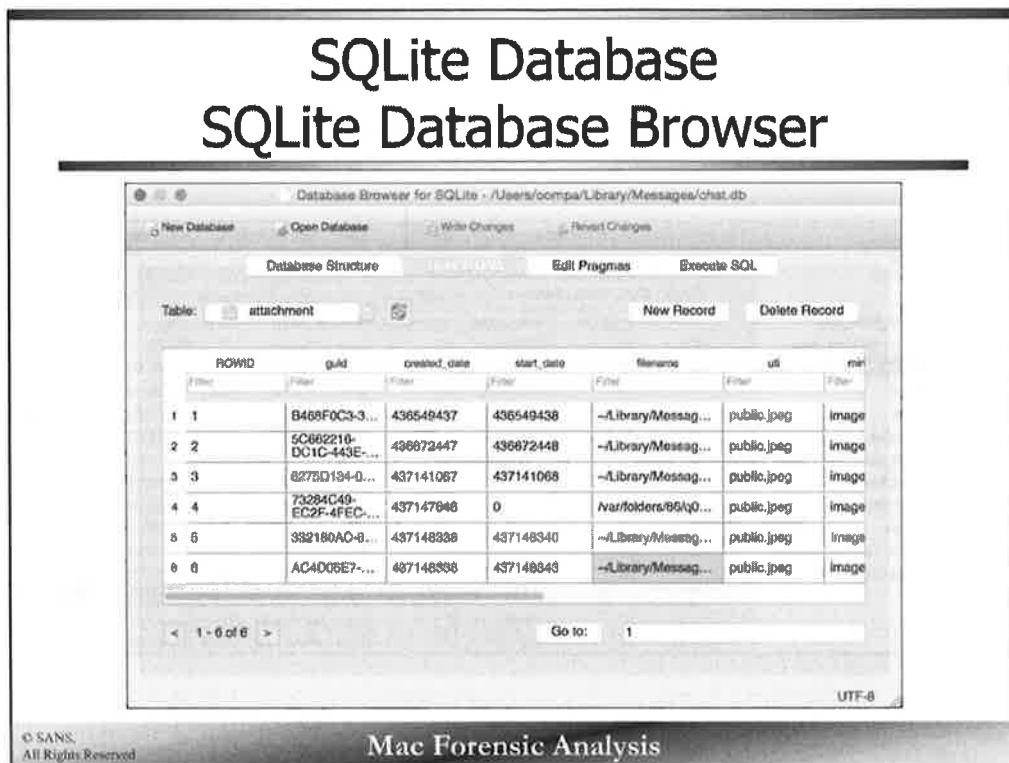
On the left under “Tables” there are five tables shown:

- cfurl_cache_blob_data
- cfurl_cache_receiver_data
- cfurl_cache_response
- cfurl_cache_schema_version
- sqlite_sequence

Each table is made up of row (or tuples) and columns.

In the example shown, this table has seven columns:

- entry_ID
- version
- hash_value
- storage_policy
- request_key
- time_stamp
- partition



This screenshot shows an example of SQLite Database Browser.

This is a standalone application that does not require a browser. Each analyst may have their own favorite SQLite database viewer, please choose what you prefer to use.

Property List Files

PLIST

- XML - File Signature – "<?xml version="
- Binary - File Signature – "bplist00"
- Open on Mac – Xcode, plutil
- Open on PC – iBackupBot Plist Editor, Reaper

```

00000000: 3c3f 786d 6c20 7665 7273 696f 6e3d 2231 <?xml version="1
00000010: 2e30 2220 656e 636f 6469 6e67 3d22 5534 .0" encoding="UT
00000020: 462d 3822 3f3e 0a3c 2144 4f43 5459 5045 F-8"?>.<!DOCTYPE
00000030: 2070 6c69 7374 2050 5542 4c49 4320 222d plist PUBLIC "-

```

```

00000000: 6270 6c69 7374 3030 d501 0203 0405 0601 bplist00.....
00000010: 0c0d 0c5f 1010 546f 6767 6c65 436f 5e74 ...._.ToggleCont
00000020: 726f 6c4b 6579 5f10 1c44 6f6e 7451 7569 rolKey_.DontQui
00000030: 7457 6865 6e4c 6173 7457 696e 646f 7743 tWhenLastWindowC

```

© SANS.
All Rights Reserved

Mac Forensic Analysis

Property list files usually contain configuration data, similar to the Windows registry. These files come in two formats, XML or binary. The older format, XML is human-readable and has the file signature of "<?xml version=". The newer binary format is used as it is more efficient on disk space. This format has the file signature "bplist00" for binary plist.

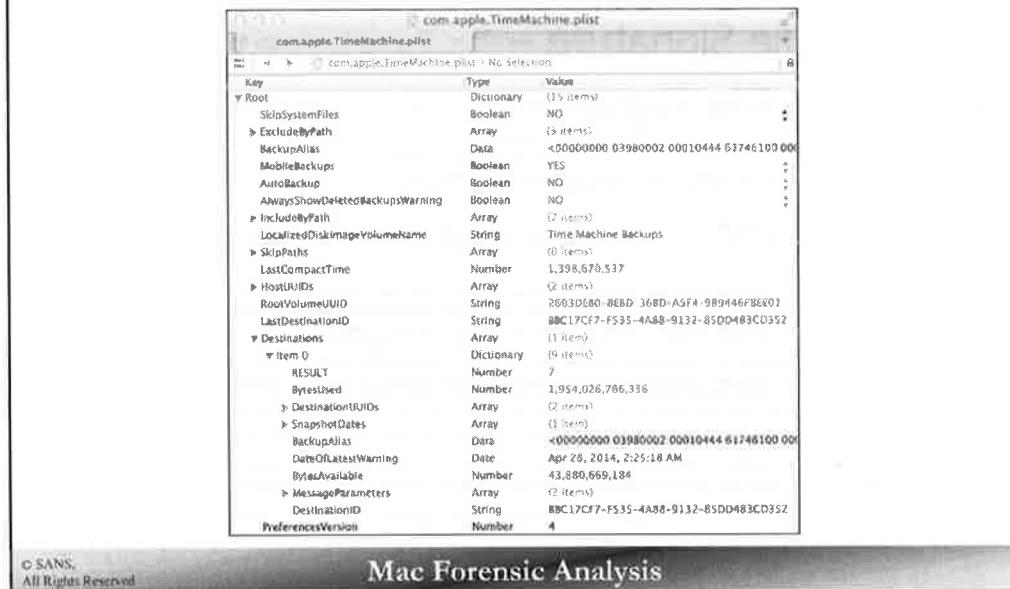
These files can be read using the Xcode or by the command-line tool plutil.

Xcode is not installed by default on Mac OS X, it can be installed by downloading it from Apple's website or by using the Mac App Store. Older versions of Xcode had a utility named "Property List Editor" that reads property list files without the extraneous bulk of Xcode. Newer versions of Xcode include a property list editor; however, it may take a while to load the whole Xcode programming environment.

A command line tool, plutil, can be used to convert property list files into other formats such as binary to XML, or XML to JSON.

On windows systems, there is one tool that the author has found to parse the property list files, it doesn't work nearly as well as the Mac tools but it works in a pinch. iBackupBot Plist Editor – [http://www.icopybot.com/download.htm] or Reaper (http://insidethecore.com/reaper/reaper.html)

Property List Files Xcode Example



© SANS.
All Rights Reserved

Mac Forensic Analysis

An example of a property list is shown in the screenshot above. This property list is being viewed with the internal Xcode plist viewer.

Each property list contains keys and values. Each value can have different data types such as:

- Boolean - On/Off, Yes/No, 0/1
- Array - Contains additional keys/values
- Data - Binary Data Blob, can be extracted and viewed in a hex editor
 - May contain embedded property lists
- Number
- String
- Date – Dates are shown in local host system time (make sure to check your time zones!)
- Dictionary – Contains additional keys/values (similar to Array)

Property List Files plutil -p Example

```
word:Preferences sledwards$ plutil -p com.apple.TimeMachine.plist | less
{
    "SkipSystemFiles" => 0
    "ExcludeByPath" => [
        0 => "//Users/Shared/adi"
        1 => "/Users/Shared/adi"
        2 => "/Library/Application Support/Microsoft/PlayReady"
    ]
    "BackupAlias" => <00000000 03950002 00010444 61746100 00000000 00000000
00000000 00000000 00000000 0002caae 2d3e0000 00000000 0000f
0061000f 00000004 00440061 00740061 00120000 0013000f 2f566f6c 756d6573 2
0000 000000 000000844 656c6f72 65616e00 00000000 00000000 000000
00000056f 6f6d7061 00000000 00000000 00000000 00000000 00000000 00
00 000000 000000 000000 000000 000000 000000 000000 000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
"MobileBackups" => 0
"AutoBackup" => 0
"AlwaysShowDeletedBackupsWarning" => 0
"IncludeByPath" => [
    0 => "/Applications"
    1 => "/Library"
    2 => "/System"
    3 => "/bin"
    4 => "/private"
    5 => "/sbin"
    6 => "/usr"
]
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

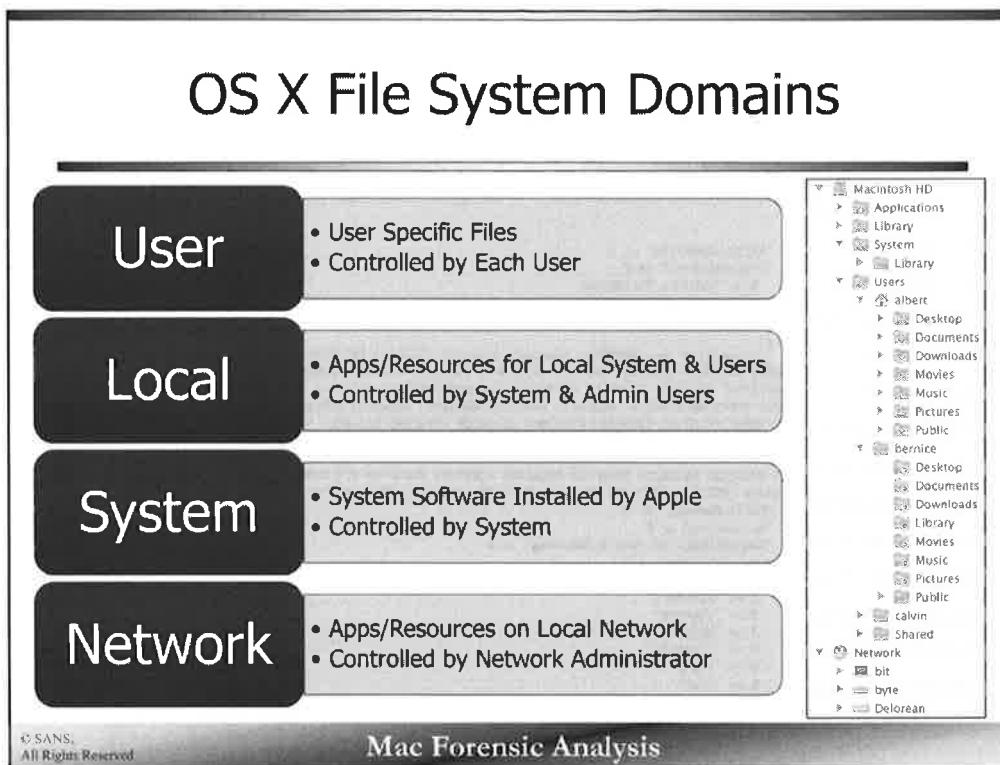
You can also view property lists on the command line using the OS X native tool, plutil. Use the **-p** parameter to “print” the property list. The default print view is JSON formatted.

You can also use plutil to convert property lists from binary to XML.

Reference:

[plutil Man Page](#)

OS X File System Domains



The Mac OS X file system is made up of four domains; user, local, system, and network.

Each domain has a purpose, they may contain user files, system resources, or network data. The domains are used to separate files based on their usage. The breakdown of the domains helps implement access controls so a file is cannot intentionally or unintentionally modified to keep the system secure and functional.

In the following slides, each domain section will be highlighted to show where in the logical file system the files are located.

Reference:

File System Programming Guide – File System Basics

[<https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html>]

- Each Default User Directory contains:
 - /Desktop
 - /Documents
 - /Downloads
 - /Library (hidden in 10.7+)
 - User Library
 - /Movies
 - /Music
 - /Pictures
 - /Public

The User Domain consists of all the user files. Their documents, pictures, music, etc.

Each user has their own home directory in the /Users/ folder, marked by a house icon in the screenshot above. If a user is logged on, default permissions will not allow them to view the files in another user's directory noted by the red circle with the line icon.

Most of the directories are self explanatory in terms of information they may contain. The public folder may contain data the user wants to share with other users. The user's Library directory contains app-specific data.

This domain may also contain "Sites" directory if web sharing is enabled; this would contain the user's personal web site.

Starting with Mac OS X Lion, the user's Library folder was hidden. Some users may prefer to be able to view the files easily. The author finds that it helps forensic research to have the Library directory to be viewable. The command to permanently change this for a specific account is below. It may also be accessed by holding down the "option" key when accessing it via the 'Go' menu in the Finder Toolbar.

```
chflags nohidden /Users/<username>/Library
```

Reference:

File System Programming Guide – File System Basics

[<https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html>]

Local Domain

- /Applications
 - Contains applications for all users
- /Library
 - Local Library - Contains application specific data
- /Developer
 - ...or /Library/Developer
 - ...or /Applications/Xcode.app/...

Mac Forensic Analysis

© SANS. All Rights Reserved.

The local domain consists of the applications folder, the (local) Library directory, and if installed, the Developer directory. This domain is used to store the files that may be shared amongst the users, such as applications.

The /Applications directory contains the applications available to all users of the system. It is not necessary to run an application from the /Applications directory, it may be run from a user directory if needed. Applications from the Mac App Store are installed in the /Applications directory.

The /Developer directory is created when Xcode is installed. The Developer directory may be located in either the root (/), under /Library or embedded in the Xcode.app application in the /Applications directory. The Developer folder contains Apple Developer (Mac, iOS,) related data and resources.

Reference:

File System Programming Guide – File System Basics

[<https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html>]

The slide has a decorative vertical bar on the left side.

System Domain

- /System
 - Apple System Resources
- /System/Library
 - System Library

© SANS.
All Rights Reserved

Mac Forensic Analysis

The file browser window shows the following directory structure:

```

Macintosh HD
  Applications
  Library
    System
      Library
    Users
      albert
        Desktop
        Documents
        Downloads
        Movies
        Music
        Pictures
        Public
      bernice
        Desktop
        Documents
        Downloads
        Library
        Movies
        Music
        Pictures
        Public
      calvin
        Shared
    Network
      air
      brie
      Delorean
  
```

The System Domain is used to store Apple specific system software.

The System Domain contains the System Library directory which contains files associated with Apple system resources.

The primary files we will be looking at in this domain are system preferences and data files.

You may have noticed a common theme in the directory structure. There are three Library directories on Mac OS X, each with its own purpose. It is easy to get confused about which Library directory contains what data, this class will discuss the difference between each Library directory and the contents located within.

- User Library - /Users/<username>/Library/
- Local Library - /Library/
- System Library – /System/Library/

Reference:

File System Programming Guide – File System Basics

[<https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html>]

File System Program Guide – OS X Library Directory Details

[https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/MacOSXDirectories/MacOSXDirectories.html##apple_ref/doc/uid/TP40010672-CH10-SW1]

The screenshot shows a Mac OS X desktop environment. In the center is a Finder window with the title bar "Network Domain". The left pane contains a list of network resources:

- Macintosh HD
- Applications
- Library
- System
- Library
- Users
 - albert
 - Desktop
 - Documents
 - Downloads
 - Movies
 - Music
 - Pictures
 - Public
 - bernice
 - Desktop
 - Documents
 - Downloads
 - Library
 - Movies
 - Music
 - Pictures
 - Public
 - calvin
 - Shared
- Network
 - bit
 - byte
 - DeLorean

© SANS.
All Rights Reserved.

Mac Forensic Analysis

The Network Domain contains the network resources such as network area storage, Time Capsules, and other systems on the network.

Reference:

File System Programming Guide – File System Basics

[<https://developer.apple.com/library/mac/DOCUMENTATION/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html>]

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

This page intentionally left blank.



Section 1 – Part 2

Acquisition & Live Response

© SANS,
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Dead vs. Live Acquisition & Considerations

Desktop & Laptop Hard Drives

Flash SSD Hard Drives

Encryption

Target Disk Mode

Disk Arbitration

Boot Camp

Tools



© SANS.
All Rights Reserved

Mac Forensic Analysis

Many of the same tools used for other **systems** can be used to acquire a Mac. If necessary and physically possible, the hard drives can be removed for **acquisition**, the author recommends using the website <http://www.ifixit.com/> for hard drive removal instructions. Some **Macs** are known to be harder to extract the hard drive from than others.

The MacBook Air has introduced **solid-state/flash** hard drives to a wide variety of users. Most models do not use the “regular” 1.8” or 2.5” Solid-state Drive (**SSD**) hard drive; rather they use a bank of flash chips on a board with a non-standard interface. As of this writing, **Old** World Computing makes an external enclosure for the MacBook Air SSD drives that will allow use of standard **interfaces**. Search for “OWC Mercury On-The-Go USB External Enclosure Solution for MacBook Air” (<http://eshop.macsales.com/item/OWC/SSDAPOTGU3/>)

Encryption can make **dead-drive acquisition** difficult with Legacy FileVault and FileVault 2 encryption. Legacy FileVault encrypts the user's home **directory**, while FileVault2 implements full disk encryption. This does not make acquisition impossible, you may image the **drive**, but you better hope you can get the password! If a system is using FileVault and is up and running – it is best **to** image unencrypted while you can.

Target Disk Mode (TDM) is used to **acquire** a Mac hard drive (primary drive only) while still in its original machine. This allows the computer to be seen by an **analysis** system as an external FireWire/Thunderbolt drive. Some systems, including older MacBook Airs do not have **FireWire** or Thunderbolt ports, therefore TDM is not possible. When a hard drive has a Boot Camp partition, you **will** see two volumes – one OS X the other for Windows. If a hard drive has been encrypted using FileVault2, this **volume** will need to be unlocked. This is described in detail in Section 4. Booting the target system with the “T” key **held** down can access this mode.

It is worth noting that older systems may **have** an Open Firmware password may be set, there are a few methods can be used (with varying levels of **succes**s):

- Change the amount of **RAM** **in** the system and reboot.
- Press Command+Option+P+R to clear the PRAM.

You may enter the Open Firmware terminal by using Command+Option+O+F key sequence.

On newer systems, if the EFI Firmware password is set you will need to obtain a specific key hash and send to Apple (subpeonas@apple.com). Instructions can be found here: <http://www.cnet.com/news/efi-firmware-protection-locks-down-newer-macs/>.

Disk arbitration is the process that automatically mounts disks. When acquiring disks using a Mac be aware if Disk Arbitration is enabled or not, you may automatically mount an evidential disk (which makes writes to it). This can be turned on/off using the launchctl commands (these commands are not persistent after a reboot):

Enable (Enabled by default):

```
sudo launchctl load  
/System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist
```

Disable:

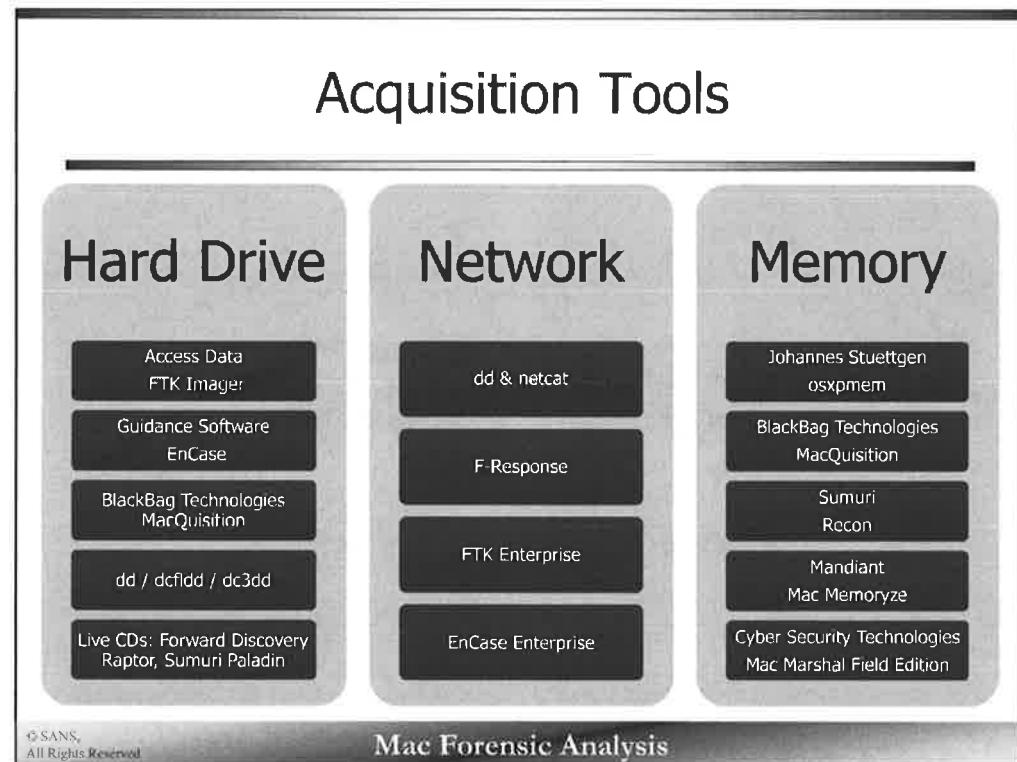
```
sudo launchctl unload  
/System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist
```

To determine if disk arbitration is running run the following command to search for the diskarbitrationd process (should be under the root user context):

```
ps auxw | grep diskarbitrationd
```

Boot Camp is used to create two bootable partitions, one for OS X and one for Windows. While in Target Disk Mode, if the Mac has a Boot Camp partition (a Windows installation on a separate partition) it should be noted that this may be automatically mounted on a Windows acquisition system. It is also possible to triple-boot (Linux!) Mac systems using third-party tools, the most notable is rEFIt (<http://refit.sourceforge.net/>).

SSD picture from iFixit.com [www.ifixit.com/Teardown/MacBook+Air+11-Inch+Late+2010+Teardown].



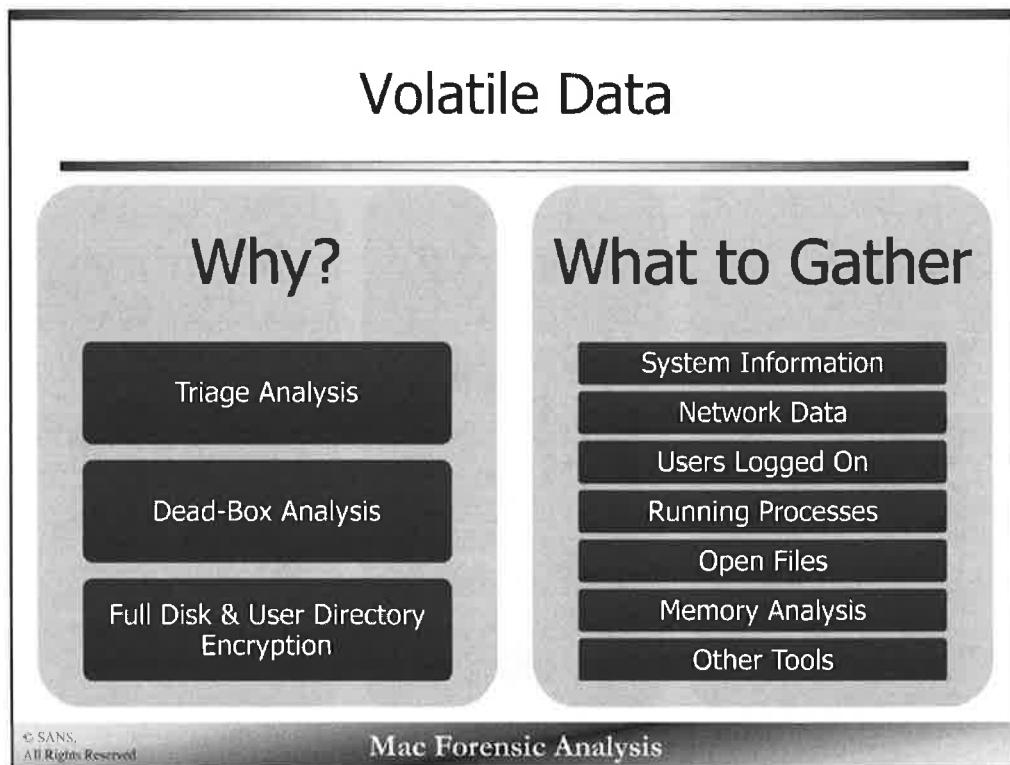
Many of the same acquisition tools can be used on hard drives. Hard drives are hard drives – the same tools you use for Windows-based imaging can be used for Mac systems.

Live acquisition should be used if you need to acquire volatile data or a drive that has implemented encryption. Many of the same tools used for dead-drive acquisition have live response functions as well.

It may be necessary to acquire a system over the network; whether for enterprise or simple distance issues. There are tools that range from a simple over-the-network implementation of dd and netcat or full-blown enterprise suites.

Memory acquisition may be needed for investigations that are highly dependent on volatile data such as network investigations.

The tools listed are not the only tools available, there are many options at various price points on the market, it is up to you and/or your organization to find the right product for your needs.



Volatile data may be necessary to gather in many types of investigations.

A quick triage may be needed to verify certain data points of a system such as the IP address, a running process, or if a specific file is open.

A dead-box analysis may be incomplete without volatile data, the investigation may need information such as running processes or network data to get a full picture of the system.

If a system is completely inaccessible due to encryption, gathering volatile data may be able to make these types of cases successful. With any luck, a memory image may contain a plaintext password!

Collection of volatile information may help an investigator:

- Decrypt an encrypted volume
- Pin-point a connection to a malicious IP address or a server containing lewd material
- Determine a malicious running process
- Determine what files and documents were open at the time of acquisition
- Determine who was logged onto the system at the time of acquisition

Memory Acquisition

Johannes Stuettgen
osxpmem *10.9+

BlackBag Technologies
MacQuisition *10.9+

Sumuri
Recon *10.9+

Mandiant
Mac Memoryze

Cyber Security Technologies
Mac Marshal Field Edition

© SANS,
All Rights Reserved

Mac Forensic Analysis

We will go more in-depth into Mac memory acquisition and analysis in Section 4. The tools listed above are some of the recommended tools to acquire Mac memory.

Each tool has their own limitations and benefits, it is up to the user to determine which tool is best for their current situation. For example, MacQuisition and Recon are part of an applications that acquires volatile data, hard drives, and memory. Mac Marshal is a Mac triage/analysis utility that is available for purchase. Mac Memoryze and osxpmem are free and/or open source but do not acquire additional volatile data or hard drive images.

Note: Each tool requires administrative system privileges to run.

*10.9 changes things up a bit, introducing compress memory. As of this writing OSXPMem, MacQuisition, and Recon are the only tools capable of acquiring memory from systems running 10.9+.

Live Response

System Information

date hostname uname -a sw_vers

```
bit:~ compa$ date
Sun Aug 12 18:13:26 EDT 2012
bit:~ compa$ hostname
bit
bit:~ compa$ uname -a
Darwin bit 12.0.0 Darwin Kernel Version 12.0.0: Sun Jun 24 23:00:16 PDT 2012; ro
ot:xnu-2050.7.9~1RELEASE_X86_64 x86_64
bit:~ compa$ sw_vers
ProductName: Mac OS X
ProductVersion: 10.8
BuildVersion: 12A269
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

These four commands will give you the basic information about the system.

date – The local time of the system.

hostname – The hostname of the system.

uname -a – Prints the OS name, network node name, system architecture/release/name/version information.
(the -a parameter displays all the information available for this command)

sw_vers – Prints the OS X version and build information.

Many similar commands in other systems do not necessarily work by default on the Mac. The Mac may use a completely different command to perform the same action, or it may just implement different option flags. Getting to know the commands and using man pages will help you understand the command line intricacies of the Mac.

More information about mapping the Darwin Kernel versions can be found here:
[http://en.wikipedia.org/wiki/Darwin_\(operating_system\)](http://en.wikipedia.org/wiki/Darwin_(operating_system))

```
bit:~ oompa$ date
Sun Aug 12 18:13:26 EDT 2012
bit:~ oompa$ hostname
bit
bit:~ oompa$ uname -a
Darwin bit 12.0.0 Darwin Kernel Version 12.0.0: Sun Jun 24 23:00:16 PDT 2012; ro
ot:xnu-2050.7.9~1RELEASE_X86_64 x86_64
bit:~ oompa$ sw_vers
ProductName: Mac OS X
ProductVersion: 10.8
BuildVersion: 12A269
```

Live Response

Active Network Connections

```
netstat -anf inet
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp4	0	0	192.168.1.206.61743	74.125.228.71.443	ESTABLISHED
tcp4	0	0	192.168.1.206.61742	74.125.228.37.443	ESTABLISHED
tcp4	37	0	192.168.1.206.61735	107.20.249.221.443	CLOSE_WAIT
tcp4	37	0	192.168.1.206.61734	107.20.249.221.443	CLOSE_WAIT
tcp4	37	0	192.168.1.206.61733	108.160.166.10.443	CLOSE_WAIT
tcp4	37	0	192.168.1.206.61732	108.160.165.253.443	CLOSE_WAIT
tcp4	0	0	192.168.1.206.61720	8.18.203.20.80	ESTABLISHED
tcp4	0	0	192.168.1.206.61719	8.18.203.20.80	ESTABLISHED
tcp4	0	0	192.168.1.206.61718	8.18.203.20.80	ESTABLISHED
tcp4	0	0	192.168.1.206.61717	8.18.203.20.80	ESTABLISHED
tcp4	0	0	192.168.1.206.61651	174.129.31.99.443	ESTABLISHED
tcp4	0	0	192.168.1.206.61650	174.129.31.99.443	ESTABLISHED

© SANS.
All Rights Reserved

Mac Forensic Analysis

Was the system connected to a lewd server or a malicious IP address? The netstat -an command lists active network connections, these can be used to determine what connections the system has.

The -a parameter shows the stat of all network sockets.

The -n parameter shows network addresses as numbers (rather than host and domain names).

The -f parameter limits the amount of data shown to only IPv4 and IPv6 network sockets, this does not include UNIX sockets.

The output fields:

- Protocol (TCP or UDP)
- Received Queue (bytes)
- Send Queue (bytes)
- Local IP Address and Port
- Foreign (Remote) IP Address and Port
- State of the Connection

Active Internet connections (including servers)					
	Proto	Recv-Q	Send-Q	Local Address	Foreign Address (state)
nibble:vm sledwards\$	tcp4	0	0	192.168.1.206.61743	74.125.228.71.443 ESTABLISHED
	tcp4	0	0	192.168.1.206.61742	74.125.228.37.443 ESTABLISHED
	tcp4	37	0	192.168.1.206.61735	107.20.249.221.443 CLOSE_WAIT
	tcp4	37	0	192.168.1.206.61734	107.20.249.221.443 CLOSE_WAIT
	tcp4	37	0	192.168.1.206.61733	108.160.166.10.443 CLOSE_WAIT
	tcp4	37	0	192.168.1.206.61732	108.160.165.253.443 CLOSE_WAIT
	tcp4	0	0	192.168.1.206.61720	8.18.203.20.80 ESTABLISHED
	tcp4	0	0	192.168.1.206.61719	8.18.203.20.80 ESTABLISHED
	tcp4	0	0	192.168.1.206.61718	8.18.203.20.80 ESTABLISHED
	tcp4	0	0	192.168.1.206.61717	8.18.203.20.80 ESTABLISHED
	tcp4	0	0	192.168.1.206.61651	174.129.31.99.443 ESTABLISHED
	tcp4	0	0	192.168.1.206.61650	174.129.31.99.443 ESTABLISHED

Live Response

Active Network Connections by Process

lsof -i

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
SystemUIS	236	oompa	8u	IPv4	0x9ed2b1a706515075	0t0	UDP	*:*
NetworkBr	241	oompa	5u	IPv4	0x9ed2b1a7065179cd	0t0	UDP	*:*
Imagent	258	oompa	12u	IPv4	0x9ed2b1a706763535	0t0	UDP	localhost:58160->localhost:58160
Imagent	258	oompa	13u	IPv4	0x9ed2b1a71bc9bc65	0t0	TCP	bit:61027->bos->0005b-rdr4.blue.aol.com:https (ESTABLISHED)
Imagent	258	oompa	16u	IPv4	0x9ed2b1a706eeb84d	0t0	TCP	bit:61011->qc-in-f125.1e100.net:5223 (ESTABLISHED)
Dropbox	279	oompa	18u	IPv4	0x9ed2b1a71bcab0c65	0t0	TCP	bit:60999->jc-not18.sjc.dropbox.com:http (ESTABLISHED)
Dropbox	279	oompa	18u	IPv4	0x9ed2b1a7065151fd	0t0	UDP	*:17500
Dropbox	279	oompa	19u	IPv4	0x9ed2b1a70cace20d	0t0	TCP	*:17500 (LISTEN)
Dropbox	279	oompa	21u	IPv4	0x9ed2b1a710367df5	0t0	TCP	bit:61139->v-client-1a.sjc.dropbox.com:https (CLOSE_WAIT)
Dropbox	279	oompa	25u	IPv4	0x9ed2b1a709419c65	0t0	TCP	localhost:26164 (LISTEN)
Dropbox	279	oompa	28u	IPv4	0x9ed2b1a710259ad5	0t0	TCP	bit:61035->ec2-107-22-245-91.compute-1.amazonaws.com:https (CLOSE_WAIT)
Dropbox	279	oompa	29u	IPv4	0x9ed2b1a706eebf85	0t0	TCP	bit:61045->v-client-2b.sjc.dropbox.com:https (CLOSE_WAIT)
Mail	824	oompa	34u	IPv4	0x9ed2b1a70fed8df5	0t0	TCP	bit:61017->mail.csh.rit.edu:imaps (ESTABLISHED)

© SANS.
All Rights Reserved

Mac Forensic Analysis

The `lsof -i` command lists Internet and network connections by process. We can see in the screenshot the Dropbox and Mail applications had active connections.

The output fields:

- Process Name
- Process ID
- User Account
- File Descriptor (See man page for details)
- Type (IPv4, IPv6)
- Device (See man page for details)
- Size/Offset (See man page for details)
- Node (TCP or UDP)
- Connection Information & Status (See man page for details)

Other flags that could be of use are the following:

- `-n` – Show IP addresses instead of network names.
- `-P` – Show port numbers versus names.
- `-l` – Show numeric user IDs versus account names.

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE NAME
SystemUIS	236	oompa	8u	IPv4	0x9ed2b1a706515075	0t0	UDP *;*
NetworkBr	241	oompa	5u	IPv4	0x9ed2b1a7065179cd	0t0	UDP *;*
imageant	258	oompa	12u	IPv4	0x9ed2b1a706763535	0t0	UDP localhost:58160->localhost:58160
imageant	258	oompa	13u	IPv4	0x9ed2b1a71bc9bc65	0t0	TCP bit:61027->bos-d005b-rdr4.blue.aol.com:https (ESTABLISHED)
imageant	258	oompa	16u	IPv4	0x9ed2b1a706eeb84d	0t0	TCP bit:61011->qc-in-f125.1e100.net:5223 (ESTABLISHED)
Dropbox	279	oompa	10u	IPv4	0x9ed2b1a71bca0c65	0t0	TCP bit:60999->sjc-not18.sjc.dropbox.com:http (ESTABLISHED)
Dropbox	279	oompa	16u	IPv4	0x9ed2b1a7065151fd	0t0	UDP *:17500
Dropbox	279	oompa	19u	IPv4	0x9ed2b1a70cace20d	0t0	TCP *:17500 (LISTEN)
Dropbox	279	oompa	21u	IPv4	0x9ed2b1a710367df5	0t0	TCP bit:61139->v-client-1a.sjc.dropbox.com:https (CLOSE_WAIT)
Dropbox	279	oompa	25u	IPv4	0x9ed2b1a709419c65	0t0	TCP localhost:26164 (LISTEN)
Dropbox	279	oompa	28u	IPv4	0x9ed2b1a710259ad5 (CLOSE_WAIT)	0t0	TCP bit:61035->ec2-107-22-245-91.compute-1.amazonaws.com:https
Dropbox	279	oompa	29u	IPv4	0x9ed2b1a706eebf85	0t0	TCP bit:61045->v-client-2b.sjc.dropbox.com:https (CLOSE_WAIT)
Mail	824	oompa	34u	IPv4	0x9ed2b1a70fed8df5	0t0	TCP bit:61017->aail.csh.rit.edu:imaps (ESTABLISHED)

Live Response Routing Table

netstat -rn

bit:~ compas\$ netstat -rn more Routing tables							
Internet:							
Destination	Gateway	Flags	Refs	Use	Netif	Expire	
default	192.168.1.254	UGSc	22	0	en1		
127	127.0.0.1	UCS	0	0	lo0		
127.0.0.1	127.0.0.1	UH	5	23616	lo0		
169.254	link#6	UCS	1	0	en1		
169.254.204.125	b8:c7:5d:cc:5:80	UHLSW	0	1	en1		
192.168.1	link#6	UCS	5	0	en1		
192.168.1.1	c0:3f:e:8c:59:59	UHLWII	1	104	en1	850	
192.168.1.101	127.0.0.1	UHS	0	0	lo0		
192.168.1.133	3c:7:54:3:65:20	UHLWII	0	4502	en1	295	
192.168.1.209	d0:23:db:72:91:10	UHLWII	0	45	en1	1163	
192.168.1.254	e0:69:95:50:4c:6	UHLWIIir	23	577	en1	1188	
192.168.1.255	ff:ff:ff:ff:ff:ff	UHLWbI	0	1	en1		

© SANS,
All Rights Reserved

Mac Forensic Analysis

Was this system connected to a specific server recently? The netstat -rn command prints the routing table. The routing table can be used to determine recent network connections or static routes.

The -r parameter shows the routing table.

The -n parameter shows network addresses as numbers (rather than host and domain names).

Note: Leading 0's in MAC addresses may not show in output.

```
bit:~ compass$ netstat -rn | more
Routing tables
```

Internet:

Destination	Gateway	Flags	Refs	Use	Netif	Expire
default	192.168.1.254	UGSc	22	0	en1	
127	127.0.0.1	UCS	0	0	lo0	
127.0.0.1	127.0.0.1	UH	5	23616	lo0	
169.254	Link#6	UCS	1	0	en1	
169.254.204.125	b8:c7:5d:cc:5:80	UHLSM	0	1	en1	
192.168.1	Link#6	UCS	5	0	en1	
192.168.1.1	c0:3f:e:8c:59:59	UHLWII	1	104	en1	850
192.168.1.101	127.0.0.1	UHS	0	0	lo0	
192.168.1.133	3c:7:54:3:65:20	UHLWII	0	4502	en1	295
192.168.1.209	d0:23:db:72:91:10	UHLWII	0	45	en1	1163
192.168.1.254	e0:69:95:50:4c:6	UHLWIIr	23	577	en1	1188
192.168.1.255	ff:ff:ff:ff:ff:ff	UHLWBI	0	1	en1	

Live Response ARP Table

```
arp -an
```

```
bit:~ oompa$ arp -an
? (169.254.204.125) at b8:c7:5d:cc:5:80 on en1 [ethernet]
? (192.168.1.1) at c0:3f:e:8c:59:59 on en1 ifscope [ethernet]
? (192.168.1.133) at 3c:7:54:3:65:20 on en1 ifscope [ethernet]
? (192.168.1.209) at d0:23:db:72:91:10 on en1 ifscope [ethernet]
? (192.168.1.254) at e0:69:95:50:4c:6 on en1 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en1 ifscope [ethernet]
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

Does the target have a NAS hiding in a closet somewhere?

The `arp -an` command prints the Address Resolution Protocol (ARP) table. ARP table shows the IP address to MAC address resolution used for systems on the local network. They may aid in identifying other systems to pursue.

The `-a` parameter shows the stat of all network sockets.

The `-n` parameter shows network addresses as numbers (rather than host and domain names).

Note: Leading 0's in MAC addresses may not show in output.

Live Response

Network Configuration

ifconfig

```
bit:~ compa$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xffffffff
            inets ::1 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=2b<RXCSUM,TXCSUM,VLAN_HWTAGGING,TS04>
    ether c4:2c:03:09:ca:fd
    media: autoselect (none)
    status: inactive
lw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 4078
    lladdr e8:06:88:ff:fe:d5:5d:08
    media: autoselect <full-duplex>
    status: inactive
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 90:27:e4:f8:e6:5f
    inet6 fe80::9227:e4ff:fe8:e65f%en1 prefixlen 64 scopeid 0x6
        inet 192.168.1.101 netmask 0xffffffff broadcast 192.168.1.255
        media: autoselect
        status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
    ether 02:27:e4:f8:e6:5f
    media: autoselect
    status: inactive
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

The `ifconfig` command prints the network configuration of the system.

In the screenshot we can see this system as multiple network interfaces. The active interface, `en1` has an IP address of 192.168.1.101. We can also see the various MAC addresses for each interface, this may be helpful if network logs were captured.

Network Interfaces:

- `lo#` - Loopback
- `gif#` - Generic Tunnel
- `stf#` - IPv6 to IPv4 Tunnel
- `en#` - Ethernet or Wireless – These are likely the most important to an investigator.
- `fw#` - FireWire
- `p2p#` - Point-to-Point

More information about the wireless interface can be done by using the `airport` command. This utility is located in a directory that is not in the default user's path therefore it should be run as follows. This command can supply the station name, its MAC address, and its authentication status. (i.e., WPA2, WEP).

/System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport -I

The `ifconfig -v` command will produce an additional output starting with "type: *" that contains the type of connection the interface is implementing (i.e.: Wi-Fi, Ethernet, etc.).

```

bit:~ oompa$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xff000000
            inet6 ::1 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=2b<RXCSUM,TXCSUM,VLAN_HWTAGGING,TSO4>
    ether c4:2c:03:09:ca:fd
    media: autoselect {none}
        status: inactive
fw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 4078
    lladdr e8:06:88:ff:fe:d5:5d:08
    media: autoselect <full-duplex>
        status: inactive
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 90:27:e4:f8:e6:5f
    inet6 fe80::9227:e4ff:fef8:e65f%en1 prefixlen 64 scopeid 0x6
        inet 192.168.1.101 netmask 0xffffffff broadcast 192.168.1.255
            media: autoselect
                status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
    ether 02:27:e4:f8:e6:5f
    media: autoselect
        status: inactive

```

Live Response

Open Files

lsof

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE NAME
logwindow	57	com.apple	0	REG	1,4	4	145
logwindow	57	com.apple	0x1	REG	1,4	1498816	5949407 /System/Library/CoreServices/Logwindow.app/Contents/MacOS/Logwindow
logwindow	57	com.apple	0x2	REG	1,4	98816	578525 /System/Library/PrivateFrameworks/BezelServices.framework/Versions/A/BezelServices
logwindow	57	com.apple	0x3	REG	1,4	65746	5789745 /System/Library/PrivateFrameworks/MachineSettings.framework/Versions/A/MachineSettings
logwindow	57	com.apple	0x4	REG	1,4	3940	5821456 /System/Library/Caches/com.apple.IntelCache.kext
logwindow	57	com.apple	0x5	REG	1,4	17269	5723163 /System/Library/PrivateFrameworks/CarbonCore.framework/CarbonCore
logwindow	57	com.apple	0x6	REG	1,4	214	5734643 /System/Library/PrivateFrameworks/CarbonCore.framework/CarbonCore
logwindow	57	com.apple	0x7	REG	1,4	36537	5736188 /System/Library/PrivateFrameworks/heimdalPlugins/heimdalpass.bundle/Contents/MacOS/heimdalpass
logwindow	57	com.apple	0x8	REG	1,4	1982212	3746813 /usr/share/libexec/ldd/ldd101.dylib

© SANS.
All Rights Reserved

Mac Forensic Analysis

The lsof command prints the open files (and network connections) by the process that is using them. We can see what data files and documents the users' processes has open per application.

It includes the process ID, username, and filename.

For example, keyloggers may have an open file handle to the keylog file they are writing keystrokes to. Using the lsof command if we are able to identify a rogue keylogger process we may be able to identify the keylog file.

bit#	comps	loop	more					
COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
loginwind	57	oempa	cwd	DIR	1,4	1155	2	/
loginwind	57	oempa	txt	REG	1,4	1490816	5949467	/System/Library/CoreServices/loginwindow.app/Contents/MacOS/loginwindow
loginwind	57	oempa	txt	REG	1,4	98816	5005825	/System/Library/PrivateFrameworks/BezelServices.framework/Versions/A/BezelServices
loginwind	57	oempa	txt	REG	1,4	65744	5709745	/System/Library/PrivateFrameworks/MachineSettings.framework/Versions/A/MachineSettings
loginwind	57	oempa	txt	REG	1,4	39404	6331456	/System/Library/Caches/Code.apple.IntelDataCache.le.kbdx
loginwind	57	oempa	txt	REG	1,4	17289	5736772	/System/Library/PrivateFrameworks/CoreMLKit.framework/Versions/A/Resources/AirPort0.pdf
loginwind	57	oempa	txt	REG	1,4	21216	5806643	/System/Library/KerberosPlugins/KerberosFrameworkPlugins/Relayaloopac.bundle/Contents/MacOS/heimdalopac
loginwind	57	oempa	txt	REG	1,4	16537	5736788	/System/Library/PrivateFrameworks/CoreMLKit.framework/Versions/A/Resources/AirportOff.pdf
loginwind	57	oempa	txt	REG	1,4	19022112	5746813	/System/share/icu/1icudt49l.dat

Live Response

Users Logged On [1]

who -a

w

```
bit:~ oompa$ who -a
reboot ~ Aug 4 11:24 00:24 1
oompa console Aug 4 11:26 old 57
oompa ttys000 Aug 12 18:13 . 13949
oompa ttys001 Aug 12 18:18 00:09 13976
.
run-level 3
bit:~ oompa$ w
18:55 up 8 days, 7:31, 3 users, load averages: 1.07 0.99 0.88
USER TTY FROM LOGIN@ IDLE WHAT
oompa console -
oompa s000 -
oompa s001 -
04Aug12 8days -
18:13 - w
18:18 9 /usr/bin/less -is
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The who -a command displays the following:

- Time of last system boot
- Logged in users
- Type of Logon
 - console - Login via GUI
 - ttys* - Login via Terminal
- User Login Time
- User Idle Time
- Associated Process ID

The w command has much of the same information as who -a, however it also contains:

- Local System Time
- System Uptime
- Number of Users
- Remote Login System Information
- Current Activity of the Logon Process

The ‘oompa’ user in the screenshot has one login process open (through the normal login screen) and two Terminal sessions open. Each Terminal session/window is a separate login process.

You may also want to identify the currently logged in user, try these commands:

- who am i – Show the currently logged in username, console number, and login process timestamp.
- whoami – Show the username of currently logged in user.
- id – Show the “User Identity” of the current user, including user ID (UID) and group (GID) affiliations.

Live Response Users Logged On [2]

last

```
nibble:/ sledwards$ last
sledwards  ttys001          Sat Feb  1 12:18  still logged in
sledwards  ttys000          Fri Jan 31 20:26  still logged in
sledwards  ttys000          Sun Jan 26 21:49 - 21:52 (00:03)
sledwards  ttys006          Wed Jan 22 18:40 - 19:59 (01:18)
sledwards  ttys005          Tue Jan 21 18:21 - 21:17 (7+02:55)
sledwards  ttys004          Tue Jan 21 18:15 - 21:17 (7+03:02)
sledwards  ttys002          Tue Jan 21 16:48 - 21:17 (7+04:29)
sledwards  ttys008          Tue Jan 21 05:31 - 12:29 (06:57)
sledwards  ttys007          Tue Jan 21 05:29 - 16:48 (11:18)
sledwards  ttys006          Tue Jan 21 05:18 - 16:48 (11:29)
sledwards  ttys005          Tue Jan 21 05:15 - 16:48 (11:33)
sledwards  ttys004          Tue Jan 21 04:47 - 16:48 (12:01)
sledwards  ttys002          Tue Jan 21 04:40 - 16:48 (12:07)
sledwards  ttys003          Mon Jan 20 18:01 - 21:42 (2+03:41)
sledwards  ttys002          Mon Jan 20 12:56 - 20:09 (07:12)
```

© SANS.
All Rights Reserved.

Mac Forensic Analysis

The `last` command lists the last logins of the system by user, terminal session, date, a hostname (if applicable), and session length.

The timestamp shown gives the duration of the login process. For example, “7+02:55” means the user has had this login process open for one week (7 days), 2 hours, and 55 minutes.

The end of the `last` command output contains the timestamp when the “`wtmp`” logging has started. For example, “`wtmp begins Mon Jul 7 10:08`”.

Live Response

Running Processes

ps aux

pompa	8814	0.3	1.2	1003732	99268	??	S	Wed06PM	1:56.18	/Applications/Microsoft Office 2011/		
kelly	14583	0.2	0.5	2525132	45848	??	S	7:09PM	0:02.11	/System/Library/PrivateFrameworks/He		
pompa	8586	0.2	0.3	2572748	22848	??	S	Tue08PM	4:49.34	/Applications/Utilities/Activity Mon		
pompa	237	0.2	1.6	3047120	137032	??	S	4Aug12	10:45.67	/System/Library/CoreServices/Finder.		
root	8589	0.1	0.0	2445112	3248	??	Ss	Tue08PM	3:36.86	/usr/libexec/activitymonitord		
pompa	254	0.1	0.2	2548272	20176	??	SN	4Aug12	0:12.17	/System/Library/CoreServices/Notifica		
pompa	279	0.1	0.0	758272	65752	??	S	4Aug12	14:27.73	/Applications/Dropbox.app/Contents/M		
charlie	14577	0.0	0.0	2484312	3532	??	SN	7:10PM	0:00.12	/usr/sbin/usernoted		
charlie	14676	0.0	0.0	2485232	3000	??	S	7:10PM	0:00.07	/System/Library/CoreServices/Network		
charlie	14673	0.0	0.2	2531632	16512	??	S	7:10PM	0:00.40	/System/Library/CoreServices/Finder.		
charlie	14672	0.0	0.6	2580296	51196	??	S	7:10PM	0:04.97	/System/Library/CoreServices/Dock.ap		
charlie	14662	0.0	0.0	2488108	2472	??	S	7:09PM	0:00.03	/System/Library/Services/AppleSpell.		
charlie	14652	0.0	0.0	2467404	2920	??	S	7:09PM	0:00.07	/System/Library/CoreServices/pbs		
charlie	14651	0.0	0.3	2599168	24472	??	S	7:09PM	0:01.43	/System/Library/CoreServices/SystemU		
charlie	14649	0.0	0.1	2581732	10120	??	S	7:09PM	0:01.18	/System/Library/Frameworks/Applicati		
charlie	14648	0.0	0.1	2504224	7332	??	S	7:09PM	0:00.11	/System/Library/CoreServices/talagent		
charlie	14642	0.0	0.0	2433976	1592	??	S	7:09PM	0:00.08	/usr/sbin/pboard		
charlie	14633	0.0	0.0	2483820	2456	??	S	7:09PM	0:00.13	/usr/sbin/distnoted agent		
charlie	14631	0.0	0.0	2465740	1500	??	S	7:09PM	0:00.25	/usr/sbin/cfprefsd agent		
charlie	14629	0.0	0.0	2471500	1576	??	Ss	7:09PM	0:00.14	/sbin/launchd		
charlie	14616	0.0	0.3	2572380	25728	??	Ss	7:09PM	0:01.50	/System/Library/CoreServices/loginwi		
Kelly	14608	0.0	0.0	2465392	2036	??	S	7:09PM	0:00.02	/System/Library/CoreServices/AirPort		
kelly	14603	0.0	0.2	2516072	14220	??	Ss	7:09PM	0:00.31	com.apple.dock.extra		

© SANS,
All Rights Reserved

Mac Forensic Analysis

The ps aux command outputs the process status. The output contains information such as (see the man page for more detail):

- User Account
- Process ID
- Percentage of CPU and RAM
- Process Start Date and Time
- Issued Command

Note the lack of the '-' in the command. Unlike other systems, putting this dash in the command will be unsuccessful. This is a good example of how OS X commands may different than other UNIX-like systems.

You may want experiment with other command arguments such as the following:

- -ww – Use this flag to display all the information, otherwise it defaults to screen width.
- -ef – Display more information including parent process IDs.

compa	8814	0.3	1.2	1003732	98268	??	S	Wed06PM	1:56.18	/Applications/Microsoft Office 2011/R
kelly	14583	0.2	0.5	2525132	45848	??	S	7:09PM	0:02.11	/System/Library/PrivateFrameworks/He
compa	8586	0.2	0.3	2572748	22840	??	S	Tue08PM	4:49.34	/Applications/Utilities/Activity Moni
compa	237	0.2	1.6	3847120	137032	??	S	4Aug12	10:45.67	/System/Library/CoreServices/Finder.c
root	8589	0.1	0.0	2445112	3248	??	Ss	Tue08PM	3:30.86	/usr/libexec/activitymonitord
compa	254	0.1	0.2	2548272	20176	??	SN	4Aug12	0:12.17	/System/Library/CoreServices/Notifica
compa	279	0.1	0.8	758272	65752	??	S	4Aug12	14:27.73	/Applications/Dropbox.app/Contents/Ma
charlie	14677	0.0	0.0	2484312	3532	??	SN	7:10PM	0:00.12	/usr/sbin/usernoticed
charlie	14676	0.0	0.0	2485232	3000	??	S	7:10PM	0:00.07	/System/Library/NetworkE
charlie	14673	0.0	0.2	2531632	16512	??	S	7:10PM	0:00.48	/System/Library/CoreServices/Finder.a
charlie	14672	0.0	0.6	2580296	51196	??	S	7:10PM	0:04.97	/System/Library/CoreServices/Dock.ap
charlie	14662	0.0	0.0	2488108	2472	??	S	7:09PM	0:00.03	/System/Library/Services/AppleSpell.s
charlie	14652	0.0	0.0	2467404	2920	??	S	7:09PM	0:00.07	/System/Library/CoreServices/pbs
charlie	14651	0.0	0.3	2599168	24472	??	S	7:09PM	0:01.43	/System/Library/CoreServices/SystemUI
charlie	14649	0.0	0.1	2501732	10120	??	S	7:09PM	0:01.18	/System/Library/Frameworks/Applicatio
charlie	14648	0.0	0.1	2504224	7332	??	S	7:09PM	0:00.11	/System/Library/CoreServices/talagent
charlie	14642	0.0	0.0	2433976	1592	??	S	7:09PM	0:00.08	/usr/sbin/pboard
charlie	14633	0.0	0.0	2483820	2456	??	S	7:09PM	0:00.13	/usr/sbin/distnoted agent
charlie	14631	0.0	0.0	2465740	1500	??	S	7:09PM	0:00.25	/usr/sbin/cfprefsd agent
charlie	14629	0.0	0.0	2471500	1576	??	Ss	7:09PM	0:00.14	/sbin/launchd
charlie	14616	0.0	0.3	2572380	25728	??	Ss	7:09PM	0:01.50	/System/Library/CoreServices/loginwin
kelly	14608	0.0	0.0	2465392	2036	??	S	7:09PM	0:00.02	/System/Library/CoreServices/AirPort
kelly	14603	0.0	0.2	2516072	14220	??	Ss	7:09PM	0:00.31	com.apple.dock.extra

Live Response System Profiler



```
system_profiler -xml -detaillevel full >  
/Volume/IR_CASE/sys_prof_MBP.spx
```

Open in "System Information.app" or system_profiler command

- Hardware Information
- USB Information
- Network Information
- Firewall Settings
- Mounted Volumes
- System Information
- Applications
- Kernel Extensions
- Log Data

© SANS.
All Rights Reserved

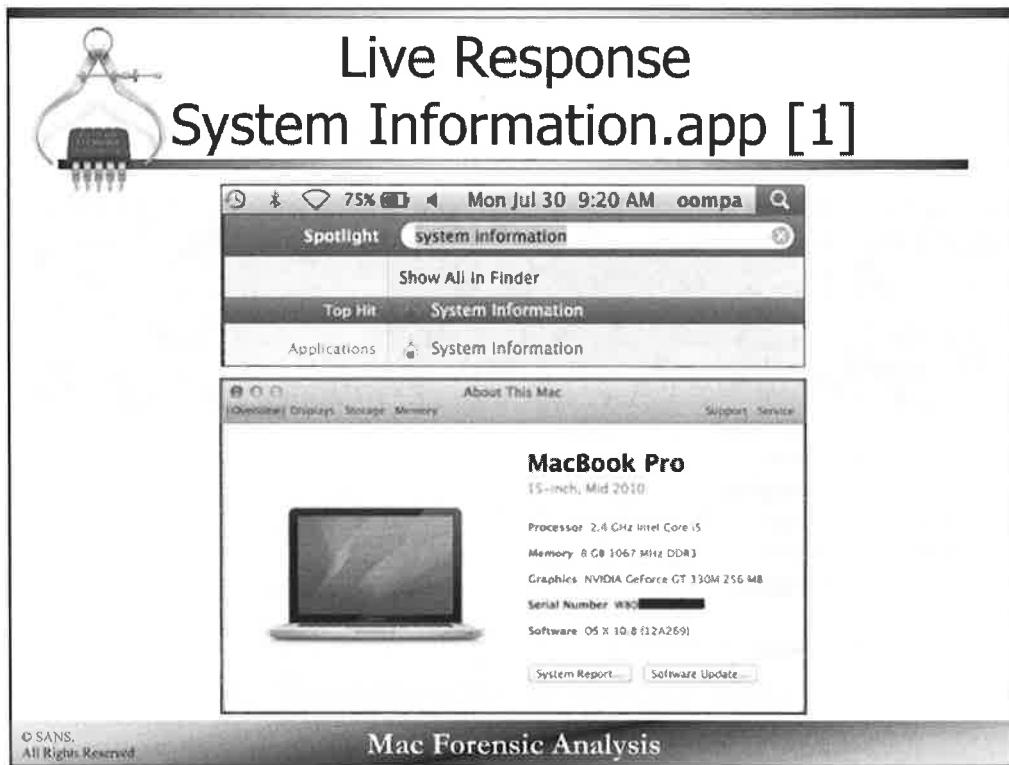
Mac Forensic Analysis

The `system_profiler` command is the command-line version of the System Information application.

The `system_profiler` command has different output formats and detail level options. As long as the output is named with a ".spx" file extension, the data can be viewed in the System Information.app. The XML output option can be used with the System Information.app located in `/Applications/Utilities/`. The output levels can be set as 'mini', 'basic', and 'full' or by the specific data type that can be accessed by the command:

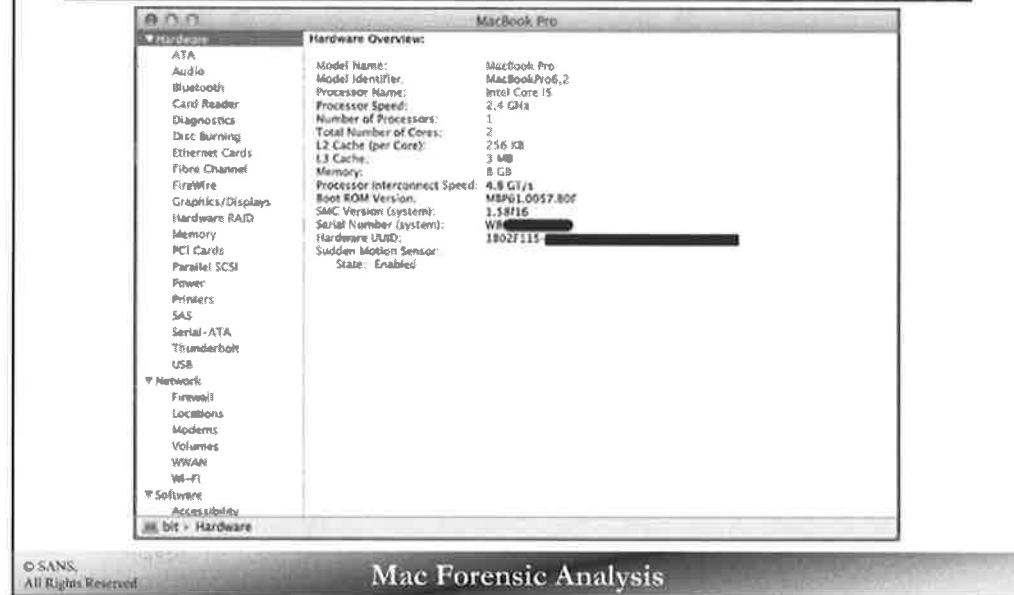
```
system_profiler -listDataTypes  
Available Datatypes:  
SPParallelATADataType  
SPUniversalAccessDataType  
SPApplicationsDataType  
SPAUDIODataType  
SPBluetoothDataType  
SPCardReaderDataType  
SPComponentDataType  
SPDeveloperToolsDataType  
SPDiagnosticsDataType  
SPDiscBurningDataType  
SPEthernetDataType  
SPExtensionsDataType  
SPFibreChannelDataType  
SPFireWireDataType  
SPFirewallDataType  
SPFontsDataType  
SPFrameworksDataType  
SPDisplaysDataType  
SPHardwareDataType
```

...



These screen shots show how to access the System Information.app application from the Spotlight menu or from the “About This Mac” menu available by selecting the Apple logo in the top left corner of the screen and selecting “System Report”.

Live Response: System Information.app [2]



This screen shot shows the types of data available from the System Information.app application or from the system_profiler command.

This tool is native to OS X.

You may also open the System Information.app application and choose File | Save to save this information directly from the GUI rather than on the command line.



MacBook Pro

Hardware Overview

ATA	Model Name:	MacBook Pro
Audio	Model Identifier:	MacBookPro6,2
Bluetooth	Processor Name:	Intel Core i5
Card Reader	Processor Speed:	2.4 GHz
Diagnostics	Number of Processors:	1
Disc Burning	Total Number of Cores:	2
Ethernet Cards	L2 Cache (per Core):	256 kB
Fibre Channel	L3 Cache:	3 MB
FireWire	Memory:	8 GB
Graphics/Displays	Processor Interconnect Speed:	4.8 GT/s
Hardware RAID	Boot ROM Version:	MBP61.0057.B0f
Memory	SMC Version (System):	1.58f16
PCI Cards	Serial Number (System):	W8 [REDACTED]
Parallel SCSI	Hardware UUID:	1802F115-[REDACTED]
Power	Sudden Motion Sensor:	State: Enabled
Printers		
SAS		
Serial-ATA		
Thunderbolt		
USB		
▼ Network		
Firewall		
Locations		
Modems		
Volumes		
WWAN		
Wi-Fi		
▼ Software		
Accessibility		

bit • Hardware



Exercise 1.1 – Mac Live Response

This page intentionally left blank.

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

© SANS,
All Rights Reserved

Mac Forensic Analysis

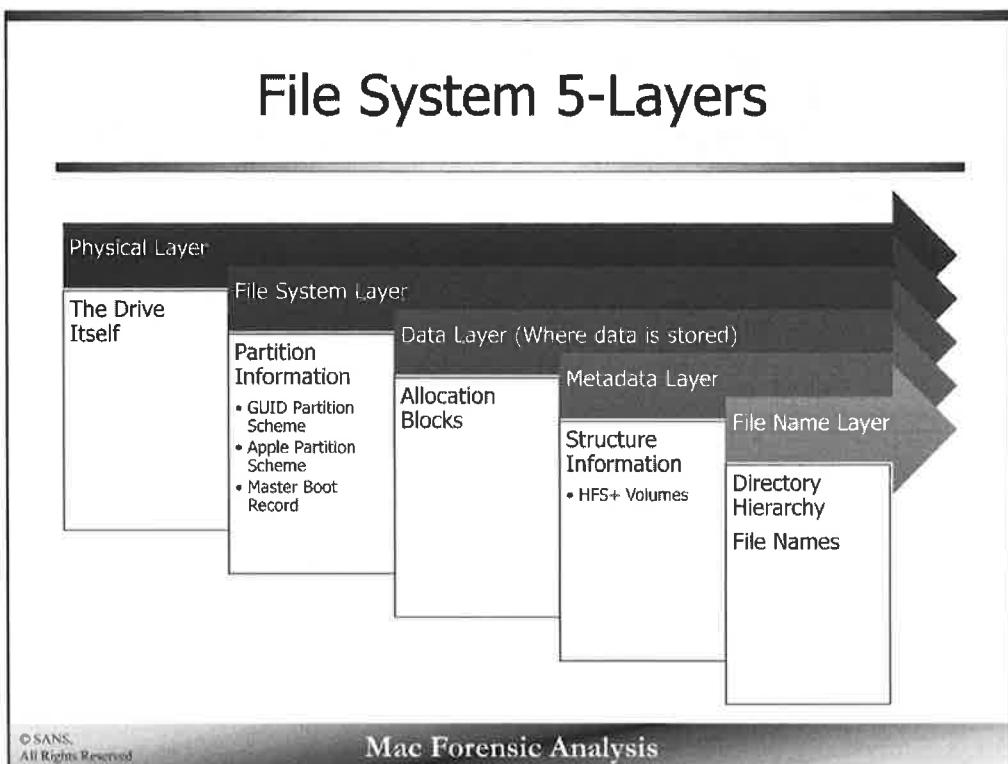
This page intentionally left blank.



Section 1 – Part 3 Disks & Partitions

This page intentionally left blank.

File System 5-Layers



A file system is comprised of five layers; physical, file system, data, metadata, and file name. These will be discussed in this section in detail.

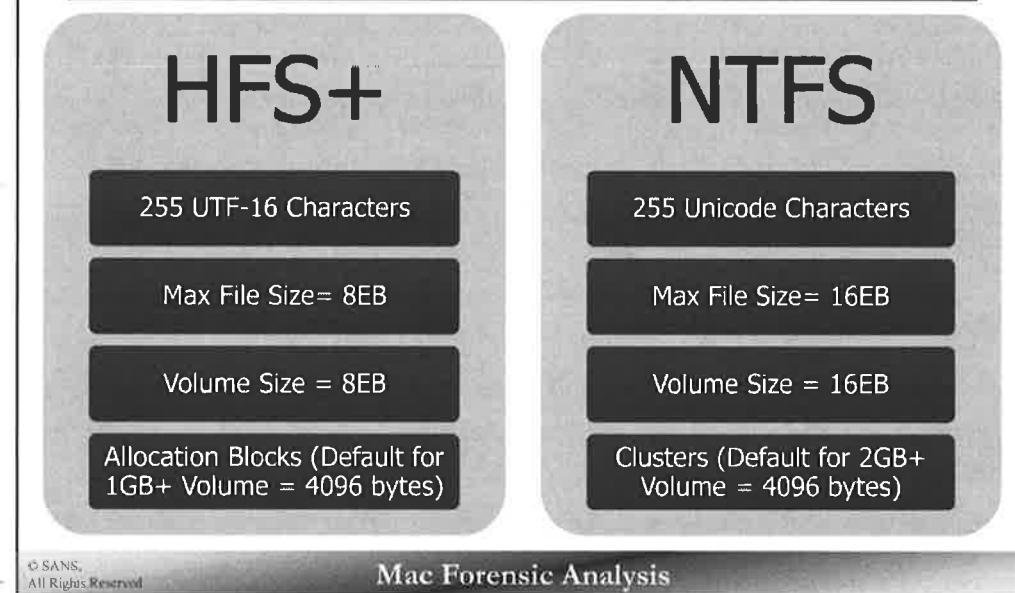
The physical and file system layers contain non-specific file system information such as the hardware and partitioning scheme. These will be discussed first to give a basis for the file system data that is presented on top of these structures.

The primary partitioning schemes found on Mac systems are the Globally Unique Identifier (GUID) Partition Scheme, Apple Partition Scheme and the Master Boot Record partitions.

The Allocation blocks used in the Data Layer are analogous to “Clusters” used in NTFS file systems.

The data, metadata, and file name layers contain the majority of the data that is presented and reviewed by forensic analysts. This will be discussed in greater detail in the following section.

HFS+ vs. NTFS



The primary file system used on Mac systems is the Hierarchical File System Plus (HFS+). Compared with New Technology File System (NTFS), it has a smaller maximum file and volume size, but similar in allocation block/cluster size.

Mac Partition Schemes

Apple Partition Map
(Apple Partition Scheme)

GUID Partition Table
(GUID Partition Scheme)

Master Boot Record

© SANS.
All Rights Reserved

Mac Forensic Analysis

Mac systems primarily use three types of partition schemes for different purposes.

A default installation of Mac OS X will partition the drive using the GUID Partition Scheme, creating an EFI partition and a protective MBR.

Disk image files (.dmg), can use the GUID Partition Scheme, the Apple Partition Scheme, a Master Boot Record, or no partition at all depending on what the user wants to do with the disk image.

Apple Partition Map vs. GUID Partition Table

APM

Open Firmware

PowerPC

GPT

Extensible
Firmware Interface

Intel x86

© SANS.
All Rights Reserved

Mac Forensic Analysis

Mac OS X installations on PowerPC hardware will have a Apple Partition Map and use Open Firmware, while Intel x86 based systems will implement GUID Partition Table with the Extensible Firmware Interface.

diskutil list



```

byte:~ oompa$ diskutil list
/dev/disk0
#:          TYPE NAME      SIZE    IDENTIFIER
0: GUID_partition_scheme *500.1 GB  disk0
   #:              NAME
1:   EFI             209.7 MB  disk0s1
2:   Apple_HFS Macintosh HD 499.2 GB  disk0s2
3:   Apple_Boot Recovery HD 650.0 MB  disk0s3

/dev/disk1
#:          TYPE NAME      SIZE    IDENTIFIER
0: FDisk_partition_scheme *8.0 GB   disk1
   #:              NAME
1:   DOS_FAT_32 NO NAME  8.0 GB   disk1s1

/dev/disk2
#:          TYPE NAME      SIZE    IDENTIFIER
0: FDisk_partition_scheme *2.0 TB   disk2
   #:              NAME
1:   Windows_NTFS WDPassport 2.0 TB   disk2s1

/dev/disk3
#:          TYPE NAME      SIZE    IDENTIFIER
0: FDisk_partition_scheme *3.5 GB   disk3
   #:              NAME
1:   DOS_FAT_32 Kindle   3.5 GB   disk3s1

/dev/disk4
#:          TYPE NAME      SIZE    IDENTIFIER
0: FDisk_partition_scheme *1.0 GB   disk4
   #:              NAME
1:   DOS_FAT_16 ORANGE  1.0 GB   disk4s1

```

© SANS,
All Rights Reserved

Mac Forensic Analysis

To view the disk partitions on the drives associated with a specific Mac you can issue the `diskutil list` command.

The output of the command shown in the screenshot above shows a variety of disk partition types.

- `/dev/disk0` contains the Mac OS X operating system. This disk uses the GUID Partition Scheme (shown as `GUID_partition_scheme`). Note the EFI partition.
- `/dev/disk1` is an external drive named “NO NAME” that uses an Master Boot Record (shown as `FDisk_partition_scheme`) as it is formatted as FAT32.
- `/dev/disk2` is another external drive named “WDPassport” that is formatted with NTFS and also uses the MBR.
- `/dev/disk3` is a Kindle device that uses the MBR with a FAT32 file system.
- `/dev/disk4` is another external USB named “ORANGE” drive that uses the MBR, however it is formatted with FAT16.

The output above also shows the number of partitions each disk contains, with the size and disk identifier.

This disk identifier is formatted in the format `disk#s#` where the first number is the disk number and the second is the partition or “slice”. This will be used later to reference a specific partition on a drive.

The same information can be found using the `Disk Utility.app` application located in `/Applications/Utilities/`.

```

byte:~ oompa$ diskutil list
/dev/disk0
#:
  #:          TYPE NAME
  0: GUID_partition_scheme
  1:           EFI
  2:      Apple_HFS Macintosh HD
  3:      Apple_Boot Recovery HD

/dev/disk1
#:
  #:          TYPE NAME
  0: FDisk_partition_scheme
  1: DOS_FAT_32 NO NAME

/dev/disk2
#:
  #:          TYPE NAME
  0: FDisk_partition_scheme
  1: Windows_NTFS WDPassport

/dev/disk3
#:
  #:          TYPE NAME
  0: FDisk_partition_scheme
  1: DOS_FAT_32 Kindle

/dev/disk4
#:
  #:          TYPE NAME
  0: FDisk_partition_scheme
  1: DOS_FAT_16 ORANGE

```

diskutil info

Command [1]

```
byte:~ kompa$ diskutil info disk0
Device Identifier: disk0
Device Node: /dev/disk0
Part of Whole: disk0
Device / Media Name: TOSHIBA MK5065GSXF Media

Volume Name: Not applicable (no file system)
Mounted: Not applicable (no file system)
File System: None
Content (IOContent): GUID_partition_scheme
OS Can Be Installed: No
Media Type: Generic
Protocol: SATA
SMART Status: Verified

Total Size: 500.1 GB (500107062016 Bytes) {exactly 976773168
512-Byte-Blocks}
Volume Free Space: Not applicable (no file system)
Device Block Size: 512 Bytes

Read-Only Media: No
Read-Only Volume: Not applicable (no file system)
Ejectable: No

Whole: Yes
Internal: Yes
Solid State: No
OS D Drivers: No
Low Level Format: Not supported
Device Location: "Lower"
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The `diskutil info` command is used on a specific disk or partition to display additional information about the disk or partition.

In the screenshot above `disk0` was used. The output shows various information such as:

- Make/Model of the hardware (Toshiba MK5065GSXF)
- The drive interface (SATA)
- Drive size in bytes and blocks
- Block Size
- Read-Only
- Location, useful for systems with multiple internal disks such as Mac Pro systems.

```
byte:~ oompa$ diskutil info disk0
Device Identifier:          disk0
Device Node:                /dev/disk0
Part of Whole:              disk0
Device / Media Name:        TOSHIBA MK5065GSXF Media

Volume Name:                Not applicable (no file system)
Mounted:                   Not applicable (no file system)
File System:                None
Content (IOContent):       GUID_partition_scheme
OS Can Be Installed:       No
Media Type:                 Generic
Protocol:                  SATA
SMART Status:              Verified

Total Size:                 500.1 GB (500107862016 Bytes) (exactly 976773168
512-Byte-Blocks)
Volume Free Space:         Not applicable (no file system)
Device Block Size:         512 Bytes

Read-Only Media:            No
Read-Only Volume:           Not applicable (no file system)
Ejectable:                  No

Whole:                      Yes
Internal:                   Yes
Solid State:                No
OS 9 Drivers:               No
Low Level Format:           Not supported
Device Location:             "Lower"
```

diskutil info Command [2]

```
byte#> osxps: diskutil info disk0s2
Device Identifier: disk0s2
Device Node: /dev/disk0s2
Part of Whole: disk0
Device / Media Name: Customer HD

Volume Name: Macintosh HD
Escaped with Unicode: Macintosh%F%FE%20%0BHD

Mounted: Yes
Mount Point: /
Escaped with Unicode: /

File System Personality: Journaled HFS+
Type (Bundle): hfs
Name (User Visible): Mac OS Extended (Journaled)
Journal: Journal size 40960 KB at offset 0xc30a000
Owners: Engbind

Partition Type: Apple_HFS
Can Be Installed: Yes
Media Type: Generic
Protocol: SATA
SMART Status: Verified
Volume UUID: C51CD139-A54F-3988-A787-213C0CBA6D71

Total Size: 409.2 GB (409248103424 Bytes) (exactly 975093952
512-byte-blocks)
Volume Free Space: 272.1 GB (272126102648 Bytes) (exactly 531495304
512-byte-blocks)
Device Block Size: 512 Bytes

Read-Only Media: No
Read-Only Volume: No
Ejectable: No

Wheate: No
Internal: Yes
Solid State: No
Device Location: "Lower"
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

Another example of the `diskutil info` command, used on the partition `disk0s2`.

Additional information is shown about the partition:

- Volume Name (Macintosh HD)
- File System (HFS+, Mac OS Extended)
- Volume Universal Unique Identifier
- Volume Size and Free Space

```
byte:~ oompa$ diskutil info disk0s2
  Device Identifier:          disk0s2
  Device Node:              /dev/disk0s2
  Part of Whole:            disk0
  Device / Media Name:      Customer

  Volume Name:              Macintosh HD
  Escaped with Unicode:     Macintosh%FF%FE%20%00HD

  Mounted:                  Yes
  Mount Point:              /
  Escaped with Unicode:     /

  File System Personality:  Journaled HFS+
  Type (Bundle):            hfs
  Name (User Visible):     Mac OS Extended (Journaled)
  Journal:                 Journal size 40960 KB at offset 0xe3Ba000
  Owners:                  Enabled

  Partition Type:           Apple_HFS
  OS Can Be Installed:     Yes
  Media Type:               Generic
  Protocol:                SATA
  SMART Status:             Verified
  Volume UUID:              C51CD139-A54F-3988-A787-213C0CBA6D71

  Total Size:               499.2 GB (499248103424 Bytes) (exactly 975093952
  512-Byte-Blocks)
  Volume Free Space:       272.1 GB (272126107648 Bytes) (exactly 531496304
  512-Byte-Blocks)
  Device Block Size:        512 Bytes

  Read-Only Media:          No
  Read-Only Volume:         No
  Ejectable:                No

  Whole:                    No
  Internal:                Yes
  Solid State:              No
  Device Location:          "Lower"
```

Apple Partition Map

diskutil & TSK's mmls

- Introduced in 1987 - Contains partitions for drivers and patches
- Apple_partition_map – Partition Map
- Apple_Driver_ATAPI – ATAPI Device Driver
- Apple_HFS – HFS+ Volume

/dev/disk5	#:	TYPE	NAME	SIZE	IDENTIFIER
	0:	Apple_partition_scheme		*41.0 MB	disk5
	1:	Apple_partition_map		32.3 KB	disk5s1
	2:	Apple_HFS	Secrets	41.0 MB	disk5s2

word:Documents oompa\$ mmls SecretDocs.dmg
MAC Partition Map
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: ----- 0000000000 0000000000 0000000001 Unallocated
01: 00 0000000001 0000000063 0000000063 Apple_partition_map
02: Meta 0000000001 0000000002 0000000002 Table
03: 01 0000000064 0000080063 0000080000 Apple_HFS

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Apple Partition Map partitioning scheme was introduced in 1987. APM is known to create multiple partitions with various drivers and patches contained in them. This creates partitions with names such as Apple_Driver_ATAPI that contains ATAPI drivers or Apple_FWDriver that contains FireWire drivers. Many of these partition names have been enumerated on the Wikipedia page for Apple Partition Map.

[http://en.wikipedia.org/wiki/Apple_Partition_Map]

The partition names that you will want to look for should be named Apple_HFS. This is the primary partition containing the user and system data. The Apple_partition_map partition contains the partition map and is also a partition in and of itself.

In the screenshot above, the example shown as disk3 is the partition table located on a Mac OS X installation DVD.

References:

File System Forensic Analysis – Brian Carrier - pp.101-107

Mac OS X Internals - Amit Singh – pp. 1349-1352

TN2166 – Secrets of the GPT

https://developer.apple.com/library/mac/technotes/tn2166/_index.html

		TYPE	NAME	SIZE	IDENTIFIER
#:					
0:		Apple_partition_scheme		*41.0 MB	disk5
1:		Apple_partition_map		32.3 KB	disk5s1
2:		Apple_HFS	Secrets	41.0 MB	disk5s2

word:Documents oompa\$ mm1s SecretDocs.dmg
MAC Partition Map
Offset Sector: 0
Units are in 512-byte sectors

	Slot	Start	End	Length	Description
00:	---	0000000000	0000000000	0000000001	Unallocated
01:	00	0000000001	0000000063	0000000063	Apple_partition_map
02:	Meta	0000000001	0000000002	0000000002	Table
03:	01	0000000064	0000080063	0000080000	Apple_HFS

GUID Partition Table on OS X

Booting from a GPT Partitioned disk available on Intel-based Macs.
(~2006+)

Mounting a GPT Partitioned disk available in 10.4+

APM limited to 2TB disks

Little Endian (mostly)

Technical Note 2166 – Secrets of the GPT

© SANS.
All Rights Reserved.

Mac Forensic Analysis

The GUID Partition Table implemented on Intel-based Mac systems started in 2006 with the transition to Intel hardware; however, mounting a GPT partition disk was available in Mac OS X 10.4.

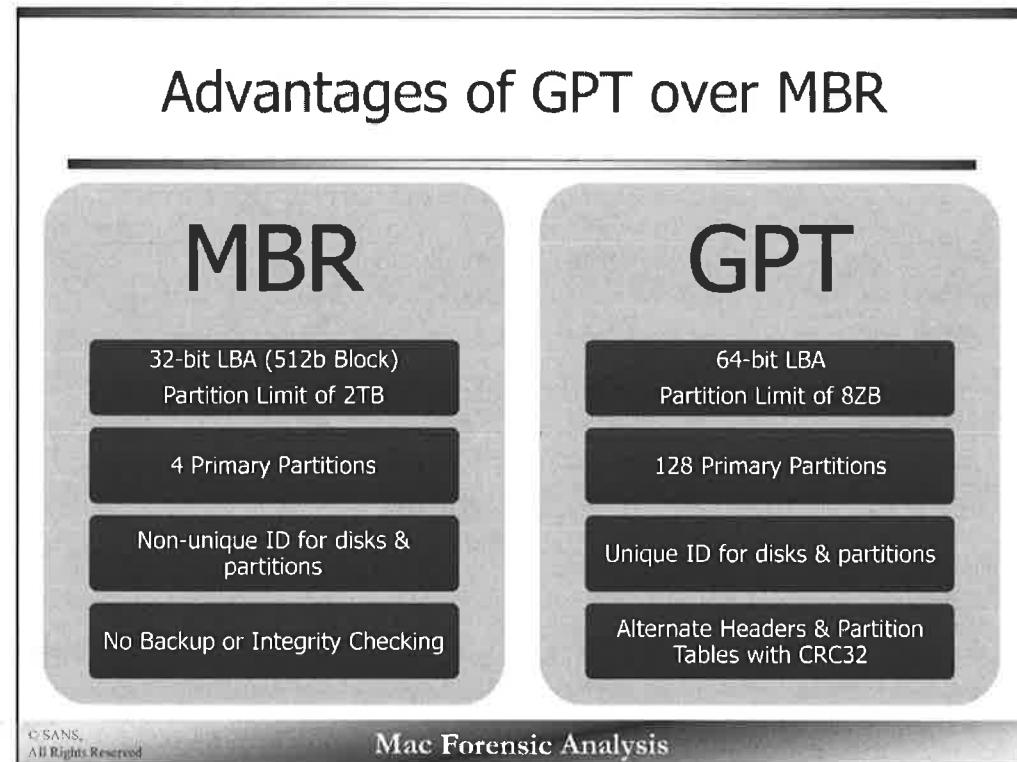
The switch to GPT from APM was due to the limited disk size that APM could not provide for.

Technical Note 2166 provides information pertaining to GPT used on Mac systems.

References:

TN2166 – Secrets of the GPT

https://developer.apple.com/library/mac/technotes/tn2166/_index.html



As stated earlier, the switch to GPT to APM was due to limitations such as a limited to a 2TB drive size. The Master Boot Record also had this drive size limit.

There were also limits with the number of partitions. MBR is limited to four primary partitions, while GPT has room for 128 primary partitions.

Other advantages of GPT are the use of unique identification (GUID) for disks and partitions and backups of the GPT header and partition table.

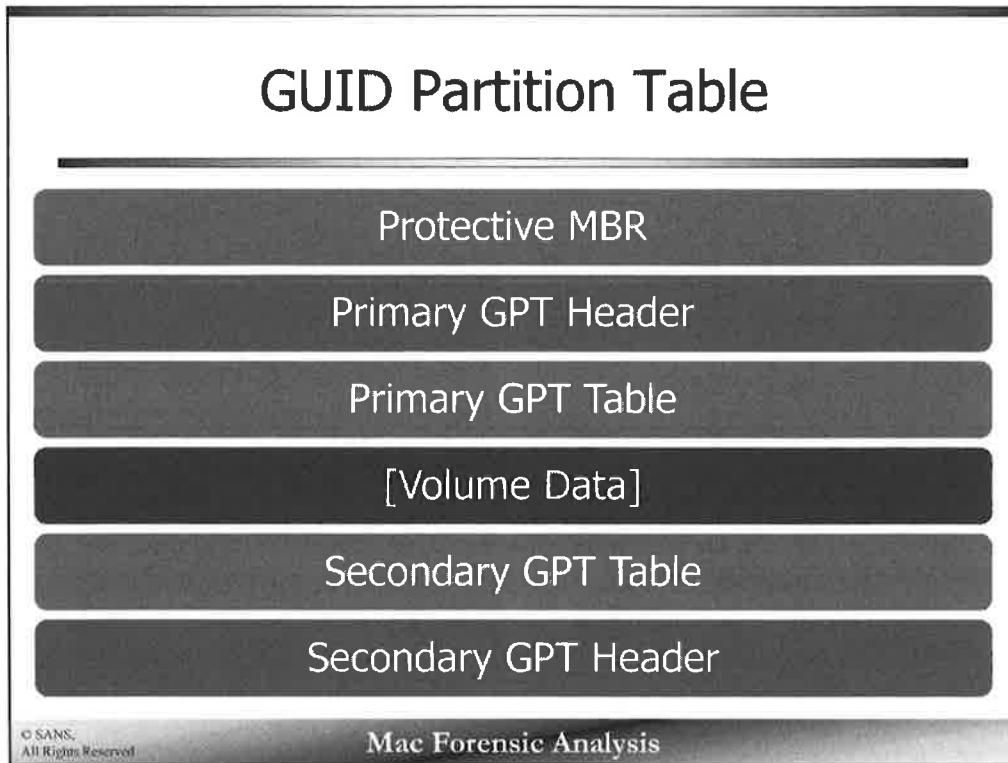
Note:

ZB = Zetta Bytes

LBA = Logical Block Addressing

References:

Forensic Analysis of GPT Disks and GUID Partition Tables – Bruce J Nikkel –
<http://www.digitalforensics.ch/nikkel09.pdf>



A GUID partition table is comprised of five parts:

- Protective Master Boot Record
- Primary GUID Partition Table Header
- Primary GUID Partition Table
- Secondary GUID Partition Table
- Secondary GUID Partition Table Header

The secondary GPT Header and tables are backups of their primary counterparts.

sudo gpt -r show [-l] Command

```
byte:~ oompa$ sudo gpt -r show disk0
      start    size index contents
        0        1
        1        1
        2       32
       34        6
       40     409600
409640 975093952
975503592 1269536
976773128 7
976773135 32
976773167 1
byte:~ oompa$ sudo gpt -r show -l disk0
      start    size index contents
        0        1
        1        1
        2       32
       34        6
       40     409600
409640 975093952
975503592 1269536
976773128 7
976773135 32
976773167 1
          1 GPT part - C12A732B-F81F-11D2-BA4B-00A0C93EC938
          2 GPT part - 48465300-0000-11AA-AA11-00306543ECAC
          3 GPT part - 426F6F74-0000-11AA-AA11-00306543ECAC
          1 Sec GPT table
          2 Sec GPT header
          3 Sec GPT header
          1 Pri GPT header
          2 Pri GPT table
          3 Pri GPT table
          1 GPT part - "EFI System Partition"
          2 GPT part - "Customer"
          3 GPT part - "Recovery HD"
          1 Sec GPT table
          2 Sec GPT header
          3 Sec GPT header
```

Mac Forensic Analysis

© SANS.
All Rights Reserved.

The screenshot above shows the output of the command 'sudo gpt -r show' and the 'sudo gpt -r show -l'. The 'l' option is used to toggle the display of the partition label from the GUID number.

This command is used to print the GUID **partition** table information for the specified disk. The '-l' option can be used to toggle the display of the **partition** label from the GUID number.

The -r argument is used to show the **GPT** information in a read-only format.

The output of the command contains the **starting** sector, the size of the partition (in sectors), and the contents. The PMBR is the Protective Master Boot **Record**. Note the primary and secondary GPT header and tables.

Each partition type has an associated **GUID** that will be the same across all partitions of the same type.

```

byte:~ oompa$ sudo gpt -r show disk0
      start    size   index contents
      0        1       1      PMBR
      1        1       1      Pri GPT header
      2        32      1      Pri GPT table
      34      32      6      Sec GPT table
      40      409600   1      GPT part - C12A7328-F81F-11D2-BA4B-00A0C93EC938
      409640  975093952  2      GPT part - 48465300-0000-11AA-AA11-00306543ECAC
      975503592 1269536  3      GPT part - 426F6F74-0000-11AA-AA11-00306543ECAC
      976773128 976773135  7
      976773167 32      1      Sec GPT table
      1        1       1      Sec GPT header
start    size   index contents
      0        1       1      PMBR
      1        1       1      Pri GPT header
      2        32      1      Pri GPT table
      34      32      6      Sec GPT table
      40      409600   1      GPT part - "EFI System Partition"
      409640  975093952  2      GPT part - "Customer"
      975503592 1269536  3      GPT part - "Recovery HD"
      976773128 976773135  7
      976773167 32      1      Sec GPT table
      1        1       1      Sec GPT header

```

GPT Partition Type GUIDs

Partition Type GUID – Unique to Partition Type

Volume GUID – Unique Across All Volumes, Globally

Type	GUID
EFI System Partition	C12A7328-F81F-11D2-BA4B-00A0C93EC93B
HFS+ Partition	48465300-0000-11AA-AA11-00306543ECAC H F S
Apple Boot Partition	426F6F74-0000-11AA-AA11-00306543ECAC B o o t
Apple Core Storage (FileVault)	53746F72-6167-11AA-AA11-00306543ECAC S t o r a g
Basic Data Partition (Boot Camp)	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7

© SANS.
All Rights Reserved.

Mac Forensic Analysis

Similar to the MBR, each partition type has its own identifier. GPT uses GUIDs. In the GPT Table example, there were three partitions:

- EFI System Partition - C12A7328-F81F-11D2-BA4B-00A0C93EC93B (EFI System Partition)
- Untitled - 48465300-0000-11AA-AA11-00306543ECAC (HFS+ Partition)
- Recovery HD - 426F6F74-0000-11AA-AA11-00306543ECAC (Apple Boot Partition)

The Apple created GUIDs contain “hidden” mnemonic type hints.

- HFS+ Partition GUIDs contain the string “HFS”
- Apple Boot Partition GUIDs contain the string “Boot”
- Apple Core Storage Partition GUIDs contain the string “Storag”

More GUID partition types can be found on the Wikipedia article for GUID Partition Table

[http://en.wikipedia.org/wiki/GUID_Partition_Table]

GPT GUID Format

On-Disk Format:

726F7453-6761-AA11-AA11-00306543ECAC

Converted Format:

53746F72-6167-11AA-AA11-00306543ECAC

53746F72 | 6167 | 11AA | AA11 | 00306543ECAC

Little Endian

Big Endian

© SANS.
All Rights Reserved

Mac Forensic Analysis

GPT GUIDs use a combination of little and big endian.

The converted format uses little endian (by byte) for the first three sections of the GUID, while using big endian for the last two.

Show GPT with The Sleuth Kit (TSK)

mmls <image>

```
byte:IMAGES compa$ mmls lion_macbook.dmg
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

  Slot  Start      End      Length     Description
00: Meta 0000000000 0000000000 0000000001 Safety Table
01: ----- 0000000000 0000000039 0000000040 Unallocated
02: Meta 0000000001 0000000001 0000000001 GPT Header
03: Meta 0000000002 0000000033 0000000032 Partition Table
04: 00 0000000040 0000409639 0000409600 EFI System Partition
05: 01 0000409640 0233172071 0232762432 Untitled
06: 02 0233172072 0234441607 0001269536 Recovery HD
07: ----- 0234441608 0234441647 0000000040 Unallocated
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Sleuth Kit (TSK) can be used to review the GPT partition scheme using the `mmls` command. TSK is a free, open-source tool used to interact with disks, or disk images to provide forensic analysis via the command-line.

In the screenshot above `mmls` is directed at a DMG image of a MacBook with a Mac OS X Lion installation. The Sleuth Kit calls the Protective MBR the Safety Table.

The Sleuth Kit can be downloaded from <http://sleuthkit.org/>. TSK man pages are located here: <http://www.sleuthkit.org/sleuthkit/man/>.

byte:IMAGES oompa\$ mmls lion_macbook.dmg

GUID Partition Table (EFI)

Offset Sector: 0

Units are in 512-byte sectors

Slot	Start	End	Length	Description
00:	Meta	000000000	000000000	Safety Table
01:	---	000000000	000000039	Unallocated
02:	Meta	000000001	000000001	GPT Header
03:	Meta	000000002	000000033	Partition Table
04:	00	000000040	0000409639	EFI System Partition
05:	01	0000409640	0233172071	Untitled
06:	02	0233172072	0234441607	Recovery HD
07:	---	0234441608	0234441647	Unallocated

Protective MBR (Safety Table)

LBA 0

Backwards Compatibility

Same Format

- Partition Table - Offset 446
- 4 - 16 byte Partition Entries
- One Partition Entry – Entire Disk
 - Partition Type:
 - 0xEE (EFI GPT Disk)
 - 0xAF – Apple Mac OS X HFS & HFS+
 - Three Partition Entries – Zeros

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Protective Master Boot Record, or Safety Table as it may be called, is used for backwards compatibility. If the software does not recognize the GPT format, the Protective MBR may be used to safeguard the GPT data.

This MBR uses the same format as any other MBR with the exception that it only has one partition with the partition type 0xEE for EFI GPT Disk. The remaining three primary partitions are left blank.

Typical Mac-related Partition Types:

- 0xAF – Apple Mac OS X HFS & HFS+
- 0xEE – EFI GPT Disk (EFI Protective MBR)

Protective MBR Example & Format

The format of the highlighted partition entry is shown in the table below. The partition type outlined in the box, 0xEE, is associated with an EFI GPT Disk partition. The size of the partition is 234,441,647 (0xAF4BF90D) in sectors. Note the 0x55AA signature at the bottom. For more information about the Master Boot Record please review book, File System Forensic Analysis by Brian Carrier.

<u>Note: GPT data uses little-endian</u>	Offset	Byte(s)	Field
CHS = Cylinder-Head-Sectors	446	1	Bootable Flag <ul style="list-style-type: none"> • 0x00 = non-bootable • 0x80 = bootable
References: File System Forensic Analysis – Brian Carrier - p.64, Table 5.2	447	3	Starting CHS Address (Little Endian)
	450	1	Partition Type
	451	3	Ending CHS Address (Little Endian)
	454	4	Starting LBA Address (Little Endian)
	458	4	Size in Sectors (Little Endian)

GPT Header Example

© SANS.
All Rights Reserved.

Mac Forensic Analysis

The GPT Header is 512 bytes and contains a human-recognizable signature, “EFI PART”. Items of interest include:

- At offset 24, the LBA (Little Endian) of the GPT Header: 1 (0x0100000000000000)
 - At offset 32, the LBA (Little Endian) of the alternate GPT header: 234441647 (0xAF4BF90D)
 - At offset 56, the unique GUID of this disk:
 - 71EB5B4D-5799-4224-9E64-3A5F11A5D050
 - At offset 72, the LBA (Little Endian) of the GUID Partition Table: 2 (0x0200000000000000)

Note: GPT data uses little-endian

- GUID Endianess - The first three sections are in little-endian while the last two are not.

Reference:

TN2166 – Secrets of the GPT

https://developer.apple.com/library/mac/technotes/tn2166/_index.html

http://www.uefi.org/specs/download/2_3_1_ErrataC.pdf
available through <http://www.uefi.org/specs/access>

(Registration Required)	80	4	Number of Partition Entries Available
Section 5.3.2			

Section 3.3.2	84	4	Available
	88	4	Size of Partition Entry
			Partition Entry Array CRC32

Offset	Size (bytes)	Field
0	8	Signature (EFI PART)
8	4	Revision (1.0)
12	4	Size of Header (bytes)
16	4	Header CRC32
20	4	Reserved
24	8	LBA of GPT Header
32	8	LBA of Backup GPT Header
40	8	First Usable LBA
48	8	Last Usable LBA
56	16	GUID of Partition
72	8	Starting LBA of GUID Partition Table
80	4	Number of Partition Entries Available
84	4	Size of Partition Entry
88	4	Partition Entry Array CRC32
92	Rest	Reserved

	45	46	49	20	50	41	52	54	00	00	01	00	5C	00	00	1D	B2	06	32	00	00	00	00	EFI PART . \2...	
000	00	00	00	00	00	00	00	00	AF	48	F9	0D	00	00	00	22	00	00	00	00	00	00	00	K.....K....."	
004	01	00	00	00	00	00	00	00	00	00	00	00	99	57	24	42	9E	64	3A	5F	11	A5	D0	50	K.....M[.q.W\$B.d:_P.....S\.....
008	0E	48	F9	0D	00	00	00	40	5B	EB	71	99	57	24	42	00	00	00	00	00	00	00	00		
012	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
016	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
024	144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
028	168	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
032	192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
036	216	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
040	240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
044	264	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
048	288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
052	312	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
056	336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
060	360	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
064	384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
068	408	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
072	432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
076	456	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
080	480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
084	504	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

GPT Table Example

00000	28	73	2A	C1	1F	F8	D2	11	BA	4B	00	A0	C9	3E	C9	3B	(s*.....K...>:)
00016	CF	A2	13	DE	62	C7	B2	4D	AE	50	8B	63	C7	7C	4C	36	...b..M.P.c. L6
00032	28	00	00	00	00	00	00	00	27	48	06	00	00	00	00	00	(.....'@.....
00048	00	00	00	00	00	00	00	00	45	00	46	00	49	00	20	00E.F.I..
00064	53	00	79	00	73	00	74	00	65	00	60	00	20	00	50	00	S.y.s.t.e.m. .P.
00080	61	00	72	00	74	00	69	00	74	00	69	00	6F	00	6E	00	a.r.t.i.t.i.o.n.
00096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00128	00	53	46	48	00	00	AA	11	AA	11	00	30	65	43	EC	AC	.SFH.....8eC..
00144	E5	40	CD	83	CE	72	54	4C	AE	90	2C	0D	60	9F	44	64	.@...rTl...,.`Dd
00160	28	40	86	00	00	00	00	00	67	EC	E5	0D	00	00	00	00	(@.....g.....
00176	00	00	00	00	00	00	00	00	55	00	6E	00	74	00	69	00U.n.t.i.
00192	74	00	6C	00	65	00	64	00	80	00	00	00	00	00	00	00	t.l.e.d.....
00208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00256	74	6F	42	00	AA	11	AA	11	00	30	65	43	EC	AC	tooB.....8eC..		
00272	28	C7	C9	0C	03	86	9E	4E	A5	D7	21	6E	97	90	53	B9	(.....N.!n..S.
00288	68	EC	E5	0D	00	00	00	00	87	4B	F9	0D	00	00	00	00	h.....K.....
00304	00	00	00	00	00	00	00	00	52	00	65	00	63	00	6F	00R.e.c.o.
00320	76	00	65	00	72	00	79	00	20	00	48	00	44	00	00	00	v.e.r.y. .H.D...
00336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

© SANS.
All Rights Reserved

Mac Forensic Analysis

The GPT Table is made up of an array of partition table entries with the format shown in the table below. In the screenshot above, a line splits each partition entry. The **middle** partition table will be analyzed; this partition entry contains the following items:

At offset 128, the **partition type** GUID is:

48465300-0000-11AA-AA11-00306543ECAC
(0x005346480000AA11AA1100306543ECAC)

This is a unique GUID to this **type** of partition.

At offset 144, the **unique partition** GUID is:

83CD40E5-72CE-4C54-AE90-2C0D609F4464
(0xE540CD83CE72544CAE902C0D609F4464)

This is a unique GUID only to **this** partition.

At offset 160, the starting LBA is 409640 (0x2840060000000000)

At offset 168, the ending LBA is 233172071 (0x67ECE50D00000000)

At offset 184, the partition name (in Unicode) is 'Untitled'.

Offset	Size (bytes)	Field
0	16	Partition Type GUID
16	16	Unique Partition GUID
32	8	Starting LBA (Little Endian)
40	8	Ending LBA (Little Endian)
48	8	Attributes
56	72	Partition Name
128	Rest	Reserved

Note: GPT data uses little-endian

- GUID Endianess - The first three sections are in little-endian while the last two are not.

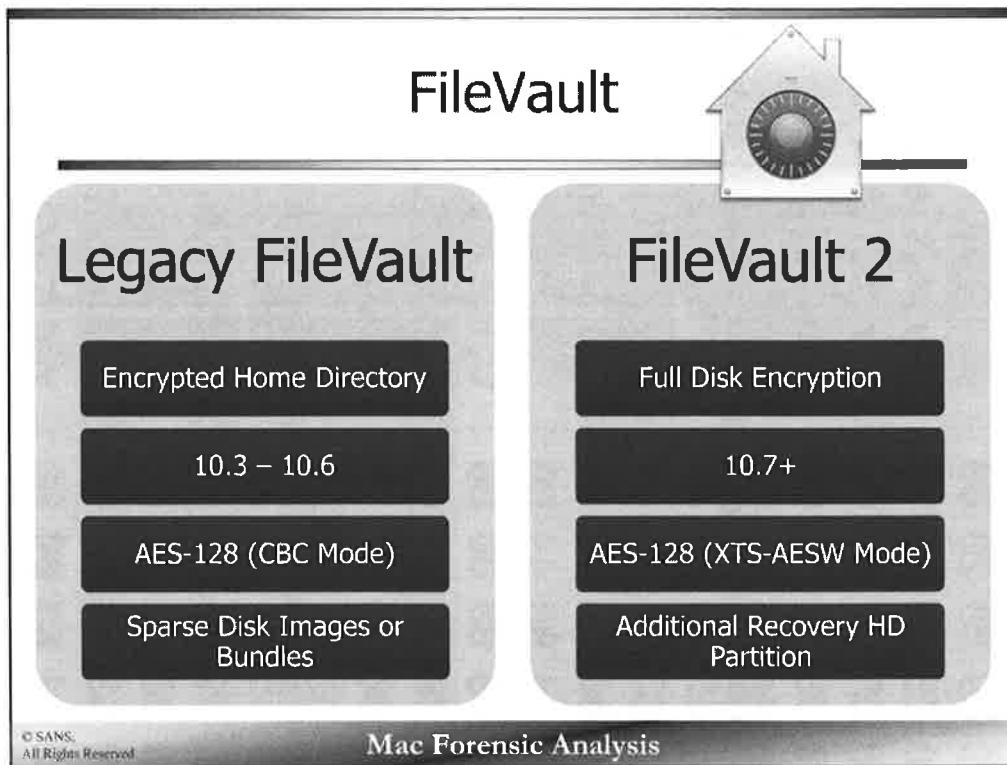
Reference:

TN2166 – Secrets of the GPT

https://developer.apple.com/library/mac/technotes/tn2166/_index.html

http://www.uefi.org/specs/download/2_3_1_ErrataC.pdf available though <http://www.uefi.org/specs/access>
(Registration Required)

000000	28	73	2A	C1	1F	F8	D2	11	B4	4B	00	A0	C9	3E	C9	3B	(s*K...	>;
000016	CF	A2	13	DE	62	C7	82	4D	AE	50	88	63	C7	7C	4C	36b..M.P.c..lL6		
000032	28	00	00	00	00	00	00	27	40	06	00	00	00	00	00	00	(.....@.....		
000048	00	00	00	00	00	00	00	45	00	46	00	49	00	20	00	00E.F.I..		
000064	53	00	79	00	73	00	74	00	65	00	60	00	20	00	50	00	S.y.s.t.e.m..P.		
000080	61	00	72	00	74	00	69	00	74	00	69	00	6F	00	6E	00	a.r.t.i.t.i.o.n.		
000096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000128	00	53	46	48	00	00	AA	11	AA	11	00	30	65	43	EC	AC	.SFH.....@eC..		
000144	E5	40	0D	83	CE	72	54	4C	AE	90	2C	00	60	9F	44	64@rTL...`..Dd		
000160	28	40	06	00	00	00	00	00	67	EC	E5	00	00	00	00	00	(@.....g.....		
000176	00	00	00	00	00	00	00	00	55	00	6E	00	74	00	69	00U.n.t.i.		
000192	74	00	6C	00	00	00	00	00	64	00	00	00	00	00	00	00	t.l.e.d..		
000208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	tooB.....@eC..		
000256	74	6F	6F	42	00	00	AA	11	00	30	65	43	EC	AC			(.....N..In..S..		
000272	28	C7	C9	0C	03	86	9E	45	07	21	6E	97	90	53	B9				
000288	68	EC	E5	00	00	00	00	00	87	48	F9	00	00	00	00	00	h.....K.....		
000304	00	00	00	00	00	00	00	00	00	00	00	00	52	00	65	00R.e.c.o.		
000320	76	00	65	00	72	00	79	00	20	00	48	00	44	00	00	00	y.e.r.y. .H.D...		
000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		



FileVault (now called Legacy FileVault) was introduced in 10.3. FileVault, if implemented, encrypts the home directory of a user. All other user directories and system files remain unencrypted. The encrypted home directory is stored in a sparse disk image, or a sparse bundle.

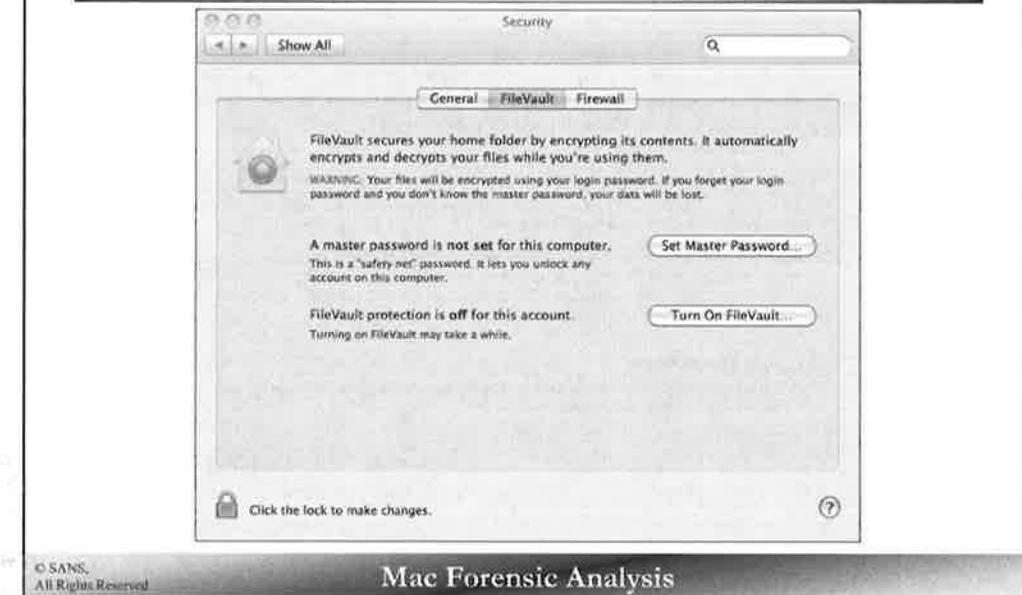
FileVault 2 was introduced in Mac OS X Lion, and encrypts the whole disk, except for EFI and Recovery partitions.

Reference:

OS X: About FileVault 2

[<http://support.apple.com/kb/HT4790>]

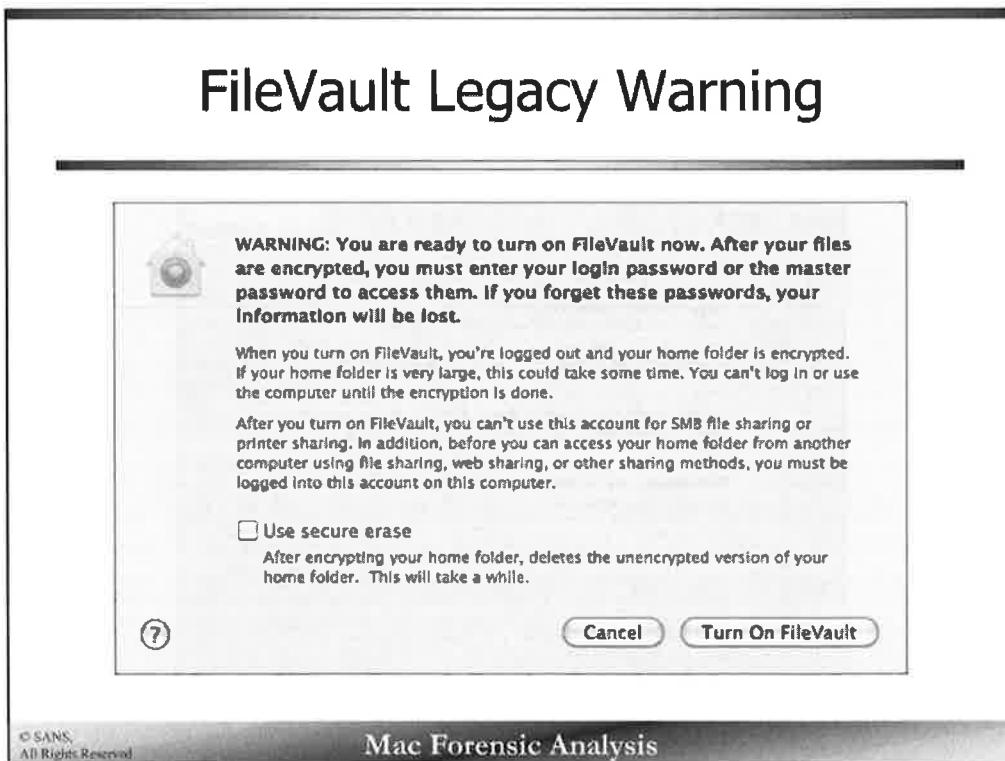
Legacy FileVault Setup



The screenshot above shows the window a user needs to implement the Legacy FileVault setting.

A master password needs to be set before FileVault can be turned on. This master password is used to decrypt any FileVault volume on the system. You may want to ask for the Master Password in the event you are in an enterprise situation.

FileVault Legacy Warning

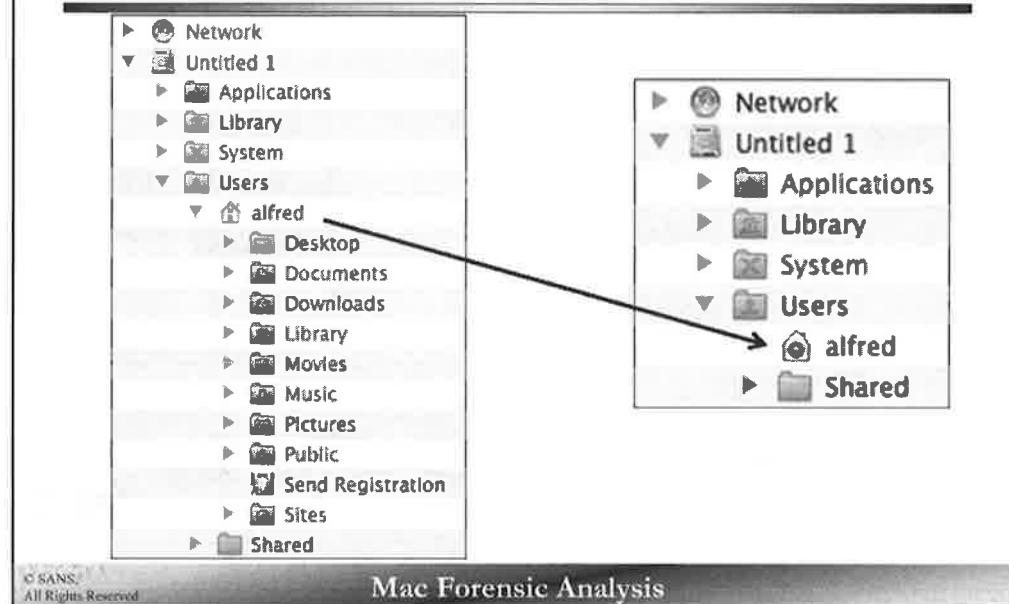


A warning is presented to the user before they are able to turn on FileVault.

A checkbox is enabled for the user to secure erase the disk space after the encryption process takes place.

Note: If the user does not secure erase their home folder, an analyst may be able to carve items from disk free space that would have otherwise been encrypted.

Home Directory with and without FileVault



Once a user's home directory is encrypted the house icon changes to a locked house icon, shown in the screenshot above. The folder structure of the user can no longer be seen by other users.

Legacy FileVault diskutil list

```
alfreds-macbook:~ alfred$ diskutil list
/dev/disk0
#:          TYPE NAME
0: GUID_partition_scheme
1:           EFI
2:           Apple_HFS Untitled 1
/dev/disk2
#:          TYPE NAME
0: Apple_partition_scheme
1: Apple_partition_map
2:           Apple_HFS alfred
```

	SIZE	IDENTIFIER
0:	*320.1 GB	disk0
1:	209.7 MB	disk0s1
2:	319.7 GB	disk0s2
0:	*639.5 GB	disk2
1:	32.3 KB	disk2s1
2:	639.5 GB	disk2s2



User account alfred is currently logged in.

© SANS,
All Rights Reserved

Mac Forensic Analysis

Using the diskutil list command from earlier, we can see how it might look if Legacy FileVault is used on a home directory for the user 'alfred'.

Alfred's home directory is now on its own /dev/disk*. The size of the drive may be much larger than the size of the hard drive in the system. This is due to how the sparse image/bundle file works.

Legacy FileVault diskutil info

```
alfred@macbook:~ alfred$ diskutil info disk2
Device Identifier:      disk2
Device Node:            /dev/disk2
Part Of Whole:          disk2
Device / Media Name:    Apple sparse bundle disk image Media

Volume Name:
Escaped with Unicode:

Mounted:                No
File System:             None

Partition Type:          Apple_partition_scheme
Bootable:                Not bootable
Media Type:              Generic
Protocol:                Disk Image
SMART Status:            Not Supported

System Image:            Yes

Total Size:              639.5 GB (639457918976 Bytes) (exactly 1248941248
512-Byte-Blocks)
Volume Free Space:       Not Applicable

Read-Only Media:          No
Read-Only Volumes:        Not applicable (no filesystem)
Ejectable:                Yes

Whole:                  Yes
Internal:                No
OS 9 Drivers:            No
Low Level Format:        Not Supported
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

If we look at Alfred's home directory from the previous slide using the `diskutil info` command, we can see that it is contained in a sparse bundle disk image.

Sparse Bundles & Sparse Disk Images

Sparse Disk Image

Legacy FileVault
10.3 & 10.4

Single File

.sparseimage
File Extension

Sparse Bundle

Legacy FileVault
10.5 & 10.6

Multiple Files

.sparsebundle
File Extension

© SANS,
All Rights Reserved

Mac Forensic Analysis

Legacy FileVault implements the Sparse Disk Image and the Sparse Bundle to store encrypted data, these file formats may also be used without the context of FileVault. The Disk Utility.app can be used to create encrypted and unencrypted volumes using these file types.

The Sparse Disk Image is a single file with a .sparseimage file extension. The Sparse Bundle file is made up of multiple files (in a software bundle) with the file extension .sparsebundle.

These files work by increasing the size of the sparse disk or bundle file/s. When these types of volumes are mounted, their “Virtual Volume” size may be much larger than the size of the hard drive the file is located on.

The size of the files will not be reduced if data is removed, but can be fixed after volume is unmounted by using the command hdiutil compact. “Virtual volumes” may be larger than the size on which the volume is physically located.

Sparse Disk Image

Single file that grows as data is added

File Header “sprs”

```
0000000: 7370 7273 0000 0003 0000 0800 0000 0001 sprs.....
0000010: 0001 3880 0000 0000 0000 0000 0000 0000 ..8.....
0000020: 0001 3880 0000 0000 0000 0000 0000 0000 ..8.....
0000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....
0000040: 0000 0001 0000 0002 0000 0014 0000 0026 .....&
0000050: 0000 0027 0000 0028 0000 0005 0000 0003 ...'...('.....
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Sparse Disk Image is made up of one file that grows as data is added. The file has the signature “sprs”, as shown in the screenshot above.

Sparse Bundle

Multiple files that grow as data is added

Uses “bands” to hold data

```
byte:SECTION_1 oompa$ ls -laR sparse_bundle.sparsebundle/
total 16
drwxr-xr-x@ 6 oompa staff 204 Aug 11 12:33 .
drwxr-xr-x 17 oompa staff 578 Aug 11 13:17 ..
-rw-r--r-- 1 oompa staff 495 Aug 11 12:21 Info.bckup
-rw-r--r-- 1 oompa staff 495 Aug 11 12:21 Info.plist
drwxr-xr-x 5 oompa staff 170 Aug 11 12:21 bands
-rw-r--r-- 1 oompa staff 6 Aug 11 12:21 token

sparse_bundle.sparsebundle//bands:
total 30920
drwxr-xr-x 5 oompa staff 170 Aug 11 12:21 .
drwxr-xr-x@ 6 oompa staff 204 Aug 11 12:33 ..
-rw-r--r-- 1 oompa staff 4689920 Aug 11 12:33 0
-rw-r--r-- 1 oompa staff 3735552 Aug 11 12:21 2
-rw-r--r-- 1 oompa staff 7405568 Aug 11 12:21 4
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Sparse Bundle file is comprised of multiple files that grow as data is added to the volume. These files are called “bands”. These “bands” are located in a “Bundle”.

A bundle is a directory that contains files, executable, and/or other resources in one easy to move file. Any application on a Mac system is a “bundle”.

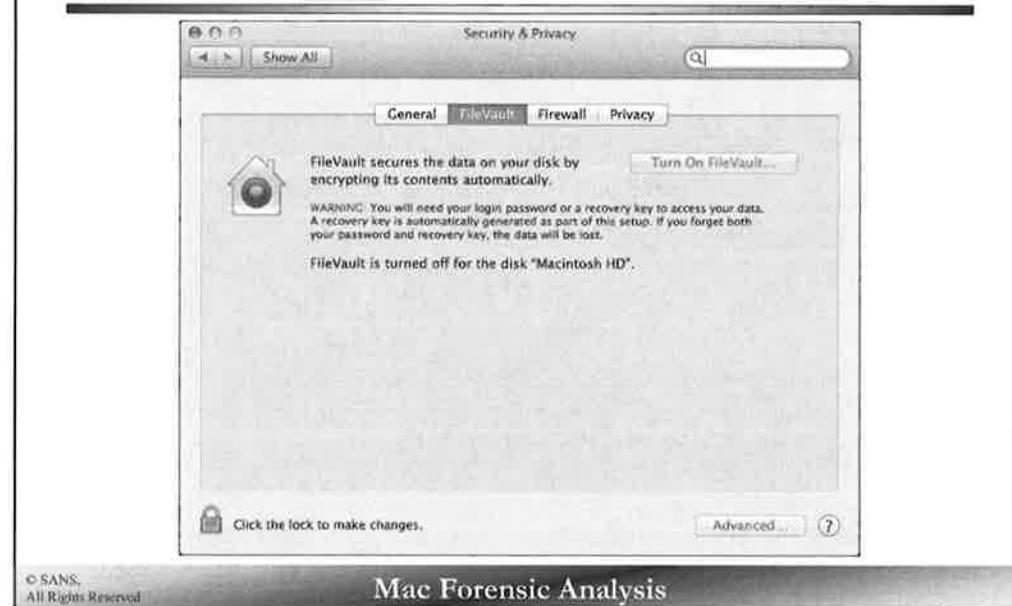
Each Sparse Bundle has an `Info.plist` file located within the bundle. This property list file contains the size of the bands, and the original size of the bundle, whether or not it is at capacity. Sizes are in bytes.

Reference:

Mac Developer Library – Cocoa Core Competencies

[<https://developer.apple.com/library/mac/#documentation/General/Conceptual/DevPedia-CocoaCore/Bundle.html>]

FileVault 2 Setup [1]



FileVault 2 uses a similar setup screen shown above.

FileVault 2 Setup [2]



© SANS,
All Rights Reserved

Mac Forensic Analysis

Before the encryption process takes place, the setup will display a recovery key that the user can choose to store with Apple. This recovery key can be used to recover a FileVault 2 encrypted volume.

GPT FileVault 2 Example

diskutil list

No FileVault

```
nibble:~ sledwards$ diskutil list
/dev/disk0
#:
```

	TYPE	NAME	SIZE	IDENTIFIER
0:	GUID_partition_scheme		*121.3 GB	disk0
1:	EFI		209.7 MB	disk0s1
2:	Apple_HFS	Macintosh HD	120.5 GB	disk0s2
3:	Apple_Boot	Recovery HD	650.0 MB	disk0s3

FileVault Enabled

```
word:~ oompa$ diskutil list
/dev/disk0
#:
```

	TYPE	NAME	SIZE	IDENTIFIER
0:	GUID_partition_scheme		*500.3 GB	disk0
1:	EFI	EFI	209.7 MB	disk0s1
2:	Apple_CoreStorage		499.4 GB	disk0s2
3:	Apple_Boot	Recovery HD	650.0 MB	disk0s3


```
/dev/disk1
#:
```

	TYPE	NAME	SIZE	IDENTIFIER
0:	Apple_HFS	Yosemite	*499.1 GB	disk1

Logical Volume on disk0s2
33F6B43B-7A75-4B04-A04D-0AC6A4321BF5
Unlocked Encrypted

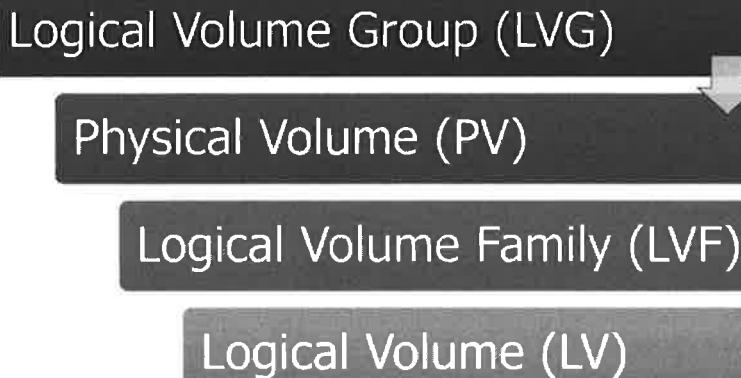
© SANS.
All Rights Reserved

Mac Forensic Analysis

In the top example, the screenshot shows the output of the `diskutil list` command on a disk that does not have FileVault 2 enabled.

The bottom example shows the same disk with FileVault 2 enabled. Note the addition of the `Apple_CoreStorage` partition and how “Macintosh HD” is now on `/dev/disk1`. The `Apple_CoreStorage` partition is the encrypted partition of the FileVault volume on `/dev/disk1`.

CoreStorage [1]



CoreStorage on by default on NEW installs of 10.10

© SANS.
All Rights Reserved

Mac Forensic Analysis

CoreStorage is Apple's version of logical volume management – or “virtual volumes”. Used with FileVault2 Full Disk Encryption and Fusion Drives.

Logical Volume Group (LVG) – The top level, associated with a specific Physical Volume.

Physical Volume (PV) - The physical disk, or disk image.

Logical Volume Family (LVF) – Contains one or more Logical Volumes.

Logical Volume (LV) – Where the file system is stored on a /dev/disk*.

As of 10.10, CoreStorage is now implemented by default on new installs of 10.10. Upgraded installs will not force CoreStorage to be implemented. This does not necessarily mean the volume is encrypted. CoreStorage can still be used without FileVault2.

References:

`diskutil manpage`

CoreStorage [2]

diskutil cs list

```
nbille:/ stewards$ diskutil cs list
CoreStorage logical volume groups (1 found)
|
+-- Logical Volume Group 068CB516-9C26-493A-9967-FC7977FE6B55
|   |
|   +-- Logical Volume Macintosh HD
|       |
|       Name: Macintosh HD
|       Status: Online
|       Size: 499418834176 B (499.4 GB)
|       Free Space: 16777216 B (16.8 kB)
|       |
|       +-< Physical Volume CAD32BAC-A06F-466C-9145-5A312C7B2C83
|           |
|           Index: 0
|           Disk: disk0s2
|           Status: Online
|           Size: 499418834176 B (499.4 GB)
|           |
|           +-> Logical Volume Family E4AGF218-0490-424C-AF11-4C208E052F58
|               |
|               Encryption Status: Unlocked
|               Encryption Type: AES-XTS
|               Conversion Status: Complete
|               Conversion Direction: -none-
|               Has Encrypted Extents: Yes
|               Fully Secure: Yes
|               Passphrase Required: Yes
|               |
|               +-> Logical Volume DD659E536-C674-4BDD-BD9E-9EE88FE857EE
|                   |
|                   Disk: disk1
|                   Status: Online
|                   Size (Total): 499082485760 B (499.1 GB)
|                   Conversion Progress: -none-
|                   Revertible: Yes (unlock and decryption required)
|                   LV Name: Macintosh HD
|                   Volume Name: Macintosh HD
|                   Content Hint: Apple_HFS
|
|   © SANS.
|   All Rights Reserved
```

Mac Forensic Analysis

The screenshot shows the output from the `diskutil cs list` command. This command lists all the CoreStorage Logical Volume Groups.

This system has only one LVG for a 499.4GB physical disk. Each part (LVG, PV, LVF, LV) has their own UUID.

We can see from the information found in the LVF section, that this disk is encrypted with AES-XTS (FileVault2).

We can see from the information found in the LV section, this disk is currently unlocked, and could be decrypted if needed. The volume is named “Macintosh HD” and is 499.1GB in size.

```
nibble:/ sledwards$ diskutil cs list
CoreStorage logical volume groups (1 found)
|
+- Logical Volume Group 068CB516-9C26-493A-9967-FC7977FE6855
=====
|   Name:      Macintosh HD
|   Status:    Online
|   Size:      499418034176 B (499.4 GB)
|   Free Space: 16777216 B (16.8 MB)
|
|   +-< Physical Volume CAD328AC-A06F-466C-9145-5A312C7B2C83
|   |
|   |   Index:    0
|   |   Disk:     disk0s2
|   |   Status:   Online
|   |   Size:     499418034176 B (499.4 GB)
|
|   +-> Logical Volume Family E4A6F218-0490-424C-AF11-4C208E052F5B
-----
|       Encryption Status:      Unlocked
|       Encryption Type:       AES-XTS
|       Conversion Status:    Complete
|       Conversion Direction: -none-
|       Has Encrypted Extents: Yes
|       Fully Secure:         Yes
|       Passphrase Required:  Yes
|
|   +-> Logical Volume DD650536-C674-4BDD-BD9E-9EEB8FE857EE
-----
|       Disk:          disk1
|       Status:        Online
|       Size (Total): 499082485760 B (499.1 GB)
|       Conversion Progress: -none-
|       Revertible:    Yes (unlock and decryption required)
|       LV Name:      Macintosh HD
|       Volume Name: Macintosh HD
|       Content Hint: Apple_HFS
```

Boot Camp & FileVault GPT Example



```
gpt show: /dev/disk0: Suspicious MBR at sector 0
      start      size  index  contents
          0          1      MBR
          1          1  Pri GPT header
          2         32  Pri GPT table
         34          6
        40     409600      1  GPT part - C12A732B-F81F-11D2-BA4B-00A0C93EC93B
  409640  877575752      2  GPT part - 53746F72-6167-11AA-AA11-00306543ECAC
 877985392   1269544      3  GPT part - 426F6F74-0000-11AA-AA11-00306543ECAC
 879254936  97850088      4  GPT part - EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
 977105024          3
 977105027         32  Sec GPT table
 977105059          1  Sec GPT header
```

```
gpt show: /dev/disk0: Suspicious MBR at sector 0
      start      size  index  contents
          0          1      MBR
          1          1  Pri GPT header
          2         32  Pri GPT table
         34          6
        40     409600      1  GPT part - "EFI System Partition"
  409640  877575752      2  GPT part - "Lion"
 877985392   1269544      3  GPT part - "Recovery HD"
 879254936  97850088      4  GPT part - "BOOTCAMP"
 977105024          3
 977105027         32  Sec GPT table
 977105059          1  Sec GPT header
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

In the screenshots above, there is an example of a Boot Camp partition and a FileVault encrypted partition. This is the same disk with the '-l' option of the gpt command used in the bottom example.

- Boot Camp Volume GUID: EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
- FileVault Volume GUID: 53746F72-6167-11AA-AA11-00306543ECAC

The “Suspicious MBR at sector 0” message shown in the screenshots above validate that it is in fact a boot camped system. When a system is boot camped, meaning it has both an bootable OS X volume and a bootable Windows volume, it writes multiple entries to the MBR making it “suspicious”.

```

gpt show: /dev/disk0: Suspicious MBR at sector 0
      start      size  index  contents
          0          1      MBR
          1          1  Pri GPT header
          2         32  Pri GPT table
         34          6
        40     409600      1 GPT part - C12A7328-F81F-11D2-BA4B-00A0C93EC93B
  409640  877575752      2 GPT part - 53746F72-6167-11AA-AA11-00306543ECAC
877985392   1269544      3 GPT part - 426F6F74-0000-11AA-AA11-00306543ECAC
879254936   97850088      4 GPT part - EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
977105024          3
977105027          32  Sec GPT table
977105059          1  Sec GPT header

```

```

gpt show: /dev/disk0: Suspicious MBR at sector 0
      start      size  index  contents
          0          1      MBR
          1          1  Pri GPT header
          2         32  Pri GPT table
         34          6
        40     409600      1 GPT part - "EFI System Partition"
  409640  877575752      2 GPT part - "Lion"
877985392   1269544      3 GPT part - "Recovery HD"
879254936   97850088      4 GPT part - "BOOTCAMP"
977105024          3
977105027          32  Sec GPT table
977105059          1  Sec GPT header

```

Fusion Drive

Hybrid Drive

Hard Disk Drive + Solid State Drive

Size & Speed

Implements CoreStorage

- Combined to create a single volume

Most accessed files located on SSD

Mac mini & iMac Systems

The Fusion Drive comes with only a few specific systems; late model MacMinis and iMacs.

These drives use a combination of a normal hard disk drive and a solid state drive to get the best of speed and drive size.

To create a present a single volume to the user, CoreStorage is implemented. The most used files are located on the physical SSD hard drive to speed up file access.

Disk Image (.dmg)

- Create with Disk Utility.app or hdiutil
- Any Size
- Formats:
 - Mac OS Extended
 - FAT
 - ExFAT (over 32GB)
- May configure with encryption, partitions, and image formats



© SANS,
All Rights Reserved

Mac Forensic Analysis

Likely the most well known volume is the DMG file, or Disk Image file. Mac systems use this format for everything from software installation to simple data archives.

Disk Image files can be any size and any format that hdiutil or Disk Utility.app can implement; HFS+, FAT or ExFAT.

These volumes can mix and match various types of encryption, partitions and image formats. Please refer to the options listed using the command `hdiutil create -help` or review the options in Disk Utility.app.

```

hdutil
imageinfo *.dmg

```

- Rename .dd to .dmg
 - Image Size
 - Compression Data
 - Image Format
 - Location
 - Detailed Partition Information
 - Resizing Information

```

file-MacBook-Pro-2012-06-14-01-26-42:~ user$ hdiutil imageinfo /dev/disk0
Backing Store Information:
  Unit Identifier: /dev/disk0
  Name: disk0
  Class Name: CoreBackingStore
  Class Model: CoreDiskImage
  Checksum Type: none
  Block Size: 512
  Total Bytes: 591107842816
  Compressed: 0
  Sector Count: 976773140
  Total Allocation Units: 5801011629916
  Compressed Bytes: 591107842816
  Total Empty Bytes: 0
Format: UDRW
Format Description: raw-read/write
Checksum Value: 0
Properties:
  Encrypted: false
  Write Committed: false
  Checksummed: false
  Software License Agreement: false
  Partitioned: false
  Compressed: no
Segments:
  @/dev/disk0
partitions:
  partition-scheme: GUID
  disk-size: 97G
  partitions:
    0:
      partition-name: Protective Master Boot Record
      partition-start: 0
      partition-size: 1M
      partition-length: 3
      partition-hints: MBR
    1:
      partition-name: GPT Header
      partition-start: 1
      partition-size: 1M
      partition-length: 1
      partition-hints: Primary GPT header
    2:
      partition-name: GPT Partition Data
      partition-start: 3M
      partition-size: 1M
      partition-length: 33
      partition-hints: Primary GPT table
    3:
      partition-name: free
      partition-start: 36
      partition-size: 1M
      partition-length: 6
      partition-hints: Apple_Free
    4:
      partition-number: 0x00000000-0000-0000-0000-000000000000
      partition-name: EFA System Partition
      partition-start: 40M
      partition-end: 41247328-P8P-1502-BA4B-004C93ECS9
      partition-size: 40M
      partition-number: 2
      partition-length: 4096000
      partition-hints: EFA
      partition-type: 0x00000000-0000-0000-0000-000000000000
      partition-filenames: /efi

```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The `hdutil` command is used to interface with disk image files (`*.dmg`) or disk device (`/dev/disk#`).

This command must be run with a `*.dmg` file, if you have a raw DD file you can add a `.dmg` extension.

The `imageinfo` parameter is used to display information about the disk image. (Due to the verbose output the screenshot had to be shortened to fit on the slide.)

Information that can be gleaned from an image include:

- Image Size (in bytes)
- Data about the compression, this image was not compressed.
- Image Format – this example is labeled UDRW or RAW Read/Write
 - A list of formats can be viewed in the man page for `hdutil`
- Image location
- Detailed Partition Information
 - Partition Name
 - Start Block
 - Length (in blocks)
 - Partition Hint – Type of Partition
 - Partition Hint UUID (unique to type of partition)
 - Partition UUID (unique to partition)
 - File System (HFS+, FAT32, etc.)
- Image file resizing information - Can be used to shrink a volume

```
byte:Macintosh HD 2012-06-14 07-36-02 oompa$ hdiutil imageinfo /dev/disk0
Backing Store Information:
    URL: file://localhost/dev/rdisk0
    Name: rdisk0
    Class Name: CDevBackingStore
Class Name: CRawDiskImage
Checksum Type: none
Size Information:
    Total Bytes: 500107862016
    Compressed Ratio: 1
    Sector Count: 976773168
    Total Non-Empty Bytes: 500107862016
    Compressed Bytes: 500107862016
    Total Empty Bytes: 0
Format: RAW*
Format Description: raw read/write
Checksum Value:
Properties:
    Encrypted: false
    Kernel Compatible: false
    Checksummed: false
    Software License Agreement: false
    Partitioned: false
    Compressed: no
Segments:
    0: /dev/rdisk0
partitions:
    partition-scheme: GUID
    block-size: 512
    partitions:
        0:
            partition-name: Protective Master Boot Record
            partition-start: 0
            partition-synthesized: true
            partition-length: 1
            partition-hint: MBR
        1:
            partition-name: GPT Header
            partition-start: 1
            partition-synthesized: true
            partition-length: 1
            partition-hint: Primary GPT Header
        2:
            partition-name: GPT Partition Data
            partition-start: 2
            partition-synthesized: true
            partition-length: 32
            partition-hint: Primary GPT Table
        3:
            partition-name:
            partition-start: 34
            partition-synthesized: true
            partition-length: 6
            partition-hint: Apple_Free
        4:
            partition-UUID: D97CE46F-D267-440E-BD20-8930EFCA721F
            partition-name: EFI System Partition
            partition-hint-UUID: C12A7328-F81F-11D2-BA4B-00A0C93EC93B
            partition-start: 40
            partition-number: 1
            partition-length: 409600
            partition-hint: C12A7328-F81F-11D2-BA4B-00A0C93EC93B
            partition-filesystems:
                FAT32: EFI
```



Exercise 1.2 – Disks & Partitions

This page intentionally left blank.

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

This page intentionally left blank.



Section 1 – Part 4

HFS+

© SANS.
All Rights Reserved.

Mac Forensic Analysis

This page intentionally left blank.

Mac OS Extended File System (HFS Plus or HFS+)

Hierarchical File System

Introduced in Mac OS 8.1 (~1998)

TN1150

Big Endian

HFS+ (Mac OS) Timestamps

- Number of Seconds since 1/1/1904 00:00:00 UTC
- 32-bit Unsigned Integer

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Mac OS Extended File System is more widely known as Hierarchical File System (HFS+), or HFS Plus. HFS+ was introduced when HFS (Mac OS Standard File System) was limiting the size of the installation media. HFS+ is able to install on larger volumes, have longer filenames, larger file sizes, and more file metadata.

This file system was introduced in Mac OS 8.1, long before Mac OS X.

A very well known Technical Note (at least to nerdy Mac forensicators) is TN1150, the HFS Plus Volume Format. This document explains the technical intricacies of the HFS+ file system and goes into great detail. The document can be found in the Mac Developer Library.

[<https://developer.apple.com/legacy/library/technotes/tn/tn1150> or
<http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html>]

Note: HFS+ data uses big-endian

TN1150 will be used as the main reference for the HFS+ file system module of this class.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source -- Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

HFS+ Volume Header & “Special Files”

Volume Header (Alternate Volume Header)

- Information about the volume

Allocation File

- Tracks Allocation Blocks

Catalog File

- File/Folder Structure & Metadata

Extents Overflow File

- Additional Extent Information

Attributes File

- Additional Metadata

Startup File

- Helps non-Mac OS Volumes boot

© SANS.
All Rights Reserved

Mac Forensic Analysis

HFS+ is comprised of five “special files” and the HFS+ volume header.

The special files are:

- Allocation File
- Catalog File
- Extents Overflow File
- Attributes File
- Startup File

References:

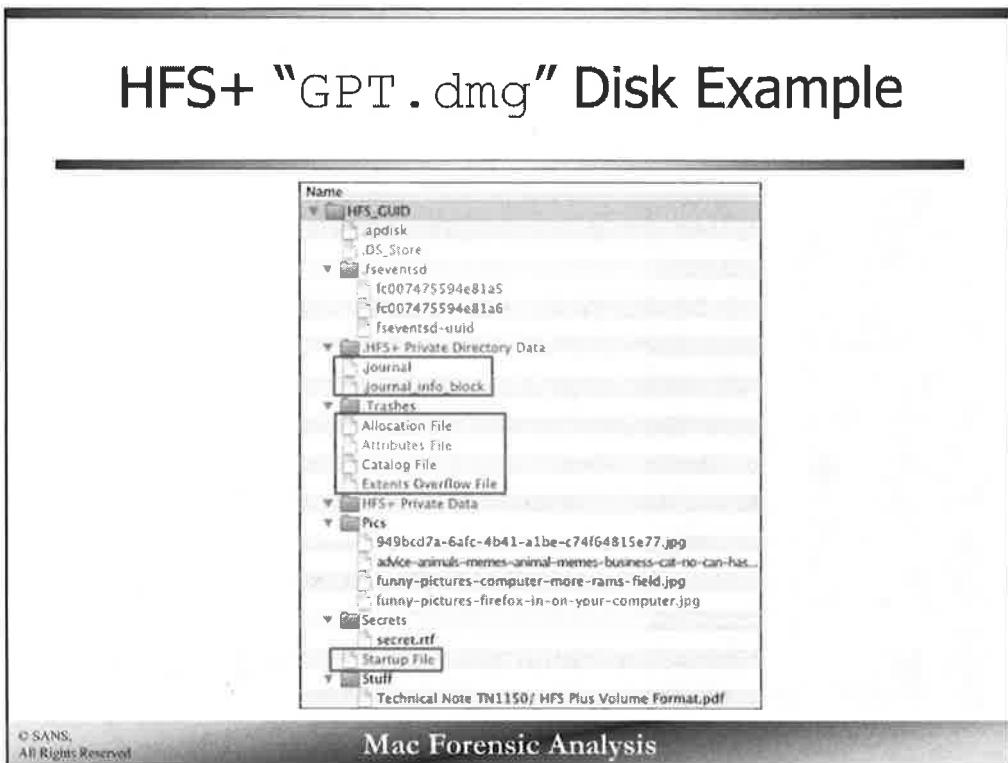
Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple’s Core by Jonathan Levin – Chapter 16

HFS+ “GPT.dmg” Disk Example



The screenshot shows our example disk image file, GPT.dmg that we will use in class.

The GPT.dmg disk was opened in the forensic software Blacklight to show all the files on the disk.

Highlighted in the boxes are the HFS+ Special Files and the Journal files:

- Allocation File
- Attributes File
- Catalog File
- Extents Overflow File
- Startup File
- Journal Files (.journal and .journal_info_block)

These files are not normally accessible by the user, hence they are greyed out in the screenshot above.

We will be looking at these files in detail in the next few sections.

This disk contains a volume (partition) named "HFS_GUID" that contains three user-created directories containing files; Pics, Secrets, and Stuff.

Volume Header

Contains data such as:

- Signature (H+)
- HFS+ Version
- Volume Creation, Modified, Backup & Checked Dates
- File & Folder Count
- Block Size
- Finder Information
- Location of other HFS+ Files

Located 1024 bytes from beginning of the volume

Alternate Volume Header located 1024 bytes from the end of the volume

512 bytes in length

© SANS
All Rights Reserved

Mac Forensic Analysis

The volume header is located 1024 bytes from the beginning of the volume, while the alternate volume header is located 1024 bytes from the end of the volume. The volume header itself is 512 bytes in size.

The volume header contains information about the volume and the locations of each of the HFS+ “special files”.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple’s Core by Jonathan Levin – Chapter 16

Volume Header Example

The screenshot shows a hex dump of a Volume Header. The top row of the dump includes a 16-column header with addresses from 000 to 504. Below this, the data is organized into several sections:

- Signature:** H+ (at offset 0)
- Version:** 2 (at offset 2)
- Attributes:** 4 (at offset 4)
- Last Mounted Version:** 4 (at offset 8)
- Journal Info Block:** 4 (at offset 12)
- Create Date:** 4 (at offset 16)
- Modify Date:** 4 (at offset 20)
- Backup Date:** 4 (at offset 24)
- Checked Date:** 4 (at offset 28)
- File Count:** 4 (at offset 32)
- Folder Count:** 4 (at offset 36)
- Block Size:** 4 (at offset 40)
- Total Blocks:** 4 (at offset 44)
- Free Blocks:** 4 (at offset 48)
- Next Allocation:** 4 (at offset 52)
- rsrc Clump Size:** 4 (at offset 56)
- Data Clump Size:** 4 (at offset 60)
- Next Catalog ID:** 4 (at offset 64)
- Write Count:** 4 (at offset 68)
- Encoding Bitmap:** 8 (at offset 72)
- Finder Info Array [0]:** 4 (at offset 80)
- Finder Info Array [1]:** 4 (at offset 84)
- Finder Info Array [2]:** 4 (at offset 88)
- Finder Info Array [3]:** 4 (at offset 92)
- Finder Info Array [4]:** 4 (at offset 96)
- Finder Info Array [5]:** 4 (at offset 100)
- Finder Info Array [6]:** 4 (at offset 104)
- Finder Info Array [7]:** 4 (at offset 108)
- Allocation File Size & Location:** 80 (at offset 112)
- Extents File Size & Location:** 80 (at offset 192)
- Catalog File Size & Location:** 80 (at offset 272)
- Attributes File Size & Location:** 80 (at offset 352)
- Startup File Size & Location:** 80 (at offset 432)

Below the dump, a copyright notice reads: © SANS. All Rights Reserved.

The volume header contains the following notable fields:

- Offset 0 - The signature for the file system, H+ for HFS+. You may also see HX, for HFS+ with case sensitivity, on iOS devices.
- Offset 16 – The volume’s creation date. (Stored in local time)
- Offset 20 – Last modification date of the volume. (Stored in GMT)
- Offset 32 – Number of files on the volume.
- Offset 36 – Number of folders on the volume.
- Offset 40 – Allocation Block Size (4096 bytes)
- Offset 112 – Location and size of Allocation File.
- Offset 192 – Location and size of Extents Overflow File.
- Offset 272 – Location and size of Catalog File.
- Offset 352 – Location and size of Attributes File.
- Offset 432 – Location and size of Startup File.

Note: HFS+ data uses big-endian

Note: Only the Volume Creation Date stored as local time, all other dates in the volume header are stored as GMT.

This screenshot was created with the Synalyze It! Application.

Offset	Size (in bytes)	Data
0	2	Signature
2	2	Version
4	4	Attributes
8	4	Last Mounted Version
12	4	Journal Info Block
16	4	Create Date
20	4	Modify Date
24	4	Backup Date
28	4	Checked Date
32	4	File Count
36	4	Folder Count
40	4	Block Size
44	4	Total Blocks
48	4	Free Blocks
52	4	Next Allocation
56	4	rsrc Clump Size
60	4	Data Clump Size
64	4	Next Catalog ID
68	4	Write Count
72	8	Encoding Bitmap
80	4	Finder Info Array [0]
84	4	Finder Info Array [1]
88	4	Finder Info Array [2]
92	4	Finder Info Array [3]
96	4	Finder Info Array [4]
100	4	Finder Info Array [5]
104	4	Finder Info Array [6]
108	4	Finder Info Array [7]
112	80	Allocation File Size & Location
192	80	Extents File Size & Location
272	80	Catalog File Size & Location
352	80	Attributes File Size & Location
432	80	Startup File Size & Location

The “size and location” field of each of the HFS+ Special Files have the format shown in the table. Each entry consists of a logical size, a clump size, total number of blocks and an array containing the size and length of each extent of the file. If there are more than eight extents, it will be located in the Extents Overflow File.

Offset	Size (in bytes)	Data
0	8	Logical Size
8	4	Clump Size
12	4	Total Blocks
16	4	Extent 1 – Start Block
20	4	Extent 1 – Block Count
24	4	Extent 2 – Start Block
28	4	Extent 2 – Block Count
32	4	Extent 3 – Start Block
36	4	Extent 3 – Block Count
40	4	Extent 4 – Start Block
44	4	Extent 4 – Block Count
48	4	Extent 5 – Start Block
52	4	Extent 5 – Block Count
56	4	Extent 6 – Start Block
60	4	Extent 6 – Block Count
64	4	Extent 7 – Start Block
68	4	Extent 7 – Block Count
72	4	Extent 8 – Start Block
76	4	Extent 8 – Block Count

HFS+ Dates & Times

The dates and times in HFS+ are unique to HFS. This value represents the number of seconds since 1/1/1904 at midnight or 00:00:00.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	
000	48	2B	00	04	80	00	21	00	48	46	53	4A	00	00	00	02	CC	1F	A2	C6	CE	B8	21	29	
024	00	00	00	00	CC	1F	DB	06	00	00	00	0D	00	00	00	07	00	00	10	00	00	00	27	0B	
048	00	00	24	B3	00	00	01	24	00	01	00	00	00	01	00	00	00	00	00	00	00	3E	00	00	39
072	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
096	00	00	00	00	00	00	00	00	67	9C	B9	05	56	5C	F3	D2	00	00	00	00	00	10	00		
120	00	00	10	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00	00	00	00	00	00	
144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
168	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
192	00	00	00	00	00	04	E0	00	00	04	E0	00	00	00	4E	00	00	00	83	00	00	00	4E	00	
216	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	4E	00	00	00	00	00	00	00	00	
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
264	00	00	00	00	00	00	00	00	00	00	00	00	00	04	E0	00	00	00	04	E0	00	00	00	4E	
288	00	00	04	2B	00	00	00	4E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
312	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04	E0	
360	00	04	E0	00	00	00	00	4E	00	00	00	D1	00	00	4E	00	00	00	00	00	00	00	00	00	
384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
408	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
456	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
504	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

H+

! .HFSJ . . . ï . t k ï , !)

... \$ 3 . . . \$. . . > . . . 9

. . . ï Ü

. . . g . 1 . W 6 0

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

. . . à . . à . . N

Pos...	Off...	Element	Value
0	0	▼HFS+ Volume Header [0]	
0	0	Disk Signature	H+
2	+2	Version	4
4	+4	Attributes	80 00 21 00
8	+8	Last Mounted Version	HFSJ
12	+12	Journal Info Block	2
16	+16	Create Date	3424625350
20	+20	Modify Date	3468173609
24	+24	Backup Date	0
28	+28	Checked Date	3424639750
32	+32	File Count	13
36	+36	Folder Count	7
40	+40	Block Size	4096
44	+44	Total Blocks	9995
48	+48	Free Blocks	9395
52	+52	Next Allocation	292
56	+56	RSRC Clump Size	65536
60	+60	Data Clump Size	65536
64	+64	Next Catalog ID	62
68	+68	Write Count	57
72	+72	Encoding Bitmap	00 00 00 00 00 00 00 01
80	+80	Finder Info Array [0]	0
84	+84	Finder Info Array [1]	0
88	+88	Finder Info Array [2]	0
92	+92	Finder Info Array [3]	0
96	+96	Finder Info Array [4]	0
100	+100	Finder Info Array [5]	0
104	+104	VSDB Volume ID Finder Info Array [6,7]	0x679CB905565CF3D2
112	+112	► ForkData [0]	
192	+192	► ForkData [1]	
272	+272	► ForkData [2]	
352	+352	► ForkData [3]	
432	+432	► ForkData [4]	

Parse Volume Header with hdiutil fsid *.dmg or /dev/disk*

```
Analyzing partition 4: disk image Apple_HFS
Volume size          0x02700000 (40939536) bytes [39.8 MB]
Min stretch size    0x00007000 (8589600) bytes [0.5 MB]
Max stretch size    0x00008000 (134217728) bytes [128 MB]
Current free space  0x02483800 (39481920) bytes [36.7 MB]
Allocation blocks   0x00002700 (9935)
Block size           0x000001000 (49961 bytes) [4 KB]
Post-e1-block space 0x00000000 (0) sectors
VH (size 0x00000002)
  creationTime      0x00000000
  version            0x000004
  attributes          0x000002100
  lastMountedVersion 0x000005340 (HP53)
  journalInfoBlock   0x00000002
  createDate         0xCCCCA2C6 7/8/12, 8:49:10 PM EDT
  modifyDate          0xCCCCB2120 11/24/13, 9:33:39 PM GMT
  backupDate          0x0000000000000000 1/1/04, 12:00:00 AM GMT
  changeDate          0xCCCC1F000 7/9/12, 12:49:10 AM GMT
  fileCount           0x0000000D
  folderCount         0x00000007
  blockSize           0x000001000
  totalBlocks         0x000002700
  freeBlocks          0x000002403
  nextAllocation      0x0000000124
  rsrcClumpSize       0x00001000
  dataClumpSize       0x00001000
  maxCatalogIO        0x0000000FC
  writeCount          0x000000039
  encodingBitMap      0x0000000000000001
  finderInfo
    0                0     Blessed folder directory ID
    1                0
    2                0     Open folder directory ID
    3                0     Mac OS 9 blessed folder directory ID
    4                0
    5                0     Mac OS X blessed folder directory ID
  VSDH Volume ID      0x679CB985565CF302
AllocationFile
  logicalSize         0x00000000000001000
  clumpSize           0x00001000
  totalBlocks         0x00000001
  extents
    startBlock        0x00000001
    blockCount        0x00000001
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

An easy way to view the volume header is to use the `hdiutil fsid` command on a DMG file or on a disk using `/dev/disk*`. This is the volume header information for `GPT.dmg`.

This command must be run with a `*.dmg` file, if you have a raw DD file you can add a `.dmg` extension.

Due to its verbosity, only a section of this output is shown.

Reference:

[hdiutil Man Page](#)

```

Analyzing partition 4: disk image Apple_HFS
HFS+
volume size          0x0270B000 (40939520) bytes [39.0 MB]
min stretch size    0x0087D000 (8900608) bytes [8.5 MB]
max stretch size    0x08000000 (134217728) bytes [128 MB]
current free space 0x024B3000 (38481920) bytes [36.7 MB]
allocation blocks   0x0000270B (9995)
block size           0x00001000 (4096) bytes [4 KB]
post-al-block space 0x00000000 (0) sectors
VH (sector 2)
  signature          H+
  version             0x0004
  attributes          0x80002100
  lastMountedVersion 0x4846534A (HFSJ)
  journalInfoBlock   0x00000002
  createDate          0xCC1FA2C6 7/8/12, 8:49:10 PM EDT
  modifyDate          0xCEB82129 11/24/13, 9:33:29 PM GMT
  backupDate          0x00000000 1/1/04, 12:00:00 AM GMT
  checkedDate         0xCC1FDB06 7/9/12, 12:49:10 AM GMT
  fileCount           0x0000000D
  folderCount         0x00000007
  blockSize            0x00001000
  totalBlocks          0x0000270B
  freeBlocks           0x000024B3
  nextAllocation      0x00000124
  rsrcClumpSize       0x00010000
  dataClumpSize        0x00010000
  nextCatalogID       0x0000003E
  writeCount           0x00000039
  encodingsBitmap     0x0000000000000001
  finderInfo
    0                 0      Blessed folder directory ID
    1                 0
    2                 0      Open folder directory ID
    3                 0      Mac OS 9 blessed folder directory ID
    4                 0
    5                 0      Mac OS X blessed folder directory ID
  VSDB Volume ID      0x679CB905565CF3D2
  allocationFile
    logicalSize        0x0000000000001000
    clumpSize           0x00001000
    totalBlocks         0x00000001
    extents
      startBlock blockCount
      0x00000001 0x00000001
  extentsFile
    logicalSize        0x000000000004E000
    clumpSize           0x0004E000
    totalBlocks         0x0000004E
    extents
      startBlock blockCount
      0x00000083 0x0000004E
  catalogFile
    logicalSize        0x000000000004E000
    clumpSize           0x0004E000
    totalBlocks         0x0000004E
    extents
      startBlock blockCount
      0x00000428 0x0000004E
  attributesFile
    logicalSize        0x000000000004E000
    clumpSize           0x0004E000
    totalBlocks         0x0000004E
    extents
      startBlock blockCount
      0x000000D1 0x0000004E
  startupFile
    logicalSize        0x0000000000000000
    clumpSize           0x00000000
    totalBlocks         0x00000000
    extents
      startBlock blockCount

```

Volume Header

The Sleuth Kit fsstat Command

```
nibble:Exercise 1.3 - HFS+ compass fsstat ~o 40 GPT.dmg
FILE SYSTEM INFORMATION
-----
File System Type: HFS+
File System Version: HFS+
Volume Name: HFS_GUID
Volume Identifier: 679cb905565cf3d2
Last Mounted By: Mac OS X, Journalized
Volume Unmounted Properly
Mount Count: 57
Creation Date: 2012-07-08 21:49:10 (EDT)
Last Written Date: 2013-11-24 16:33:29 (EST)
Last Backup Date: 0000-00-00 00:00:00 (UTC)
Last Checked Date: 2012-07-08 20:49:10 (EDT)
Journal Info Block: 2
METADATA INFORMATION
-----
Range: 2 - 61
Bootable Folder ID: 0
Startup App ID: 0
Startup Open Folder ID: 0
Mac OS 8/9 Blessed System Folder ID: 0
Mac OS X Blessed System Folder ID: 0
Number of files: 13
Number of folders: 7
CONTENT INFORMATION
-----
Block Range: 0 - 9994
Allocation Block Size: 4096
Number of Free Blocks: 9395
```

© SANS,
All Rights Reserved

Mac Forensic Analysis

Another way we can view some of the volume header information is to use The Sleuth Kit fsstat command. While this does not contain the size and location of the HFS+ special files – it does give us a concise view of the volume including timestamps, volume name, format and file/folder information.

This command can be used on a disk image or on a live system using the raw disk device (`/dev/rdisk#`)

Reference:

[fsstat Man Page](#)

nibble:Exercise 1.3 - HFS+ oompa\$ fsstat -o 40 GPT.dmg
FILE SYSTEM INFORMATION

File System Type: HFS+
File System Version: HFS+

Volume Name: HFS_GUID
Volume Identifier: 679cb905565cf3d2

Last Mounted By: Mac OS X, Journaled
Volume Unmounted Properly
Mount Count: 57

Creation Date: 2012-07-08 21:49:10 (EDT)
Last Written Date: 2013-11-24 16:33:29 (EST)
Last Backup Date: 0000-00-00 00:00:00 (UTC)
Last Checked Date: 2012-07-08 20:49:10 (EDT)

Journal Info Block: 2

METADATA INFORMATION

Range: 2 - 61
Bootable Folder ID: 0
Startup App ID: 0
Startup Open Folder ID: 0
Mac OS 8/9 Blessed System Folder ID: 0
Mac OS X Blessed System Folder ID: 0
Number of files: 13
Number of folders: 7

CONTENT INFORMATION

Block Range: 0 - 9994
Allocation Block Size: 4096
Number of Free Blocks: 9395

Extraction of HFS+ “special files” with TSK icat Command

```
icat -f hfs -o <partitionoffset>
*.dmg <inode> > special_file
```

Special File Inode Numbers

- 3 – Extents Overflow File
- 4 – Catalog File
- 6 – Allocation File
- 7 – Startup File
- 8 – Attributes File

© SANS,
All Rights Reserved

Mac Forensic Analysis

The following slides describe the special files and how they are laid out. Each of these special files has their own “inode” or Catalog Node ID (CNID). Apple has reserved the first 16 CNIDs for special files, some of which are listed above.

Some forensic tools do not allow access to these files as they are “system” files. Each special file can be accessed and extracted by their CNID using the `icat` command from Sleuthkit. Extraction of these files may be useful if you need to do additional analysis on them.

To extract the Allocation File you would use the following command:

```
icat -f hfs -o 409640 lion_macbook.dmg 6 > allocation_file
```

Allocation File

Bitmap File

Keeps track of block allocation

1 bit per allocation block (4096 bytes, 8 - 512 byte blocks)

Size of Allocation File dependent on volume size

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Allocation File is used to track which blocks on a volume are allocated (or not). The size of this file is dependent of the size of the volume.

The Allocation File uses a bitmap format to store its data. Each 512 byte block is represented by one bit. The file tracks eight blocks at a time, or 4,096 bytes.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

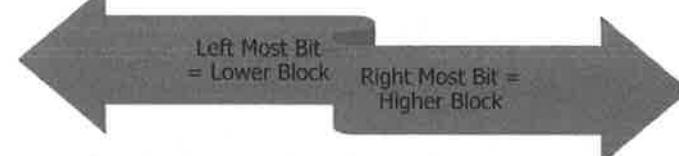
Allocation File Example

© SANS.
All Rights Reserved

Mac Forensic Analysis

In the Allocation File example above, the highlighted blocks contain the tracking status for the specific eight blocks that are assigned to that byte.

Allocation File Allocated Blocks



Hex	Binary	Allocation
0x00	00000000	No Blocks Allocated
0xFF	11111111	All Blocks Allocated
0x1F	00011111	Lowest three blocks are unallocated
0x80	10000000	Lowest block is allocated
0x07	00000111	Highest three blocks are allocated
0xF0	11110000	Highest four blocks are unallocated

© SANS.
All Rights Reserved

Mac Forensic Analysis

Each byte in the Allocation File is organized as follows:

- The **left most bit** contains the status of the block with the **lowest** number.
- The **right most bit** contains the status of the block with the **highest** number.

In the table above, the examples from the previous screenshot are shown in binary to demonstrate which blocks are allocated and which are not.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Allocation File Allocation Block Examples

0xF0

Lower Block								Higher Block
Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	
1	1	1	1	0	0	0	0	
Allocated	Allocated	Allocated	Allocated	Unallocated	Unallocated	Unallocated	Unallocated	

0x07

Lower Block								Higher Block
Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	
0	0	0	0	0	1	1	1	
Unallocated	Unallocated	Unallocated	Unallocated	Unallocated	Allocated	Allocated	Allocated	

© SANS.
All Rights Reserved.

Mac Forensic Analysis

Each byte in the Allocation File is organized as follows:

- The most significant bit contains the status of the block with the lowest number.
- The least significant bit contains the status of the block with the highest number.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

B-Trees

Used by the Catalog, Attributes, Extents Overflow File

Used for efficiency in searching stored data

Made up of Nodes, Records, Keys & Data

Ordered Keys

The balanced tree or B-tree structure is used by the three HFS+ special files.

- Catalog File
- Attributes File
- Extents Overflow File

The B-tree structure is used to efficiently search the records in the tree where file system data is stored. A B-tree is made up of nodes, records, keys, and data. The data is ordered by keys.

This B-tree concept is not very easy to understand and takes time and patience to grasp the structure of B-trees and their purpose in the HFS+ file system.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

B-Tree Structure

Node

Each node...

Records

...contains **multiple** records...

Key & Data

...and each record contains **one** key and related data.

© SANS.
All Rights Reserved

Mac Forensic Analysis

The B-tree structure is made up of nodes, records, keys, and data.

Each **node** may contain **multiple** records.

Each **record** contains only **one key and data**.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

B-Tree Nodes [1]

Header Node

- Only **one** in each B-Tree
- Always the first node
- Contains information where to find other nodes in the tree

Map Nodes

- Contain Map Records: Contains allocation data B-tree nodes

Index Nodes

- Contain Pointer Records: Contains B-tree structure information

Leaf Nodes

- Contain Data Records: Contains data corresponding to a unique key

© SANS,
All Rights Reserved

Mac Forensic Analysis

A B-tree uses four types of nodes:

- Header Node
- Map Nodes
- Index Nodes
- Leaf Nodes

There is only one Header Node in each B-tree. There is **only one** Header Node for each of the Catalog, Attributes, and Extents Overflow files. The Header Node is always the first node in the B-tree, it contains information on how to find other nodes in the B-tree.

Map Nodes contain **Map Records**. These records contain allocation data of B-tree nodes.

Index Nodes contain **Pointer Records**. These records contain B-tree structure information.

Leaf Nodes contain **Data Records**. These records contain data corresponding to a unique key.

Each node type has a standard structure that is the same across all nodes.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

B-Tree Nodes [2]

Each node is assigned a number

Calculate Node Number

- Divide offset in file by node size (8k, 4k, 1k)

Each Node Contains:

- Descriptor (Beginning of Node) – 14 bytes
- List of Records
- List of Record Offsets (End of Node)

Each B-tree node is assigned a number. This number can be calculated by dividing the offset of the node in the file (from the beginning of the special file) by the node size, found in the Header Node's Header Record.

Default Node Size for Mac OS X:

- Catalog File – 8,192 bytes
- Attributes File – 4,096 bytes
- Extents Overflow File – 1,024 bytes

Each node contains the following pieces:

- Descriptor – starts at the beginning of the node.
- List of Record Offsets – starting from the end of the node.
- List of Records

References:

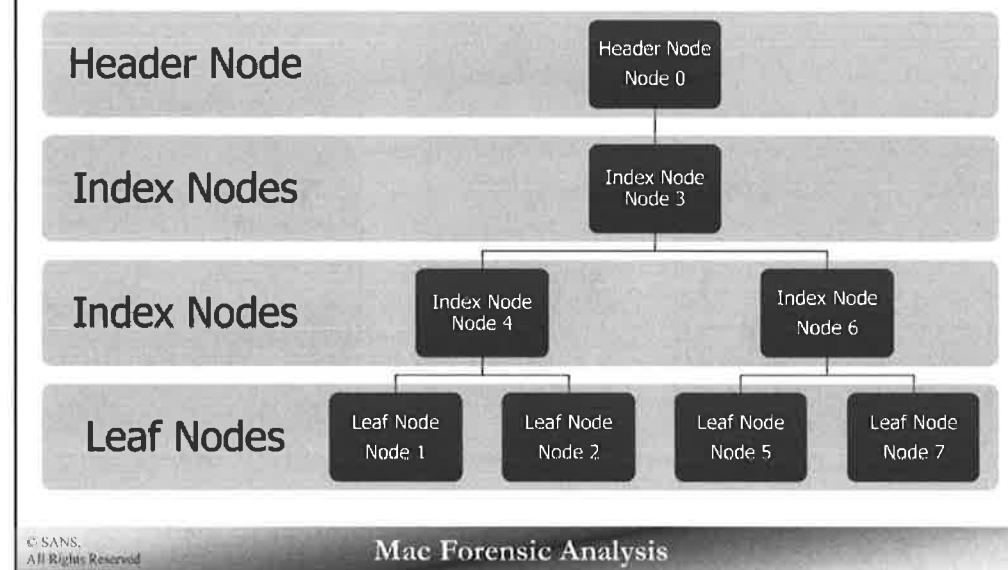
Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

B-Tree Example



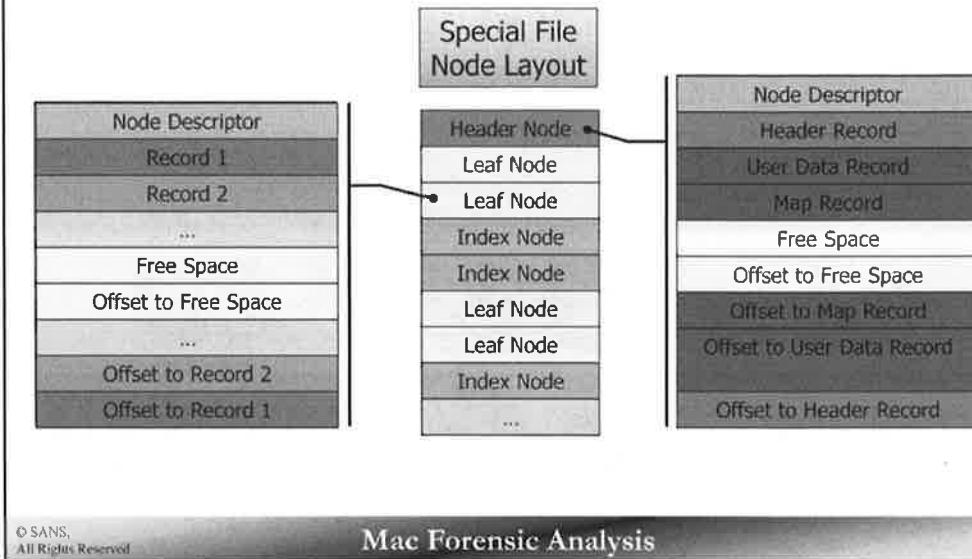
© SANS.
All Rights Reserved

Mac Forensic Analysis

A B-Tree is made up of different nodes, the Header Node at the top is always Node 0.

Index Nodes build the branches of the tree, while Leaf Nodes are the leaves. Leaf Nodes are always at the “bottom” of the tree.

Special File Nodes & Node Structure



Each special file using a B-Tree (Catalog, Attributes, Extents Overflow) will be comprised of ONE Header Node, and multiple Leaf, Index, (and possibly) Map Nodes. Each node size is determined by data found in the Header Node (Node Size), this may be 4,096 bytes for each node, for example.

Each Header Node is comprised of a Node Descriptor, Header Record, User Data Record, and Map Record.

Other Nodes (Leaf, Index) are comprised of a Node Descriptor, and Keyed Records.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Node Descriptor [14 Bytes]

- First 14 Bytes of Each Node
- Node Information
- Forward & backward Links (to other nodes)

Offset	Size	Field	Value	Node Type
0	4 Bytes	Forward Link	-1 (0xFF)	Leaf Node
4	4 Bytes	Backward Link	0 (0x00)	Index Node
8	1 Byte	Kind	1 (0x01)	Header Node
9	1 Byte	Height	2 (0x02)	Map Node
10	2 Bytes	Number of Records		
12	2 Bytes	Reserved		

© SANS,
All Rights Reserved

Mac Forensic Analysis

The Node Descriptor part of the Node starts at the beginning of the node.

The Node Descriptor is always 14 bytes in size, and contains node information such as the forward and backward links to other nodes. These links tell the system how to find the nodes. If the forward link is 0, it is the last node. If the backward link is 0, it is the first node in the list.

The Node Descriptor also contains a value describing the kind of node it is. This will help determine what type of data is contained in the node.

The height field contains the depth in the B-tree of where this node is located. A node at level 0, may be the Header Node or a Map Node. Leaf Nodes will always be at a level 1.

The Number of Records field determines the number of records contained in this node.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Node Descriptor Examples

The screenshot shows a hex dump of memory starting at address 000000. A callout arrow points from the first four bytes (00 00 00 00) to a box labeled "Example 1". Another callout arrow points from the last four bytes (00 00 00 00) to a box labeled "Example 2".

Example 1:
00 00 00 00 | 00 00 00 00 | 01 00 00 03 | 00 00

Example 2:
00 00 00 90 | 00 00 00 02 | FF 01 00 13 | 00 00

© SANS. All Rights Reserved. Mac Forensic Analysis

Example 1:

- Forward Link - 0x00000000 = 0
- Backward Link - 0x00000000 = 0
- Kind - 0x01 = 1 (Header Node)
- Height - 0x00 = 0
- Number of Records - 0x0003 = 3 Records
- Reserved - 0x0000

Example 2:

- Forward Link - 0x00000000 = 0
- Backward Link - 0x00000002 = 2
- Kind - 0xff = -1 (Leaf Node)
- Height - 0x01 = 1
- Number of Records - 0x0013 = 19 Records
- Reserved - 0x0000

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Header Node

Always Node 0

Contains data about the B-tree

Always contains three records:

- Header Record
- User Data Record
- Map Record

The Header Node is always node 0. It contains information about the specific B-tree.

The Header Node is comprised of three records:

- Header Record
- User Data Record
- Map Record

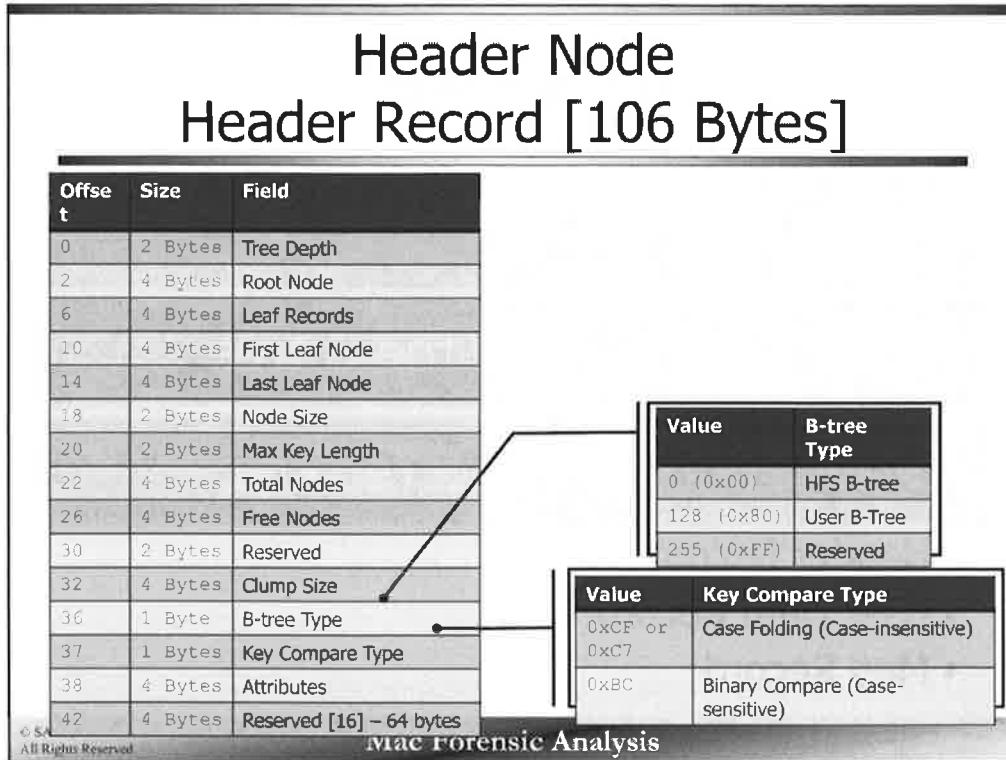
References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16



The Header Record in the Header Node is 106 bytes in size.

The Header Record contains data such:

- The number of the root Index Node (Root Node)
- Node numbers for the first and last leaf nodes
- Size of the nodes
- Total number of nodes in the B-tree
- Number of free nodes
- Type of B-tree
- Comparison type for searching keys
- Attributes specific to the B-tree (detailed in TN1150)

The type of B-tree for HFS+ special files will have the value ‘0’. The comparison type of key searching may have one of three values. The volume may be case-sensitive or case-insensitive.

More detailed information about the record fields can be found in TN1150.

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Header Record Example

© SANS.
All Rights Reserved

Mac Forensic Analysis

Header Record Example:

- Tree Depth – 0x0002 = 2
 - Root Node – 0x00000003 = 3
 - Leaf Records – 0x00000034 = 52
 - First Leaf Node – 0x00000004 = 4
 - Last Leaf Node – 0x00000001 = 1
 - Node Size – 0x1000 = 4096
 - Max Key Length – 0x0204 = 516
 - Total Nodes – 0x0000004E = 78
 - Free Nodes – 0x00000049 = 73
 - Reserved – 0x0000
 - Clump Size – 0x0004E000 = 319488
 - B-Tree Type – 0x00 = 0 = HFS B-Tree
 - Key Compare Type – 0xCF = Case Folding (Insensitive)
 - Attributes – 0x00000006
 - Reserved – 0x00000000 [Array of 16 for 64 bytes total]

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

000000	00	00	00	00	00	00	00	01	00	00	03	00	00	00	02	00	00	00	03	
000020	00	00	00	34	00	00	00	04	00	00	00	01	10	00	02	04	00	00	00	4E
000040	00	00	00	49	00	00	00	04	E0	00	00	00	CF	00	00	00	06	00	00	00
000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Header Node

User Data Record [128 Bytes]

Unused in
Catalog,
Extents
Overflow &
Attributes B-
tree Files

Used in
Hot File B-
tree

000100	30 00 00 00 00 02 00 00 01 20 00 03 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00
000120	00 00
000140	00 00
000160	00 00
000180	00 00
000200	00 00
000220	00 00
000240	00 00
000260	00 00
000280	00 00
000300	00 00

© SANS.
All Rights Reserved

Mac Forensic Analysis

The User Data Record of the Header Node is always 128 bytes in size, however it is unused in the HFS+ special files. It is used in the Hot File B-tree, explained later.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Header Node Map Record

Bitmap – Format
is the same as
Allocation File

Contains data on which nodes are free in the B-tree

Map Record Size:

- Node Size (in Header Record) minus 248 bytes

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Map Record of the Header Node has a very similar function to the Allocation File. Recall that the Allocation File uses a bitmap format to determine which blocks on a volume are used and which are free. The Map Record does the same function but for which nodes are allocated and which are free.

The size of the Map Record is always the Node Size (found in the Header Record) minus 248 bytes.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Map Nodes

Used to store allocation data if Map Record in Header Node is at capacity.

Forward Link in Header Node will be the node number of the first Map Node.

Map Nodes are an extension of those in Header Node.

Contains:

- Node Descriptor
- Map Record (Only one)

Map Nodes are used when the Header Node is at capacity, meaning there is no more space to store data about the allocation of nodes.

The forward link in the Header Node will point to the node number of the first Map Node. Map Nodes extend the Header Node. There may be multiple Map Nodes.

Map Nodes contain a Node Descriptor **and** a Map Record, both have been detailed previously.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Index Node

Points to data in another node

Contains Pointer Records

- Key Length - Always 2 bytes in HFS+ B-trees
- Key - Variable Size
- Data - 4 Bytes Node Number

Catalog File uses the Index Key:

- 4 Byte Parent Catalog Node ID (CNID)
- Variable – 2 Byte Data Size + Data

© SANS,
All Rights Reserved

Mac Forensic Analysis

The Index Node contains Pointer Records. These records contain three parts:

- Key Length
- Key (See next table)
- Node Number (4 bytes) – This node will contain the data for the specified file/folder.

Index Node Layout:	
Size (in bytes)	Field
2	Key Length
Variable	Key (see table below)
4	Node Number

For Example: The Key used in the Catalog File is comprised of the Parent Catalog Node ID (CNID) number and the variable data string (Data size + File/Folder Name)

Index Key for Catalog File:	
Size (in bytes)	Field
4	Parent CNID
Variable	Data Size [2 bytes] + Data (Empty String 0x0000 in thread records) (+padding byte if key length is odd)

Note: HFS+ data uses big-endian

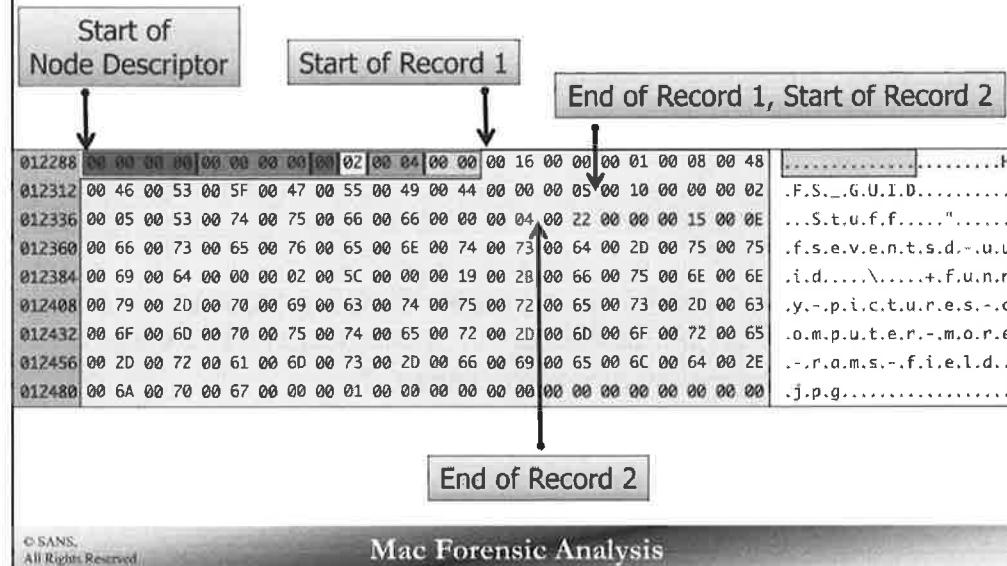
References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html
The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Index Node Example



Index Node Layout:	
Size (in bytes)	Field
2	Key Length
Variable	Key (see table below)
4	Node Number

Index Key for Catalog File:	
Size (in bytes)	Field
4	Parent CNID
Variable	Data Size [2 bytes] + Data (Empty String 0x0000 in thread records) (+padding byte if key length is odd)

Example 1:

- Forward Link - 0x00000000 = 0
- Backward Link - 0x00000000 = 0
- Kind - 0x00 = 0 (Index Node)
- Height - 0x02 = 2
- Number of Records - 0x0004 = 4 Records (HFS_GUID, Stuff, fsevents-d-uuid, funny-pictures-computer-more-rams-field.jpg)
- Reserved - 0x0000

Record 1:

- Key Length - 0x0016 = 22 bytes
- Key -
 - Parent Catalog Node ID (CNID) - 0x00000001 = 1
 - Data:
 - Data Size: 0x0008 = 8 (16 bytes Unicode)
 - Data - 0x004800460053005F0047005500490044 = "HFS_GUID"
- Node Number - 0x00000005 = Node 5

Record 2:

- Key Length – 0x0010 – 16 bytes
- Key –
 - Parent Catalog Node ID (CNID) – 0x00000002 = 2
 - Data:
 - Data Size: 0x0005 = 5 (10 bytes Unicode)
 - Data – 0x00530074007500660066 = “Stuff”
- Node Number – 0x00000004 = Node 4

Note: HFS+ data uses big-endian

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple’s Core by Jonathan Levin – Chapter 16

012288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	48H	
012312	00	46	00	53	00	5F	00	47	00	55	00	49	00	44	00	00	05	00	10	00	00	.F.S.-.G.U.I.D.
012336	00	05	00	53	00	74	00	75	00	66	00	66	00	00	04	00	22	00	00	15	00S.t.u.f.f...."
012360	00	66	00	73	00	65	00	76	00	65	00	6E	00	74	00	73	00	64	00	2D	00	.f.s.e.v.e.n.t.s.d.-.u.u
012384	00	69	00	64	00	00	00	02	00	5C	00	00	00	19	00	2B	00	66	00	75	00	.i.d....\.....+f.u.n.n
012408	00	79	00	2D	00	70	00	69	00	63	00	74	00	75	00	72	00	65	00	73	00	.y.-.p.i.c.t.u.r.e.s.-.c
012432	00	6F	00	6D	00	70	00	75	00	74	00	65	00	72	00	2D	00	6D	00	6F	00	.o.m.p.u.t.e.r.-.m.o.r.e
012456	00	2D	00	72	00	61	00	6D	00	73	00	2D	00	66	00	69	00	65	00	6C	00r.a.m.s.-.f.i.e.l.d..
012480	00	6A	00	70	00	67	00	00	01	00	00	00	00	00	00	00	00	00	00	00	.j.p.g.....	

Leaf Node

Always the bottom of the B-tree

Contains Data Records

- Key Length - Always 2 bytes in HFS+ B-trees
- Key – 4-byte Parent Catalog Node ID (CNID)
- Data – Variable per Special File

May have one-byte padding before and/or after Data to align on two-byte boundary to create an even number of bytes

© SANS,
All Rights Reserved

Mac Forensic Analysis

A Leaf Node contains Data Records. Leaf Nodes are always located at the bottom of the B-tree.

Data Records are comprised of three parts:

- Key Length
- Key
- Data

Leaf Node Layout:	
Size (in bytes)	Field
2	Key Length
4	Key – Parent CNID
4	HFSUniStr255 (Data Size + Data)

The Data in an HFS+ B-tree will contain one of the following depending on the type of special file.

- Catalog Record
- Extent Record
- Attribute Record

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Leaf Node Example

	Start of Node Descriptor	Start of Record 1	End of Record 1, Start of Record 2	
016380	00 2A 00 0E 03 00 00 00 00 00 00 00 00 00 00 10 00 00 00 02 00 05	.*.y... ..		
016405	00 53 00 74 00 75 00 66 00 66 00 01 00 80 00 00 00 01 00 00 00 1B CC 20 80 59	.S.t.u.f.f.....í .Y		
016432	CC 20 82 5D CC 20 84 C9 CC 4C 39 CB 00 00 00 00 00 00 00 01 F5 00 00 00 14 00 00	í .jì .ÉÍL9É.....ó....		
016458	41 ED 00 00 00 01 00	Aí.....		
016484	4F FA D4 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 00 00 02	0úÓI.....0...		
016510	00 15 00 00 00 00 00 00 00 00 48 00 46 00 53 00 2B 00 20 00 50 00 72 00 69H.F.S.+. .P.r.i		
016536	00 76 00 61 00 74 00 65 00 20 00 44 00 b1 00 74 00 G1 00 01 00 10 00 00 00 00	.v.a.t.e. .D.a.t.a.....		
016562	00 00 00 12 CC 1F DB 06 CC 1F DB 06 CC 1F DB 06 CC 1F DB 06 00 00 00 00 00 00í 0.í 0.í 0.í 0.....		
016588	00 00 00 00 00 00 00 02 40 00 00 00 00 01 00 00 00 00 00 00 00 00 50 00 40 00@.....,P.@.		
016614	40 00	@.....j.o		
016640	00 00 00 06 00 00 00 10 00 00 00 04 00 00 00 00 00 02 00 08 00 2E 00 6A 00 6F	.u.r.n.a.l.....		
016666	00 75 00 72 00 6E 00 61 00 6C 00 06 00 00 00 11 00 00 00 04 00 00 00 00 00 00 02			

© SANS,
All Rights Reserved

Mac Forensic Analysis

Example 1 (Node 4, from previous example):

- Forward Link - 0x00000002 = 2
- Backward Link – 0x00000000 5 = 5
- Kind – 0xFF = 0 (Leaf Node)
- Height – 0x01 = 1
- Number of Records – 0x000A = 10 Records
- Reserved – 0x0000

Leaf Node Layout:	
Size (in bytes)	Field
2	Key Length
4	Key – Parent CNID
4	HFSUniStr255 (Data Size + Data)

Record 1:

- Key Length – 0x0010 = 16 bytes
- Key – (Parent Catalog Node ID (CNID)) – 0x00000002 = 2
- Data:
 - Data Size: 0x0005 == 5 (10 bytes Unicode)
 - Data – 0x00530074 007500660066 = “Stuff”
- ...Continue on to Catalog File/Folder Records...

Note: HFS+ data uses big-endian

016380	00	2A	00	0E	00	00	00	02	00	00	00	05	FF	01	00	0A	00	00	00	10	00	00	00	02	00	05		
016406	00	53	00	74	00	75	00	66	00	66	00	01	00	80	00	00	00	01	00	00	00	00	1B	CC	20	80	59	
016432	CC	20	82	5D	CC	20	84	C9	CC	4C	39	CB	00	00	00	00	00	01	F5	00	00	14	00	00	Ì	JÌ	.ÉÍL9È.....ò.....	
016458	41	ED	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Aí.....	
016484	4F	FA	D4	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0úÔI.....	
016510	00	15	00	00	00	00	00	00	00	00	00	00	48	00	46	00	53	00	2B	00	20	00	50	00	72	00	69.....H.F.S.+..P.r.i	
016536	00	76	00	61	00	74	00	65	00	20	00	44	00	61	00	74	00	61	00	01	00	10	00	00	00	00	v.a.t.e..D.a.t.a.....	
016562	00	00	00	12	CC	1F	DB	06	00	00	00	00	00	00	00ìÙ.iÙ.iÙ.iÙ.....												
016588	00	00	00	00	00	00	00	00	00	02	40	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	@.....P.@.	
016614	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	50	00	40	00	00	00	00	00j.o.....	
016640	00	00	00	00	00	00	10	00	00	00	04	00	00	00	00	00	00	02	00	08	00	2E	00	6A	00	5Fu.r.n.a.l.....	
016666	00	75	00	72	00	6E	00	61	00	6C	00	06	00	00	11	00	00	00	04	00	00	00	00	00	02	00	

Catalog File

Directory and file hierarchy of a volume

B-tree File

Catalog Node ID (CNID)

- One for each file & directory
- 4 bytes

First 16 (0-15) CNIDs are reserved by Apple

CNID 0 – Never used

CNIDs are sequentially allocated

© SANS.
All Rights Reserved.

Mac Forensic Analysis

The Catalog File will contain the bulk of the forensic data we are interested in. This file contains the directory and file hierarchy of a volume in a B-tree structure.

Each file and folder on a volume has a unique Catalog Node ID or CNID. This is a 4-byte value that is sequentially allocated. The first 16 CNIDs are reserved by Apple for specific files. CNID 0 is never used. CNIDs may be reused.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Catalog File Reserved Apple CNIDs

CNIDs	Reservation
1	Root Parent
2	Root Folder
3	Extents Overflow File
4	Catalog File
5	Bad Block File
6	Allocation File
7	Startup File
8	Attributes File
14	Repair Catalog File
15	Bogus Extent File
16	First User Catalog Node

© SANS,
All Rights Reserved

Mac Forensic Analysis

This list contains the reserved Catalog Node IDs for the first 16 CNIDs. The assigned file or folder is standard and can always be referenced by that specific CNID.

CNID 16 is always the first CNID available for user files or directories. CNIDs 9-13 do not appear to be used at this time.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Catalog File Key

Used in file, folder, thread records

In searches, the CNID of the parent is compared first, then the Node Name

Size	Field
2 Bytes	Key Length
4 Bytes	Parent CNID
Variable	Node Name (Unicode) - Empty String in Thread Records (0x0000) - Folder/File Name

© SANS,
All Rights Reserved

Mac Forensic Analysis

The key used in the Catalog File is used for each file, folder or thread record and has the format shown in the table above.

Searches will compare the CNID of the parent first, then it will compare the Node Name (File or Folder Name) to determine if a file or folder exists.

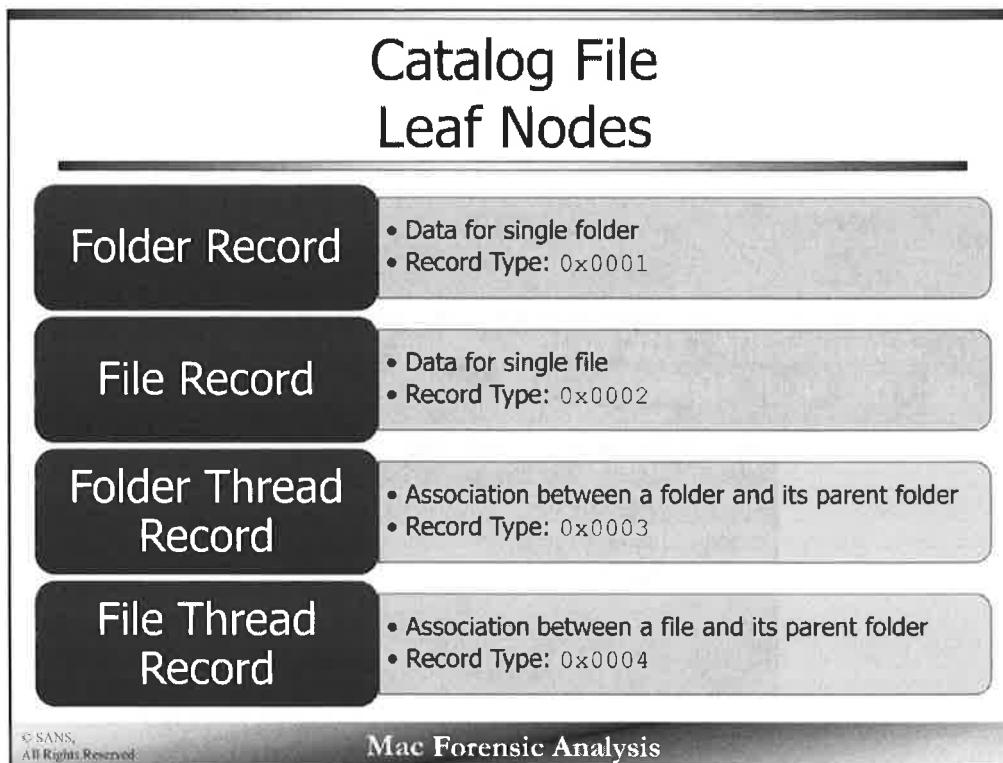
References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16



The Catalog File implements four types of Leaf Nodes.

- Folder Record
- File Record
- Folder Thread Record
- File Thread Record

The Folder Record (Type 0x0001) contains the data for a single folder.

The File Record (Type 0x0002) contains the data for a single file.

The Folder Thread Record (Type 0x0003) contains an association between a folder and its parent folder.

The File Thread Record (Type 0x0004) contains an association between a file and its parent folder.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Catalog File Folder Records [88 Bytes]

Size	Field
2 Bytes	Record Type (0x0001)
2 Bytes	Flags
4 Bytes	Valence
4 Bytes	Folder ID (CNID)
4 Bytes	Create Date
4 Bytes	Content Modification Date
4 Bytes	Attribute Modification Date
4 Bytes	Access Date
4 Bytes	Backup Date
HFSPlusBSDInfo [16 Bytes]	Permissions
FolderInfo [16 Bytes]	User Information
ExtendedFolderInfo [16 Bytes]	Finder Information
4 Bytes	Text Encoding
4 Bytes	Reserved

Size	HFSPlusBSDInfo
4 Bytes	Owner ID
4 Bytes	Group ID
1 Byte	Admin Flags
1 Byte	Owner Flags
2 Bytes	File Mode
4 Bytes	iNode Number or Link Count or Raw Device

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Folder Records of the Catalog File contain the data shown in the left table above. Each Folder Record will have the value 0x0001 in the Record Type field.

The Valence field contains the number of files and directories contained in this folder.
The Folder ID contains the CNID of this folder.

Each Folder Record contains five dates (HFS+ Timestamp 1/1/1904 00:00:00: in GMT):

- Creation Date – Date and time the folder was created
- Content Modification Date – Date and time the content of the folder was modified
- Attribute Modification Date – Date and time the Catalog Record for this folder was changed
- Access Date – Date and time the contents of the folder were accessed
- Backup Date – Date and time the folder was backed up. Likely unused.

The User Information and Finder Information fields contain information specific to the Mac OS Finder such as:

- Location in Finder window
- If the folder has a custom icon or has a colored ‘label’
- If the folder is locked or invisible

The table to the right contains the format for HFS+ file system metadata for the folder.

References:

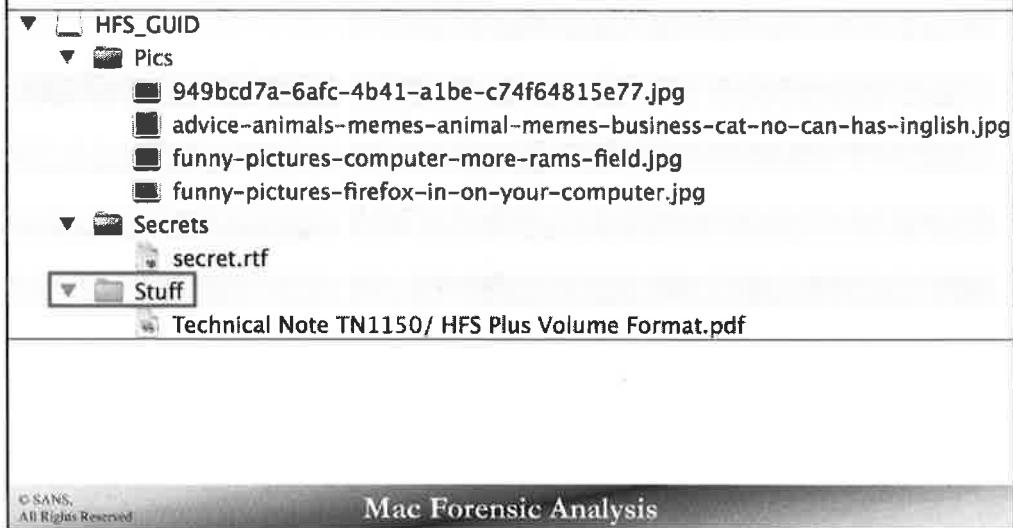
Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

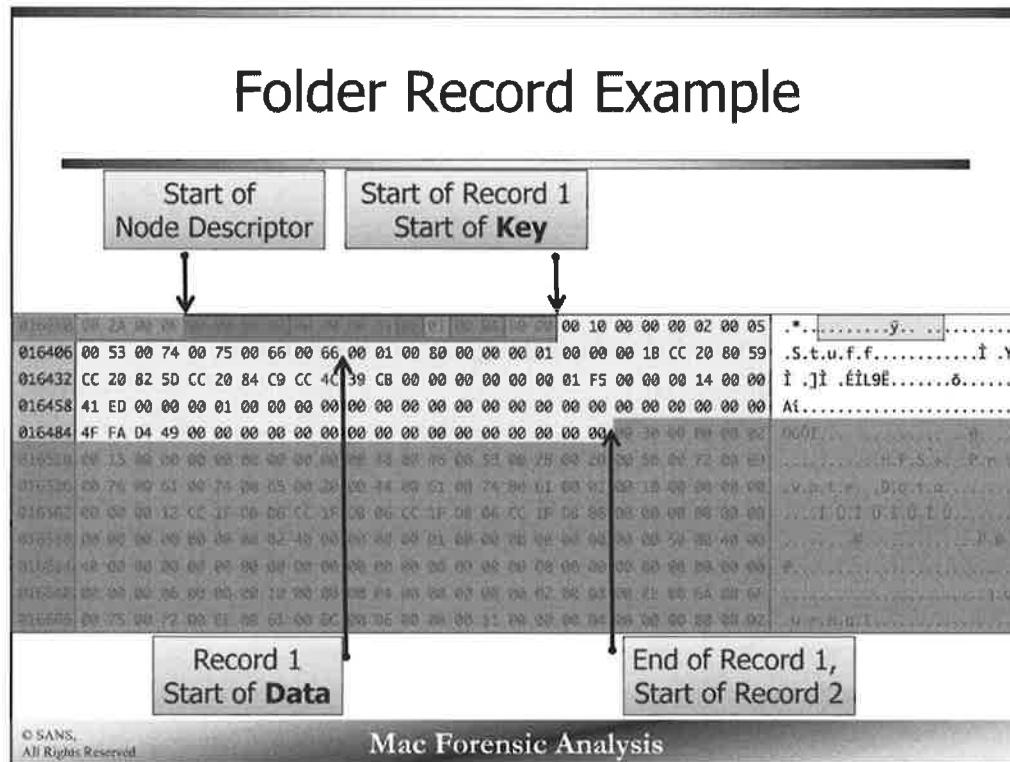
Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Folder Record Example in Finder



In our GPT .dmg example, we are going to be looking at the `Stuff` directory shown in the screenshot above.



Folder Record Example: (Note: HFS+ data uses big-endian)

KEY:

- Key Length – 0x0010 = 16 bytes
- Key – (Parent Catalog Node ID (CNID)) – 0x00000002 = 2
- Data:
 - Data Size: 0x0005 = 5 (10 bytes Unicode)
 - Data – 0x00530074007500660066 = “Stuff”

DATA:

- Record Type – 0x0001 = 1 = Folder Record
- Flags – 0x0080
- Valence – 0x00000001 – 1 (1 File in this folder)
- Folder ID (CNID) – 0x0000001B = 27
- Create Date – 0xCC208059 = 3424682073 = 2012-07-09 12:34:33 Mon UTC
- Content Modification Date – 0xCC20825D = 3424682589 = 2012-07-09 12:43:09 Mon UTC
- Attribute Modification Date – 0xCC2084C9 = 3424683209 = 2012-07-09 12:53:29 Mon UTC
- Access Date – 0xCC4C39CB = 3427547595 = 2012-08-11 16:33:15 Sat UTC
- Backup Date – 0x00000000 = Null Timestamp

...continued on next page...

File Mode Decoding (See tables 15.11-15.13 in File System Forensic Analysis by Brian Carrier):

Example: 0x41ED = 0100000111101101

0100 = 4 = Directory

000 = SetUID, SetGID, Sticky bits

111 = User Permissions = rwx = 7

101 = Group Permissions = r-x = 5

101 = Other Permissions = r-x = 5

016380	00	2A	00	01	3E	00	33	7E	70	05	FF	01	00	0A	00	00	00	10	00	00	00	02	00	05	*	
016406	00	53	00	74	00	75	00	66	00	66	00	01	00	89	00	00	02	01	00	00	00	1B	CC	20	59	
016432	CC	20	82	5D	CC	20	84	C9	CC	4C	39	CB	00	00	00	00	00	00	01	F5	00	00	14	00	1	Y
016458	41	ED	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
016484	4F	FA	D4	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
016510	00	15	00	00	00	00	00	00	00	00	00	48	00	46	00	53	00	28	00	20	00	50	00	72	00	69
016536	00	76	00	61	00	74	00	65	00	20	00	44	00	61	00	74	00	51	00	01	00	10	00	00	00	00
016562	00	00	12	CC	1F	DB	06	CC	1F	DB	06	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
016588	00	00	00	00	00	00	00	02	40	00	00	00	01	00	00	00	00	00	00	50	00	40	00	00	00	00
016614	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	P.Q.
016640	00	00	06	00	00	00	10	00	00	04	00	00	00	00	02	00	08	00	2E	00	6A	00	6F	00	00	00
016666	00	75	00	72	00	6E	00	61	00	6C	00	06	00	00	11	00	32	00	04	82	00	32	00	02	00	00

Folder Record Example Sanity Check

```
nibble:Exercise 1.3 - HFS+ oompa$ istat -o 40 -z GMT GPT.dmg 27
File Path: /Stuff
Catalog Record: 27
Allocated
Type: Folder
Mode: drwxr-xr-x
Size: 0
uid / gid: 501 / 20
Link count: 1

File Name: Stuff
Admin flags: 0
Owner flags: 0
Text encoding: 0 = MacRoman

Times:
Created: 2012-07-09 12:34:33 (GMT)
Content Modified: 2012-07-09 12:43:09 (GMT)
Attributes Modified: 2012-07-09 12:53:29 (GMT)
Accessed: 2012-08-11 16:33:15 (GMT)
Backed Up: 0000-00-00 00:00:00 (UTC)

Attributes:
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

Lets check our results by using the `istat` tool by The Sleuth Kit.

Catalog File File Records [248 Bytes]

Size	Field
2 Bytes	Record Type (0x0002)
2 Bytes	Flags
4 Bytes	Reserved
4 Bytes	File ID (CNID)
4 Bytes	Create Date
4 Bytes	Content Modification Date
4 Bytes	Attribute Modification Date
4 Bytes	Access Date
4 Bytes	Backup Date
HFSPlusBSDInfo [16 Bytes]	Permissions
FileInfo [16 Bytes]	User Information
ExtendedFileInfo [10 Bytes]	Finder Information
4 Bytes	Text Encoding
4 Bytes	Reserved
HFSPlusForkData [80 Bytes]	Data Fork
HFSPlusForkData [80 Bytes]	Resource Fork

Size	HFSPlusBSDInfo
4 Bytes	Owner ID
4 Bytes	Group ID
1 Byte	Admin Flags
1 Byte	Owner Flags
2 Bytes	File Mode
4 Bytes	iNode Number or Link Count or Raw Device

© SA All Rights Reserved

Mac OS Forensic Analysis

The File Records of the Catalog File contain the data shown in the left table above. Each File Record will have the value 0x0002 in the Record Type field. The contents of this record are very similar to the Folder Record.

The File ID contains the CNID of this file.

Each Folder Record contains five dates (HFS+ Timestamp 1/1/1904 00:00:00: in GMT):

- Creation Date – Date and time the file was created
- Content Modification Date – Date and time the content of the file was modified
- Attribute Modification Date – Date and time the Catalog Record for this file was changed
- Access Date – Date and time the contents of the file were accessed
- Backup Date – Date and time the file was backed up. Likely unused.

The User Information and Finder Information fields contain information specific to the Mac OS Finder such as

- Location in Finder window
- If the file is a Bundle or an Alias
- The file's File Type and Creator codes

The table to the right contains the format for HFS+ permissions for the file.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Resource & Data Forks

Data Fork

- Generally holds data

Resource Fork

- Generally holds resources:
 - Custom Icons
 - Executable Code
 - License Information
 - Preferences
 - Pictures
 - Metadata
 - Compressed Files/Data

© SANS,
All Rights Reserved

Mac Forensic Analysis

As you may have noticed in the previous slide, the File Record also contains the information about the Data and Resource Forks. This data includes the location and size of each fork.

Mac OS X may use one or both of the forks to contain a file data and metadata.

References:

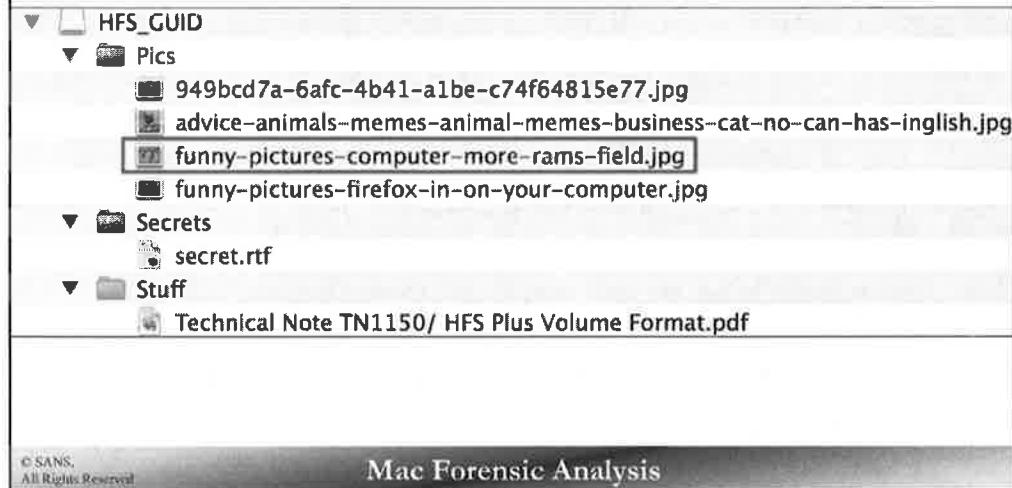
Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

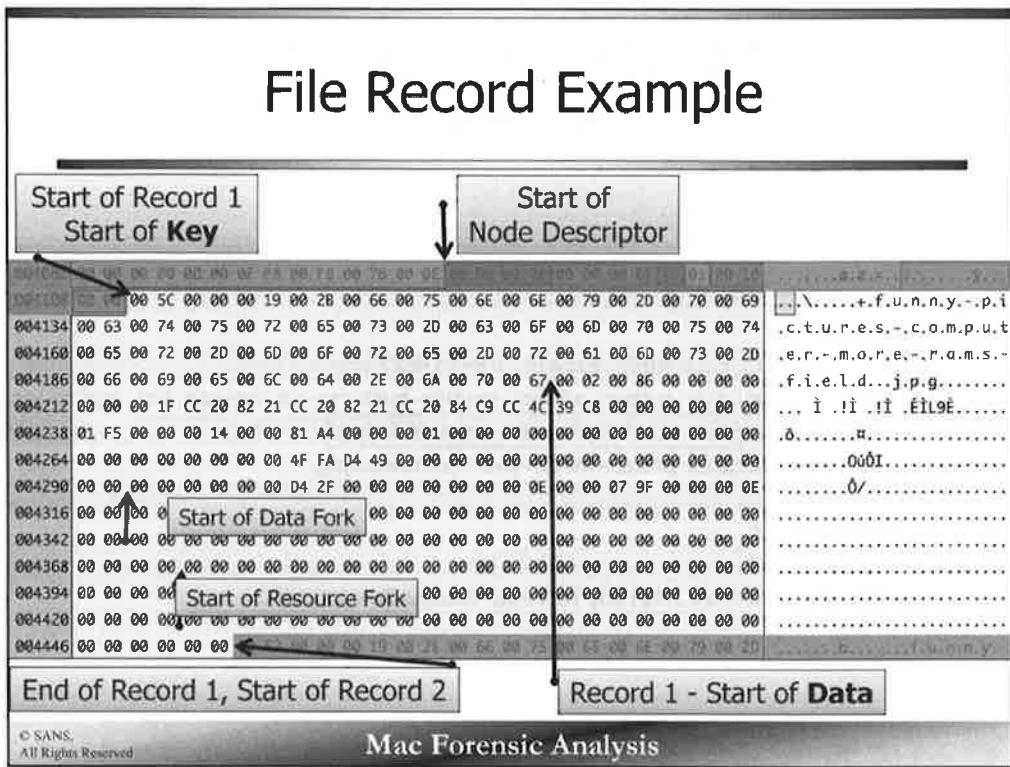
Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

File Record Example in Finder



In our GPT .dmg example, we are going to be looking at the funny-pictures-computer-more-rams-field.jpg file located in the Pics directory shown in the screenshot above.



File Record Example: (Note: HFS+ data uses big-endian)

KEY:

- Key Length – 0x005C = 92 bytes
- Key – (Parent Catalog Node ID (CNID)) – 0x00000019 = 25
- Node Name:
 - Node Size - 0x002B = 43 (86 bytes Unicode)
 - Node Name –
0x00660075006E006E0079002D00700069006300740075007200650073002D0063006F006D00700075007400650072002D006D006F00720065002D00720061006D0073002D006600690065006C0064002E006A00700067 = “funny-pictures-computer-more-rams-field.jpg”

DATA:

- Record Type – 0x0002 = 2 = File Record
- Flags – 0x0086
- Reserved – 0x00000000
- File ID (CNID) – 0x0000001F = 31
- Create Date – 0xCC208221 = 3424682529 = 2012-07-09 12:42:09 Mon UTC
- Content Modification Date – 0xCC208221 = 3424682529 = 2012-07-09 12:42:09 Mon UTC
- Attribute Modification Date – 0xCC2084C9 = 3424683209 = 2012-07-09 12:53:29 Mon UTC
- Access Date – 0xCC4C39C8 = 3427547592 = 2012-08-11 16:33:12 Sat UTC
- Backup Date – 0x00000000 – Null Timestamp

...continued on next page...

File Mode Decoding (See tables 15.11-15.13 in File System Forensic Analysis by Brian Carrier):

Example: $0x81A4 = 1000000110100100$

1000 == 8 == Regular File

000 = SetUID, SetGID, Sticky bits

110 ≡ User Permissions ≡ rw- ≡ 6

100 = Group Permissions = r-- = 4

100 = Other Permissions = r-- = 4

File Record Example Sanity Check

```
nibble:Exercise 1.3 - HFS+ nompa$ istat -o 48 -z GMT GPT.dmg 31
File Path: /Pics/funny-pictures-computer-more-rams-field.jpg
Catalog Record: 31
Allocated
Type: File
Mode: r--r--r--
Size: 54319
uid / gid: 501 / 20
Link count: 1

File Name: funny-pictures-computer-more-rams-field.jpg
Admin flags: 0
Owner flags: 0
Has extended attributes
File type: 0000
File creator: 0000
Text encoding: 0 = MacRoman
Resource fork size: 0

Times:
Created: 2012-07-09 12:42:09 (GMT)
Content Modified: 2012-07-09 12:42:09 (GMT)
Attributes Modified: 2012-07-09 12:53:29 (GMT)
Accessed: 2012-08-11 16:33:12 (GMT)
Backed Up: 0000-00-00 00:00:00 (UTC)

Data Fork Blocks:
1951-1964

Attributes:
Type: EXATTR (4354-2) Name: com.apple.metadata:kMDItemWhereFroms Resident size: 197
Type: EXATTR (4354-3) Name: com.apple.quarantine Resident size: 89
Type: DATA (4352-0) Name: DATA Non-Resident size: 54319 init size: 54319
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

Let's check our results by using the `istat` tool by The Sleuth Kit.

Catalog File

File or Folder Thread Records

Size	Field
2 Bytes	Record Type (0x0003) – Folder Thread Record (0x0004) – File Thread Record
2 Bytes	Reserved
4 Bytes	Parent ID (CNID)
HFSUniStr255	Node Name (File or Folder Name) 2 Bytes – Length <=255 Unicode Name

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Folder and File Thread Records in the Catalog File are used to associate a file or folder to the parent folder's CNID. These records can be remembered as using a thread to sew together a file or folder with its parent directory.

Each File or Folder Thread Record contains:

- The record type will be either 0x0003 or 0x0004 for folder or file, respectively.
- The CNID of the file/folder parent directory.
- The Node Name which is the name of the file or folder in Unicode.

References:

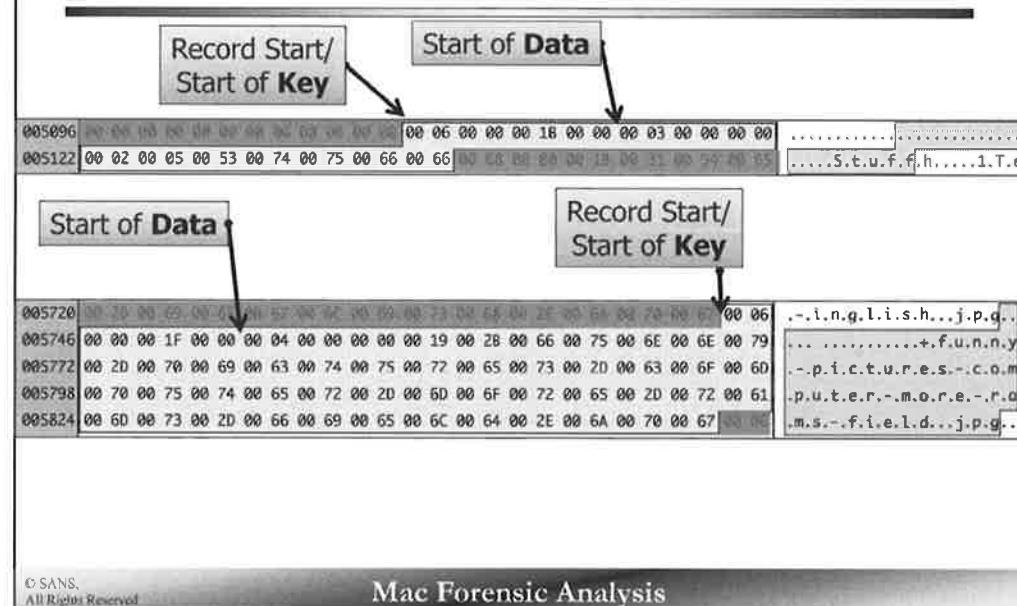
Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Folder & File Thread Record Examples



In this example we will be looking at the folder and file examples that we analyzed previously.

Folder Thread Record Example: (Note: HFS+ data uses big-endian)

Top Example:

KEY:

- Key Length – 0x0006 = 6 bytes
 - Key – (Parent Catalog Node ID (CNID)) – 0x0000001B = 27
 - Data – 0x0000 (Will be “blank” for Thread Records)

DATA:

- Record Type – $0x0003 = 3$ = Folder Thread Record
 - Reserved – $0x0000$
 - Parent CNID – $0x00000002 = 2$
 - Node Name:
 - Node Size – $0x0005 = 5$ (10 Unicode bytes)
 - Node Name – “`Chu-f5`”

Bottom Examples:

KEY

- Key Length – $0x0006$ = 6 bytes
 - Key – (Parent Catalog Node ID (CNID)) – $0x0000001F$ = 31
 - Data – $0x0000$ (Will be “blank” for Thread Records)

DATA:

- Record Type – 0x0004 = 3 = File Thread Record
- Reserved – 0x0000
- Parent CNID – 0x00000019 = 25
- Node Name:
 - Node Size – 0x002B = 43 (86 Unicode bytes)
 - Node Name – “funny-pictures-computer-more-rams-field.jpg”

005096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
005122	00	02	00	05	00	53	00	74	00	75	00	66	00	66	00	68	00	68	00	70	00	71	00	71	00	74	00	75

005720	00	20	20	69	00	35	02	67	00	6C	00	69	00	73	00	69	00	70	00	6A	00	70	00	67	00	6E	00	79
005746	00	00	00	1F	00	00	00	04	00	00	00	00	00	19	00	2B	00	66	00	75	00	6E	00	6E	00	79	00	06
005772	00	20	00	70	00	69	00	63	00	74	00	75	00	72	00	65	00	73	00	20	00	63	00	6F	00	6D	00	6D
005798	00	70	00	75	00	74	00	65	00	72	00	2D	00	6D	00	6F	00	72	00	65	00	2D	00	72	00	61	00	61
005824	00	6D	00	73	00	2D	00	66	00	69	00	65	00	6C	00	64	00	2E	00	6A	00	70	00	67	00	76	00	66

Extents Overflow File

B-tree File

Tracks overflow allocation blocks for a file

Allocation Number, Number of Blocks

First eight blocks in Catalog File Record

Fixed key length

One data record type (Extents)

© SANS,
All Rights Reserved

Mac Forensic Analysis

The Extents Overflow File is another B-tree formatted file. This file tracks the overflow allocation extents that could not fit into the Catalog File Records.

The Catalog File holds the location and size of the first eight extents of a file (eight for each fork), all others will be located in the Extents Overflow File.

The Extents Overflow File uses only one Data Record type.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Extents Overflow File Key & Data Records

Size	Field	Value	Fork Type
2 Bytes	Key Length	0x00	Data Fork
1 Byte	Fork Type	0xFF	Resource Fork
1 Byte	Pad		
4 Bytes	File ID (CNID)		
4 Bytes	Start Block		

Size	Field (For Each Extent)
4 Bytes	Start Block
4 Bytes	Block Count

© SANS.
All Rights Reserved

Mac Forensic Analysis

The Key Record for the Extents Overflow File contains:

- Key Length
- Fork Type - 0x00 for Data or 0xFF for Resource
- Padding
- Catalog Node ID
- Start Block – The block in the file where this extent group starts. (The first eight file extents are in the Catalog File.)

The Data Record will contain extent records containing a Start Block and a Block Count. There may be multiple extents, thus multiple Data Records.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Extents Overflow File Example (...a very fragmented file...)

000	00	00	00	00	00	00	00	12	FF	01	00	2B	00	00	00	0A+....
016	00	00	00	5D	EC	80	00	00	00	00	02	CF	EF	CD	00	00]
032	00	01	02	CF	EF	D0	00	00	00	02	02	CF	EF	D9	00	00
048	00	02	02	CF	EF	DD	00	00	00	01	02	CF	F0	07	00	00
064	00	01	02	CF	F0	1B	00	00	00	01	02	CF	F0	1D	00	00
080	00	01	02	CF	F0	41	00	00	00	03	00	0A	00	00	00	5DA.....]
096	EC	80	00	00	00	19	02	CF	F8	B3	00	00	00	02	02	CF
112	F8	B7	00	00	00	01	02	CF	F8	B9	00	00	00	01	02	CF
128	F8	BB	00	00	00	01	02	CF	F8	BE	00	00	00	01	02	CF
144	F8	CE	00	00	00	01	02	CF	F8	D1	00	00	00	02	02	CF
160	F8	DB	00	00	00	01	00	0A	00	00	00	5D	EC	80	00	00]

© SANS,
All Rights Reserved

Mac Forensic Analysis

Extents Overflow Example: (**Note: HFS+ data uses big-endian**)

(Remember: The first 14 bytes are Node Descriptor) The first record is the highlighted section.

KEY:

- Key Length – 0x000A = 10 bytes
- Fork Type – 0x00 = Data Fork
- Pad – 0x00
- File ID (CNID) – 0x005DEC80 = 6155392
- Start Block – 0x0000000D = 13 (this block count is 13 blocks after the first eight extents in the File Record)

DATA:

- Extent 1 – Start Block – 0x02CFEFCD = 47181773
- Extent 1 – Block Count – 0x00000001 = 1
- Extent 2 – Start Block – 0x02CFEF00 = 47181776
- Extent 2 – Block Count - 0x00000002 = 2
- Extent 3 – Start Block – 0x02CFEF09 = 47181785
- Extent 3 – Block Count – 0x00000002 = 2
- Extent 4 – Start Block – 0x02CFEF007 = 47181789
- Extent 4 – Block Count – 0x00000001 = 1
- Extent 5 – Start Block – 0x02CFF007 = 47181831
- Extent 5 – Block Count – 0x00000001 = 1
- Extent 6 – Start Block – 0x02CFF01B = 47181851
- Extent 6 – Block Count – 0x00000001 = 1
- Extent 7 – Start Block – 0x02CFF01D = 47181853
- Extent 7 – Block Count – 0x00000001 = 1
- Extent 8 – Start Block – 0x02CFF041 = 47181889
- Extent 8 – Block Count – 0x00000003 = 3

000	00	00	00	00	00	00	00	12	FF	01	00	2B	00	00	00	0A+....
016	00	00	00	5D	EC	80	00	00	00	0D	02	CF	EF	CD	00	00]
032	00	01	02	CF	EF	D0	00	00	00	02	02	CF	EF	D9	00	00
048	00	02	02	CF	EF	DD	00	00	00	01	02	CF	F0	07	00	00
064	00	01	02	CF	F0	1B	00	00	00	01	02	CF	F0	1D	00	00
080	00	01	02	CF	F0	41	00	00	00	03	00	0A	00	00	00	5DA.....]
096	EC	80	00	00	00	19	02	CF	F8	B3	00	00	00	02	02	CF
112	F8	B7	00	00	01	02	CF	F8	B9	00	00	00	01	02	CF
128	F8	BB	00	00	00	01	02	CF	F8	BE	00	00	00	01	02	CF
144	F8	CE	00	00	00	01	02	CF	F8	D1	00	00	00	02	02	CF
160	F8	DB	00	00	00	01	00	0A	00	00	5D	EC	80	00	00	00]

Attributes File

- B-tree File
- Variable Key Length
- Three Record Types
 - **Inline Attributes**
 - Fork Data Attributes
 - Extension Attributes

Size	Field
2 bytes	Key Length
2 bytes	Pad
4 bytes	File ID (CNID)
4 bytes	Start Block
2 bytes	Attribute Name Length
Variable	Attribute Name

Value	Record Type
0x00000010	Inline Data Attribute
0x00000020	Fork Data Attribute
0x00000030	Extension Attribute

The Attributes File is yet another B-tree structured file. To show the age of TN1150, the Attributes File section states “*the structure of the keys in the attribute B-tree has not been finalized and is subject to change*”. The layout of the Key for the Attributes files was found in another favorite Mac source, the Mac OS X Internals book by Amit Singh. This book, while published in 2007, still proves to be a helpful (sometimes dated) resource for Mac forensics.

The Key Record for the Attributes File is comprised of the following fields:

- Key Length
- Pad
- File ID (CNID)
- Start Block (unused for Inline Attributes)
- Attribute Name Length
- Attribut Name

This file uses three types of Data Records. While there are three types, recent versions of OS X are using Inline Attributes.

- Inline Data Attributes
- Fork Data Attributes
- Extension Attributes

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Attributes File

Inline Attributes - Data Record

Size	Field
4 bytes	Record Type (0x00000010)
8 bytes	Reserved
4 bytes	Attribute Size
Variable: Stored in single B-tree record	Attribute Data

© SANS.
All Rights Reserved

Mac Forensic Analysis

By far the most populous Data Record in the Attributes File is the Inline Attribute Data Record. TN1150, again showing its age, described this data record as “reserved for future use”.

Discussed later on in the course is Extended Attributes, which is where you will find these type of file metadata. An example of Extended Attributes is the com.apple.quarantine attribute. This attribute contains file quarantine information such as when the file was downloaded, what program was used to download the file, etc.

The Inline Attribute Data Record consists of the following fields:

- Record Type (0x00000010)
- Reserved
- Attribute Size
- Attribute Data

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple’s Core by Jonathan Levin – Chapter 16

Attributes File Examples

Two Extended Attributes

```
nibble:Exercise 1.3 ~ HFS+ compas$ istat -o 40 -z GMT GPT.dmg 31
File Path: /Pics/funny-pictures-computer-more-rams-field.jpg
Catalog Record: 31
Allocated
Type: File
Mode: rrw-r--r--
Size: 54319
uid / gid: 501 / 20
Link count: 1

File Name: funny-pictures-computer-more-rams-field.jpg
Admin flags: 0
Owner flags: 0
Has extended attributes
File type: 0000
File creator: 0000
Text encoding: 0 MacRoman
Resource fork size: 0

Times:
Created: 2012-07-09 12:42:00 (GMT)
Content Modified: 2012-07-09 12:42:09 (GMT)
Attributes Modified: 2012-07-09 12:53:29 (GMT)
Accessed: 2012-08-11 16:33:12 (GMT)
Backed Up: 0000-00-00 00:00:00 (UTC)

Data Fork Blocks:
1991-1964

Attributes:
Type: ExATTR (4354-2) Name: com.apple.metadata:kMDItemWhereFroms Resident size: 197
Type: ExATTR (4354-3) Name: com.apple.quarantine Resident size: 89
Type: DATA (4352-0) Name: DATA Non-Resident size: 54319 init size: 54319
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

We will be looking at the extended attributes from the funny-pictures-computer-more-rams-field.jpg example.

This file has two extended attributes:

- com.apple.metadata:kMDItemWhereFroms
- com.apple.quarantine

Attribute File Example: (Note: HFS+ data uses big-endian)

These extended attributes are from the funny-pictures-computer-more-rams-field.jpg example.

Attribute 1:

KEY:

- Key Length – 0x0054 = 84 bytes
 - Pad – 0x0000
 - File ID (CNID) – 0x00000001F = 31
 - Start Block – 0x00000000 = 0
 - Attribute Name Length – 0x0024 = 36 bytes (72 bytes Unicode)
 - Attribute Name – “com.apple.metadata:kMDItemWhereFroms”

DATA:

009130	0E 60 65 60	00 54 00 00 00 00 00 1F 00 00 00 00 00 00 24 00 63 00 6F 009152 00 6D 00 2E 00 61 00 70 00 70 00 6C 00 65 00 2E 00 6D 00 65 00 74 009174 00 61 00 64 00 61 00 74 00 61 00 3A 00 6B 00 4D 00 44 00 49 00 74 009196 00 65 00 6D 00 57 00 68 00 65 00 72 00 65 00 46 00 72 00 6F 00 6D 009218 00 73 00 00 10 00 00 00 00 00 00 00 00 00 00 00 C5 62 70 6C 69 009240 73 74 30 30 A2 01 02 5F 10 61 68 74 74 70 3A 2F 2F 69 63 61 6E 68 009262 61 73 63 68 65 65 7A 62 75 72 67 65 72 2E 66 69 6C 65 73 2E 77 6F 009284 72 64 70 72 65 73 73 2E 63 6F 60 2F 32 30 38 2F 30 33 2F 66 75 009306 6E 6E 79 2D 70 69 63 74 75 72 65 73 2D 63 6F 6D 70 75 74 65 72 2D 009328 6D 6F 72 65 2D 72 61 6D 73 2D 66 69 65 6C 64 2E 6A 70 67 5F 10 30 009350 68 74 74 70 3A 2F 2F 69 63 61 6E 68 61 73 63 68 65 65 7A 62 75 72 009372 67 65 72 2E 63 6F 6D 2F 70 61 67 65 2F 33 2F 73 3D 63 6F 6D 70 009394 75 74 65 72 08 0B 6F 00 00 00 00 00 01 01 00 00 00 00 00 00 00 00 009416 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A2 70 00 34 00 00
		ome.T.....\$co. .m...a.p.p.l.e...m.e.t .a.d.a.t.a.:k.M.D.I.t .e.m.W.h.e.r.e.F.r.o.m .s.....Abpli st00t._.ahttp://icanh ascheezburger.files.wo rdpress.com/2008/03/fu nny-pictures-computer- more-rams-field.jpg_0 http://icanhascheezbur ger.com/page/3/?s=comp uter. o.....dp.4..

Attributes File Example [2] *BONUS*

com.apple.quarantine

Attribute File Example: (Note: HFS+ data uses big-endian)

These extended attributes are from the funny-pictures-computer-more-rams-field.jpg example.

Attribute 2:

KEY:

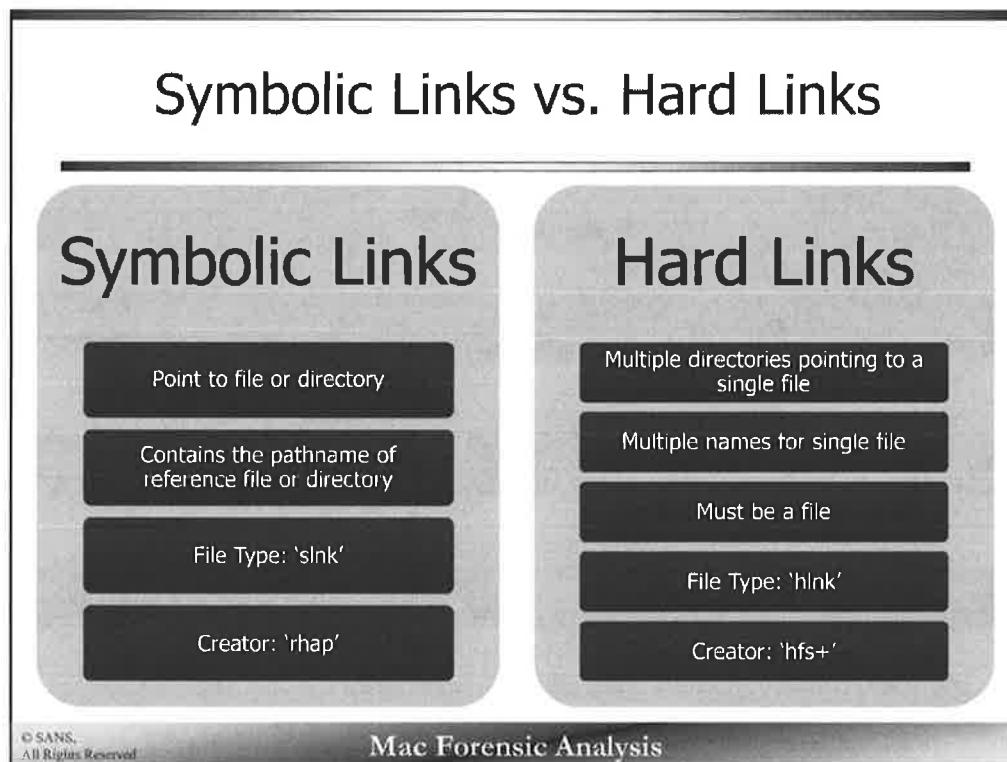
- Key Length – 0x0034 = 52 bytes
 - Pad – 0x0000
 - File ID (CNID) – 0x00000001F = 31
 - Start Block – 0x00000000 = 0
 - Attribute Name Length – 0x0014 = 20 bytes (40 bytes Unicode)
 - Attribute Name – “com.apple.quarantine”

DATA:

- Record Type – 0x00000010 – **Inline Attribute**
 - Reserved – 0x0000000000000000
 - Attribute Size – 0x00000059 = 89 bytes
 - Attribute Data – “0001;4ffad1a4;Google\x20Chrome.app;4EC2C430-4F8C-43AD-A7C7-4A54F2C3C83A|com.google.Chrome”

009416	03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A2 70	00 34 00 00dp.4...
009438	00 00 00 1F 00 00 00 00 00 14 00 63 00 6F 00 6D 00 2E 00 61 00 70c.o.m....a.p	
009460	00 70 00 6C 00 65 00 2E 00 71 00 75 00 61 00 72 00 61 00 6E 00 74	.p.l.e...q.u.a.r.a.n.t	
009482	00 69 00 6E 00 65 00 00 00 10 00 00 00 00 00 00 00 00 59	.i.n.e.....Y	
009504	30 30 30 31 3B 34 66 66 61 64 31 61 34 38 47 6F 6F 67 6C 65 5C 78	0001;4ffad1a4;Google\x	
009526	32 30 43 68 72 6F 6D 65 2E 61 70 70 3B 34 45 43 32 43 34 33 30 2D	20Chrome.app;4EC2C430-	
009548	34 46 38 43 2D 34 33 41 44 2D 41 37 43 37 2D 34 41 35 34 46 32 43	4F8C-43AD-A7C7-4A54F2C	
009570	33 43 38 33 41 7C 63 6F 6D 2E 67 6F 6F 67 6C 65 2E 43 68 72 6F 6D	3C83A com.google.Chrom	
009592	65 6B 00 54 00 00 00 20 00 00 00 00 00 24 00 63 00 6F 00 65	ek.T.....\$.c.o.m	

Symbolic Links vs. Hard Links



Mac OS X uses two types of links, Hard and Symbolic.

Hard links are used to have multiple directories point to a single file, so there can be multiple names for a single file. Hard links can only be used on a file. Hard link files will have a file type of 'hlnk' and a creator code of 'hfs+'.

Symbolic Links are used to point to a file or directory and can be used on a file or a directory. Symbolic links have a file type of 'slnk' and a creator code of 'rhap'. These are shown in the terminal as follows:

```
lrwxr-xr-x@ 1 root wheel      11 Jul 15 22:20 etc -> private/etc
lrwxr-xr-x@ 1 root wheel      11 Jul 15 22:21 tmp -> private/tmp
lrwxr-xr-x@ 1 root wheel      11 Jul 15 22:21 var -> private/var
```

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/TN1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

Hard Links

Indirect Node File

- Traditional UNIX System inode

Hard Link Files

- Refer to indirect node file
- One hard link file for each reference to the file content

Metadata Directory

- Contains indirect node files
- "\x0000\x0000\x0000\x0000HFS+ Private Data/"

Link Reference

- Identification Number, unique across a volume
- iNode# (random number from 100 – 1073741923)

© SANS
All Rights Reserved

Mac Forensic Analysis

Hard links use an Indirect Node File to store the actual file data of the hard link, making it similar to an inode in a Unix-based system.

Hard Link Files are used to point to an Indirect Node File, one for each directory that references the file content.

You may have noticed the strange directory in the root of the volume called “^\^HFS+ Private Data/”, where the caret is a null. This is the Metadata Directory. This directory contains multiple files with the filename “iNode” and some number. These files are the Indirect Node Files. The number after “iNode” is the link reference number. This number is randomly chosen from 100-1073741923. The link reference number is not the same as the files Catalog Node ID number.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16

HFS+ Journal

Journal Info Block File

- `.journal_info_block`
- Root of the volume
- Contains size and location of Journal Header and Buffer

Journal File

- `.journal`
- Root of the volume
- Contains journal header and buffer

© SANS.
All Rights Reserved

Mac Forensic Analysis

HFS+ can implement the journaling feature, but is not required. Most Mac OS X installations do, by default, install on HFS+ with journaling enabled.

The journal can be used to restore the file system to a safe state if the volume suffered a crash or was otherwise unmounted ungracefully.

The journal consists of two files, both of which are located in the root of the volume and have the hidden file attribute set.

- `.journal_info_block`
- `.journal`

The Journal Info Block file contains the size and location of the journal header and buffer.

The Journal contains the journal header and buffer. The journal header contains transaction information. The journal buffer holds transactions.

The layout and structure of the journal files will not be covered in this class, however it is detailed in TN1150.

G-C Partners has created a journal parser called “Triforce AHJP HFS+ Journal Parse” is available at <https://www.gettriforce.com/product/hfs-journal-parser/>.

References:

Apple Tech Note 1150 – Available at dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source – Available at github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h

Mac OS X Internals: A Systems Approach by Amit Singh – Chapter 12

Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin – Chapter 16



Exercise 1.3 – HFS+

This page intentionally left blank.

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

© SANS,
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.



Section 1 – Part 5

Mounting Disk Images

© SANS,
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Method 1 - xmount

```
1. $ mkdir /Volumes/dademurphy_image/  
2. $ mkdir /Volumes/dademurphy_mounted/  
3. $ sudo xmount --in ewf --out dmg  
~/FOR518/dademurphy.E01 /Volumes/dademurphy_image/  
4. $ hdiutil attach -nomount  
/Volumes/dademurphy_image/dademurphy.dmg  
5. $ mount_hfs -j -o rdonly,noexec,noowners /dev/disk#  
/Volumes/dademurphy_mounted/
```

The first method to mount an image file is to use the xmount command from www.pinguin.lu/index.php.

1. Use the `mkdir` command to create a mount point for the `xmount` output. In this class the directory name `dademurphy_image` is used because it will just host the image file.
2. Use the `mkdir` command to create a mount point for the mounted drive. The directory `dademurphy_mounted` is used in this class to represent the mounted disk image.
3. Uses `xmount` to mount the `dademurphy.e01` image (where you have your image located, the example shows `~/FOR518`) as a DMG file. This command requires you to use the `sudo` command, thus it will ask you for your administrator password when executed.
 - `--in` – Tells `xmount` what input file type to expect, our images are in a compressed EWF format.
 - `--out` – Tells `xmount` what output format you want, we want a DMG file so we can mount it in Finder.
 - Input File – Where the image file is located.
 - Mount Point – Newly created specifically for this image.
4. Uses the `hdiutil` command with the “attach” verb to mount the newly created DMG volume so it is available in Finder and Terminal application. Use the `-nomount` argument to suppress mounting (for now). The output from this command will display a `/dev/disk#`, use the appropriate disk device in the next command.
5. Use the `mount_hfs` command with the following parameters to mount the `/dev/disk#` (from the previous command) to the `/Volumes/dademurphy_mounted/` mount point. This drive will now be available in the Finder or Terminal applications.
 - `-j` – Ignore the journal
 - `-o` – Options:
 - `rdonly` – Mount in read-only mode.
 - `noexec` – Do not allow execution of binaries on mounted system.
 - `noowners` – Ignore ownership on the mounted volume.

You can access this newly created mounted drive on `/Volumes/dademurphy_mounted/`.

Method 2 - ewfmount

```
1. $ mkdir /Volumes/dademurphy_image/  
2. $ mkdir /Volumes/dademurphy_mounted/  
3. $ ewfmount ~/FOR518/dademurphy.E01  
/Volumes/dademurphy_image/  
4. $ ln -s /Volumes/dademurphy_image/ewf1  
~/FOR518/dadeimage.dmg  
5. $ hdiutil attach -nomount ~/FOR518/dadeimage.dmg  
6. $ mount_hfs -j -o rdonly,noexec,noowners /dev/disk#  
/Volumes/dademurphy_mounted/
```

© SANS:
All Rights Reserved

Mac Forensic Analysis

It can never hurt to have more than one way to mount an image. A second way uses the `ewfmount` command from the `libewf` package available at code.google.com/p/libewf/.

1. Use the `mkdir` command to create a mount point for the `ewfmount` output. The directory name `dademurphy_image` is used in the example.
2. Use the `mkdir` command again to create a mount point for the mounted disk image, the example `dademurphy_mounted` in used in the example above.
3. Use `ewfmount` to mount the `dademurphy.e01` image to the `/Volumes/dademurphy_image/` mount point.
4. Use the `ln -s` command to create a symbolic link for the `ewf1` file, name the link `dadeimage.dmg`. (A DMG file is needed for `hdiutil` to recognize the file.)
5. Uses the `hdiutil` command with the “attach” verb to mount the newly created DMG volume so it is available in Finder and Terminal application. Use the `-nomount` argument to suppress mounting (for now). The output from this command will display a `/dev/disk#`, use the appropriate disk device in the next command.
6. Use the `mount_hfs` command with the following parameters to mount the `/dev/disk#` (from the previous command) to the `/Volumes/dademurphy_mounted/` mount point. This drive will now be available in the Finder or Terminal applications.
 - j – Ignore the journal
 - o – Options:
 - rdonly – Mount in read-only mode.
 - noexec – Do not allow execution of binaries on mounted system.
 - noowners – Ignore ownership on the mounted volume.

You can access this newly created mounted drive on `/Volumes/dademurphy_mounted/`.

Disk Eject & Unmount

```
1. $ diskutil list  
  
2. $ diskutil eject /dev/disk*  
  
3. $ mount  
  
4. $ umount /Volumes/dademurphy_image/
```

© SANS.
All Rights Reserved

Mac Forensic Analysis

The disk images should stay mounted and available until you reboot or until you eject/unmount them.

When you are finished with the mounted images, you will need to eject and unmount them.

1. Use the `diskutil list` command to view the list of mounted disks. Find the disk that you want to eject.
2. Use the `diskutil eject` command on the disk you would like to eject. (This may also be done by pressing the eject button in the Finder application.)
3. Use the `mount` command to view the list of mounted disks. Find the disk that you want to unmount (likely `/Volumes/dademurphy_image/`, if you following the naming scheme from the examples.)
4. Use the `umount` command with the mount point to unmount the disk. A troublesome disk may have to be unmounted using the `-f` option, to forcibly unmount the disk.

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

This page intentionally left blank.



Section 1 – Part 6

BlackLight 101

This page intentionally left blank.

BlackBag Technologies

BlackLight

Forensic Suite specially designed for Mac analysis (will also analyze Windows systems!)

Runs natively on Mac or Windows Systems

Image Formats: dd, dmg, sparsebundle, sparseimage, vmdk, E01, L01, SMART, iXAM, Zdziarski, Cellebrite, Elcomsoft, and MPE+

Logical extraction from iDevices

© SANS:
All Rights Reserved

Mac Forensic Analysis

We will be using the BlackLight forensic suite in class for some of the labs; however, it will be available during the whole of the class via the Network License Server running on your instructor's laptop.

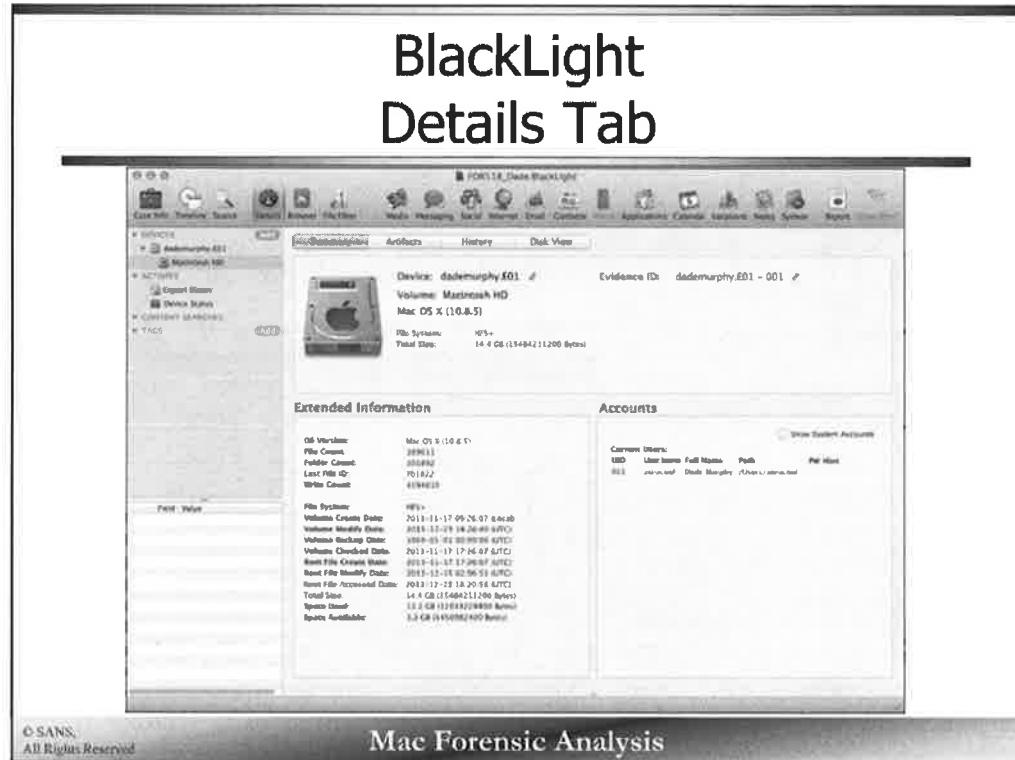
The suite runs natively on both Mac and Windows-based systems (normally with a USB dongle, or a Network License server setup at your agency).

Along with the cross-platform capabilities – the software can also analyze Mac and Windows file systems – this is convenient when a system has been Boot Camped.

Most disk image formats are accepted as well as many from the most popular iDevice acquisition suites.

While BlackLight is not an acquisition tool (MacQuisition is used for acquisition), it is able to acquire a logical extraction from iDevices.

The BlackLight software is available from BlackBag Technologies, blackbagtech.com.



By selecting the Macintosh HD drive under the DEVICES section, we can view the disk Details tab.

The information in this “Summary” window includes file system specifics and user accounts.

You may select and explore the other tabs, “Artifacts”, “History”, and “Disk View”.

BlackLight

Timeline Tab



The Timeline tab shows a timeline information for the selected hard drive. This can be zoomed in/out for specific timeline analysis.

BlackLight Search Tab



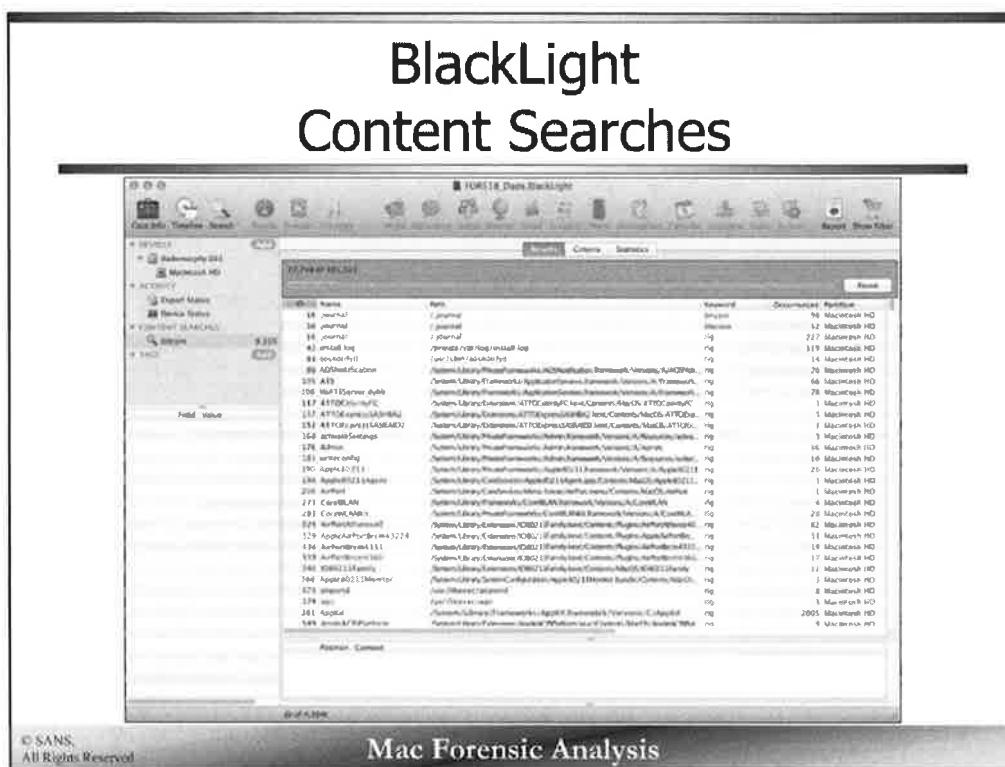
The Search tab allows an investigator to perform keyword searches.

Each keyword list has a Name as filled in the Name textbox. The keywords may be typed in the Keywords pane.

File extensions may be ignored, but typing them into the Ignore Extensions pane. Other keyword configurations may be selected in the pane on the right side.

Select “Start Search” when ready search.

BlackLight Content Searches

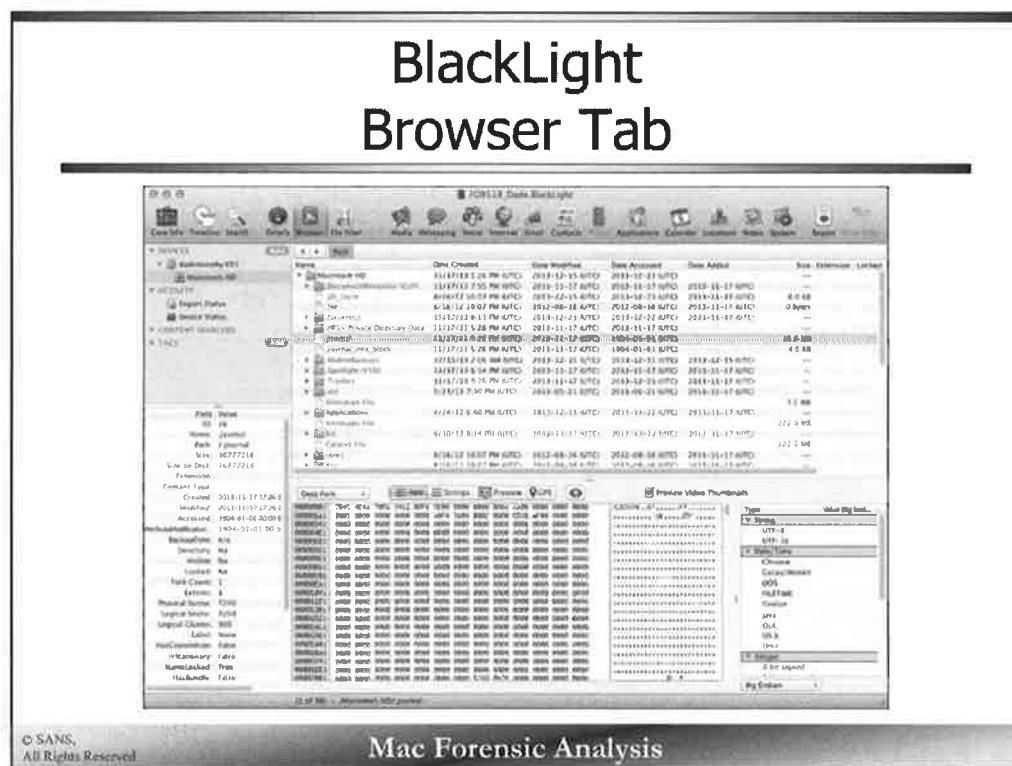


The Content Searches window will appear once a search has started. This window shows the results from the keyword search, as well as the progression.

You may select the “Criteria” or “Statistics” tabs for more information pertaining to this particular keyword search.

BlackLight

Browser Tab



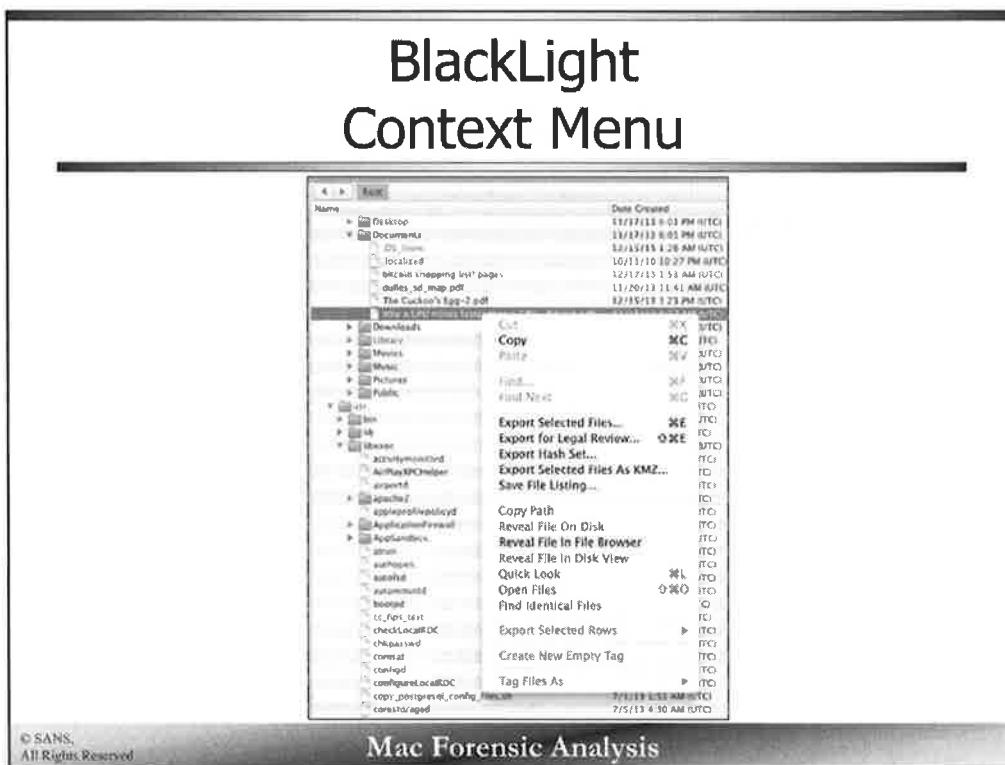
The Browser tab shows the file system as an investigator is most likely used to. This view shows the file system in a tree format with hidden files shown in gray and other file metadata including timestamps and file size.

In the lower pane (the bar may have to be moved up from the bottom of the window) shows the file. The views available include Hex, Strings, Preview, and GPS. An analyst may also select the “eye” shaped button to do a “Quick Look” on the file. The Data and Resource fork may also be chosen. In the Hex view the data-type window on the right will be shown for the analyst to select various data types if conversion is needed.

In the lower-left pane the file metadata and extended attributes are shown. Everything from file size, file name, timestamps, Finder data, disk location, to extended attributes are available in this window. Lots of good information may be found here!

BlackLight

Context Menu



A “right-click” (or two-finger click on a track pad, or control-click) will bring up a context menu as shown above. This menu allows the analyst to perform certain actions such as export information, jump to a file location, or tag a file.

File tagging is similar to bookmarking as seen in other forensic suites. These tags will show up in the left pane under “TAGS”.

BlackLight File Filter Tab

The screenshot shows the BlackLight application window. At the top, there's a toolbar with various icons. Below the toolbar is a menu bar with 'File', 'Edit', 'View', 'Search', 'Tools', 'Help'. The main area is titled 'File Filter' and contains a table with several columns: 'Name', 'Size', 'Created', 'Last Modified', 'Date Accessed', 'Extension', 'Owner', and 'Path'. The table lists numerous files, mostly with '.phar' extensions, such as '121 - AppPreference.phar', '122 - AppPreference2.phar', etc. The 'Path' column shows paths like '/Applications/Java/Phar', '/Applications/Java/Phar...', etc. At the bottom of the table, it says '0 of 17,230'.

Name	Size	Created	Last Modified	Date Accessed	Extension	Owner	Path
121 - AppPreference.phar	1.1 kB	2011-09-29 07:07	2011-10-01 07:07	2010-12-28 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/121 - AppPreference.phar
122 - AppPreference2.phar	3.3 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/122 - AppPreference2.phar
123 - AppPreference3.phar	3.4 kB	2011-12-06 07:00	2012-01-13 07:00	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/123 - AppPreference3.phar
124 - AppPreference4.phar	2.8 kB	2011-12-06 07:00	2012-01-13 07:00	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/124 - AppPreference4.phar
125 - AppPreference5.phar	41.9 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/125 - AppPreference5.phar
126 - AppPreference6.phar	61.1 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/126 - AppPreference6.phar
127 - AppPreference7.phar	1.4 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/127 - AppPreference7.phar
128 - AppPreference8.phar	2.6 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/128 - AppPreference8.phar
129 - AppPreference9.phar	4.7 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/129 - AppPreference9.phar
130 - AppPreference10.phar	1.5 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/130 - AppPreference10.phar
131 - AppPreference11.phar	3.5 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/131 - AppPreference11.phar
132 - AppPreference12.phar	3.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/132 - AppPreference12.phar
133 - AppPreference13.phar	3.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/133 - AppPreference13.phar
134 - AppPreference14.phar	64.2 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/134 - AppPreference14.phar
135 - AppPreference15.phar	1.5 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/135 - AppPreference15.phar
136 - AppPreference16.phar	4.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/136 - AppPreference16.phar
137 - AppPreference17.phar	2.1 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/137 - AppPreference17.phar
138 - AppPreference18.phar	1.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/138 - AppPreference18.phar
139 - AppPreference19.phar	773 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/139 - AppPreference19.phar
140 - AppPreference20.phar	800 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/140 - AppPreference20.phar
141 - AppPreference21.phar	3.4 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/141 - AppPreference21.phar
142 - AppPreference22.phar	8.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/142 - AppPreference22.phar
143 - AppPreference23.phar	14.1 Bytes	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/143 - AppPreference23.phar
144 - AppPreference24.phar	1.6 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/144 - AppPreference24.phar
145 - AppPreference25.phar	4.6 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/145 - AppPreference25.phar
146 - AppPreference26.phar	6.8 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/146 - AppPreference26.phar
147 - AppPreference27.phar	17.1 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/147 - AppPreference27.phar
148 - AppPreference28.phar	4.1 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/148 - AppPreference28.phar
149 - AppPreference29.phar	1.1 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/149 - AppPreference29.phar
150 - AppPreference30.phar	1.6 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/150 - AppPreference30.phar
151 - AppPreference31.phar	4.6 kB	2011-09-29 07:07	2011-10-01 07:07	2011-09-29 07:00	.phar	Application/Java/Phar	/Applications/Java/Phar/151 - AppPreference31.phar

© SANS,
All Rights Reserved

Mac Forensic Analysis

The File Filter tab allows an investigator to select certain files based on some type of data. Whether it is size, file type, or by creation date. Many different combinations can be played with. The author highly recommend spending some time with this feature. It can help you pinpoint specific files quickly.

BlackLight

Artifact Tabs



© SANS.
All Rights Reserved

Mac Forensic Analysis

BlackLight also does some pre-processing when it comes to various popular artifacts. We will be reviewing some of these more in-depth during the course exercises, but you may start reviewing the contents on your own time.



Exercise 1.4 – BlackLight and Image Mounting

© SANS.
All Rights Reserved

Mac Forensic Analysis

This page intentionally left blank.

Agenda

Part 1 – Mac Fundamentals

Part 2 – Acquisition & Live Response

Part 3 – Disks & Partitions

Part 4 – HFS+ File System

Part 5 – Mounting Disk Images

Part 6 – BlackLight 101

This page intentionally left blank.



FOR518

Mac Forensic Analysis

The **SANS** Institute



Sarah Edwards
oompa@csh.rit.edu
@iamevl twin



@sansforensics

<http://computer-forensics.sans.org>

© SANS,
All Rights Reserved

Mac Forensic Analysis

Author: Sarah Edwards
oompa@csh.rit.edu
<http://twitter.com/iamevl twin>
<http://twitter.com/sansforensics>

