518.2

File Systems and System Triage

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

518.2
File Systems and System Triage

SANS

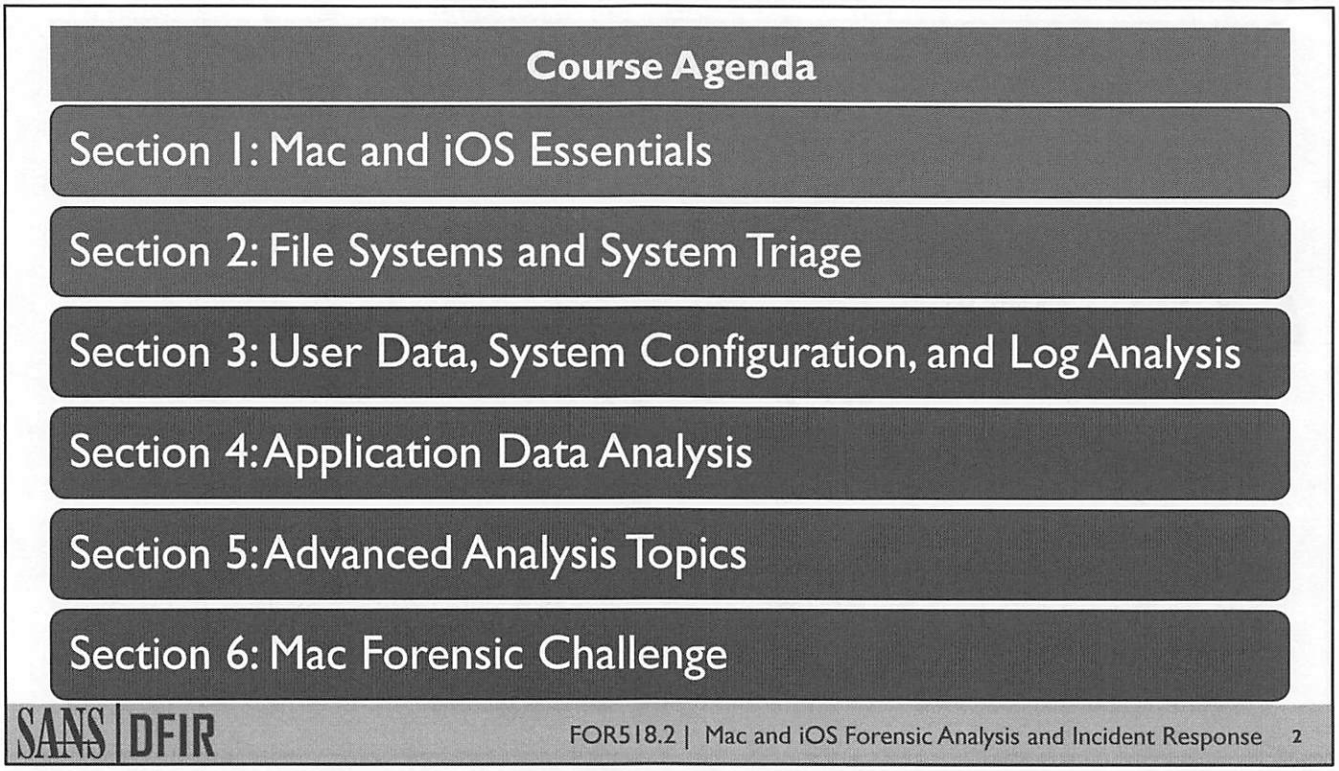
THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

FOR518.2 | Mac and iOS Forensic Analysis and Incident Response

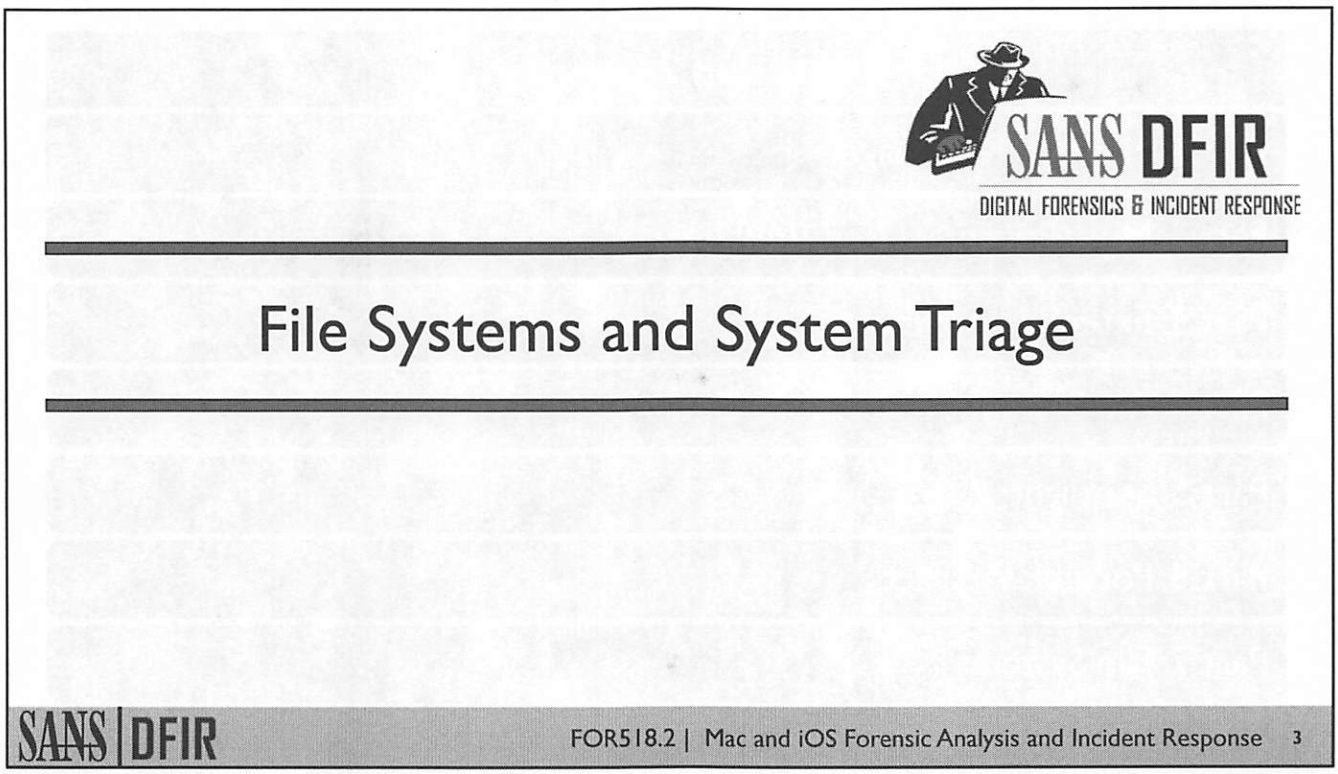
FOR518 Section 2: File Systems and System Triage

Author: Sarah Edwards
oompa@csh.rit.edu
mac4n6.com
http://twitter.com/iamevltwin

https://digital-forensics.sans.org/
http://twitter.com/sansforensics



This page intentionally left blank.

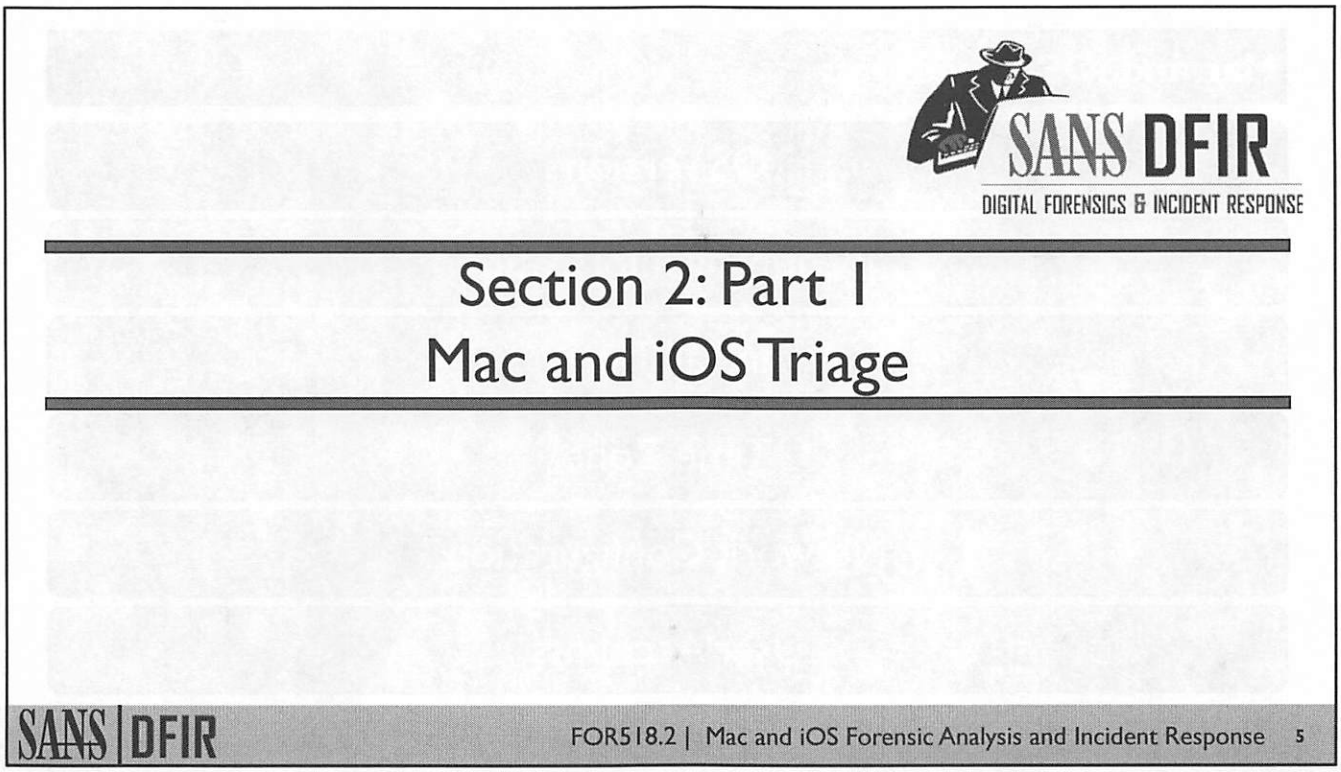


This page intentionally left blank.

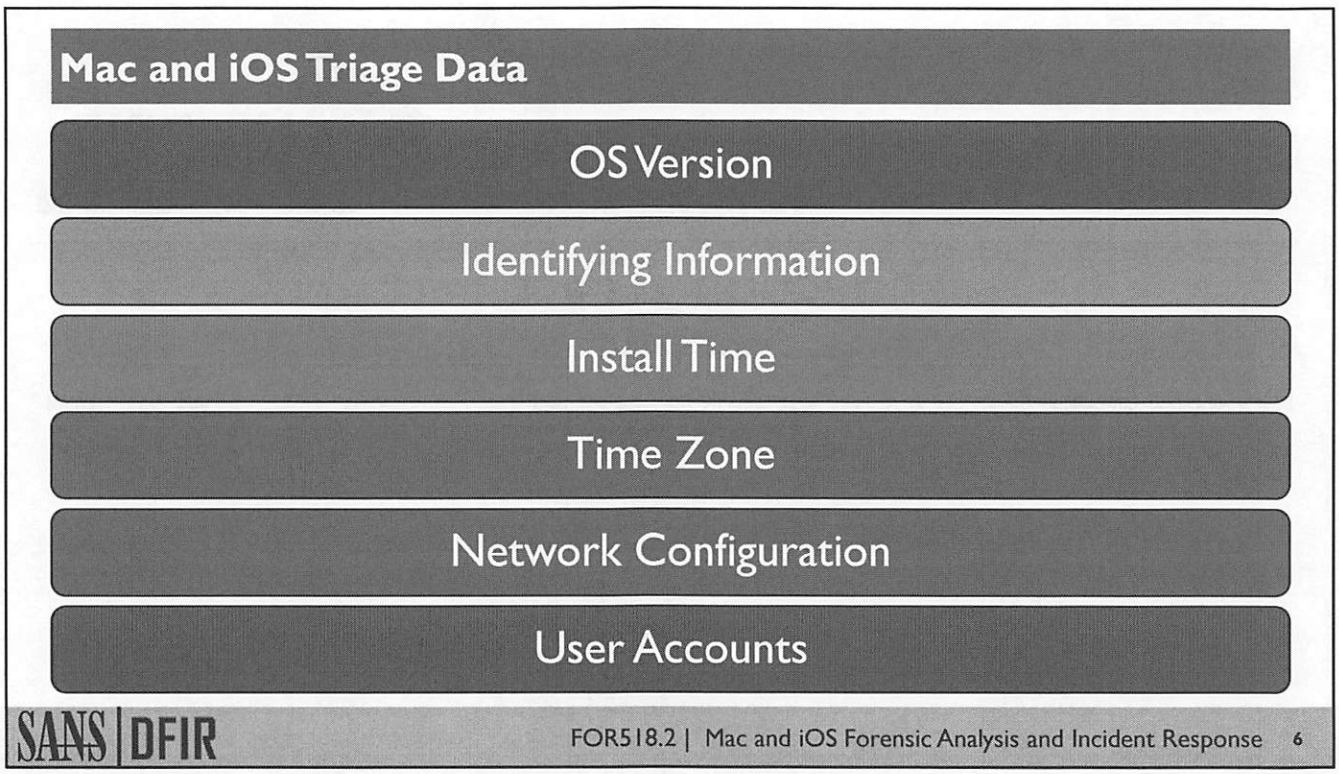
Section 2: Agenda

- Part 1: Mac and iOS Triage
- Part 2: Most Recently Used (MRUs)
- Part 3: Extended Attributes
- Part 4: File System Events Store Database
- Part 5: Spotlight
- Part 6: Portable Artifacts
- Part 7: File Systems

This page intentionally left blank.



This page intentionally left blank.



Extracting triage data from Mac and iOS can be both very similar and completely different. In this module, we will extract the basics of each system to get an idea of what we are looking at to include OS versions, configurations, and identifying information.

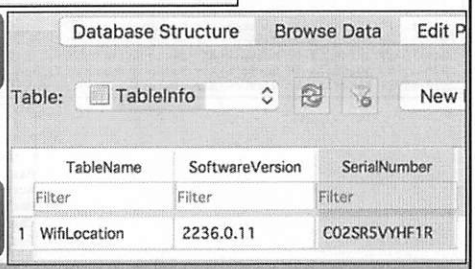
MacOS Information /System/Library/CoreServices/SystemVersion.plist

Key	Type	Value
▼ Root	Dictionary	(5 items)
ProductBuildVersion	String	16C67
ProductCopyright	String	1983-2016 Apple Inc.
ProductName	String	Mac OS X
ProductUserVisibleVersion	String	10.12.2
ProductVersion	String	10.12.2

Mac Serial Number:

- Location Database (cache_encrypted*.db and others) and System Profiler Output

Mac Model: Network Configurations

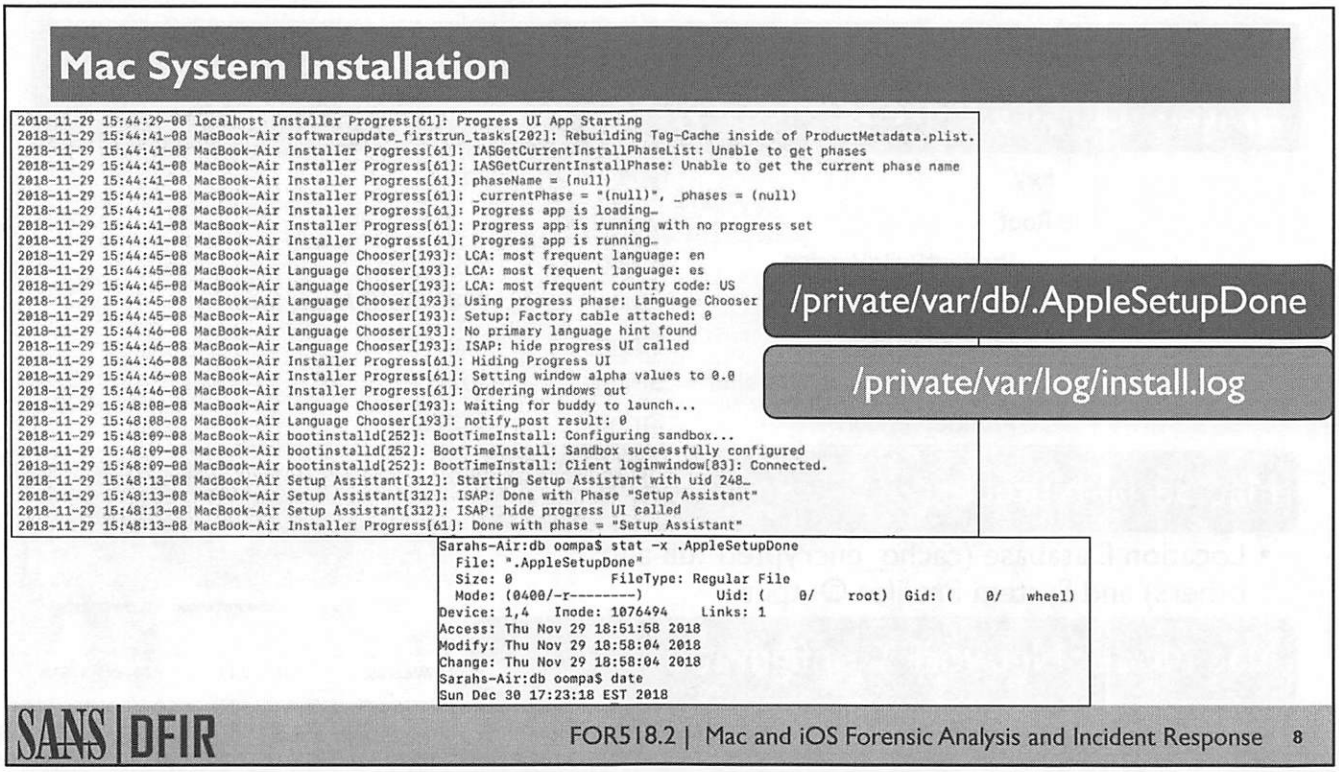


The `SystemVersion.plist` property list located in the `/System/Library/CoreServices/` directory contains the system version and build information.

In the example above, the system name and version is Mac OS X 10.12.2, while the build version is 16C67.

The Mac's serial number can be found by extracting it from a live system, perhaps from the system_profiler output. On newer versions of macOS, it can be found in a variety of different databases. The example shown above is from the cache_encryptedA.db location database. It will usually be found in a small table named TableInfo.

The Mac's model number can be found in its network configuration list files, shown later.



An original installation date may be found as the creation date for the `.AppleSetupDone` file located in the `/private/var/db/` directory.

The OS X installation date may be able to be found in the `install.log` files located in the `/var/log` directory if the older files have not been turned over. These dates are when different versions of OS X were installed after the original installation.

There may be a difference in timestamp time zones, as the original time zone for all Macs is Cupertino, CA, time. Note the three-hour difference in the example screenshots above. The timestamp in the `install.log` is three hours off from the hidden files. The `install.log` was being written to before the user had set up the time zone for the system.

Mac Time Zone Setting: Static Location

- /etc/localtime

```
nibble:etc sledwards$ ls -l localtime
lrwxr-xr-x  1 root  wheel  36 Apr 13 17:28 localtime -> /usr/share/zoneinfo/America/New_York
```

- /Library/Preferences/.GlobalPreferences.plist

com.apple.preferences.timezone.selected_city	Dictionary	(10 items)
RegionalCode	String	DC
Version	Number	1
TimeZoneName	String	America/New_York
Latitude	Number	38.89511
GeonameID	Number	4,140,963
Population	Number	601,723
Longitude	Number	-77.03637
CountryCode	String	US
Name	String	Washington D.C.

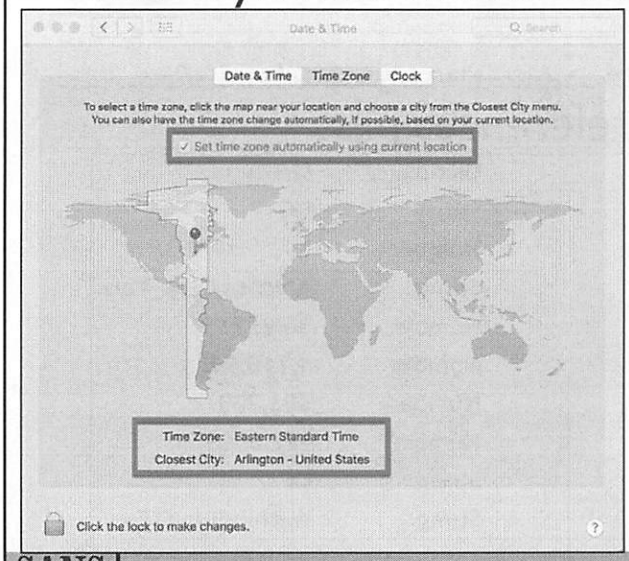
The link to the file located in `/etc/localtime` contains the current time zone value for the system. In the screenshot, you can see the local time zone is set for New York, or the Eastern Standard Time zone.

The property list located in the `/Library/Preferences/` directory contains the time zone configuration data. The time zone is set for Washington, DC. This is for the same system configured for the New York time zone shown above. While the city may change, the time zone may be located in a more general location.

This configuration was likely chosen by the user when the system was configured during setup using the time zone map feature.

Mac Time Zone Setting: Using Location Services

- /Library/Preferences/.GlobalPreferences.plist

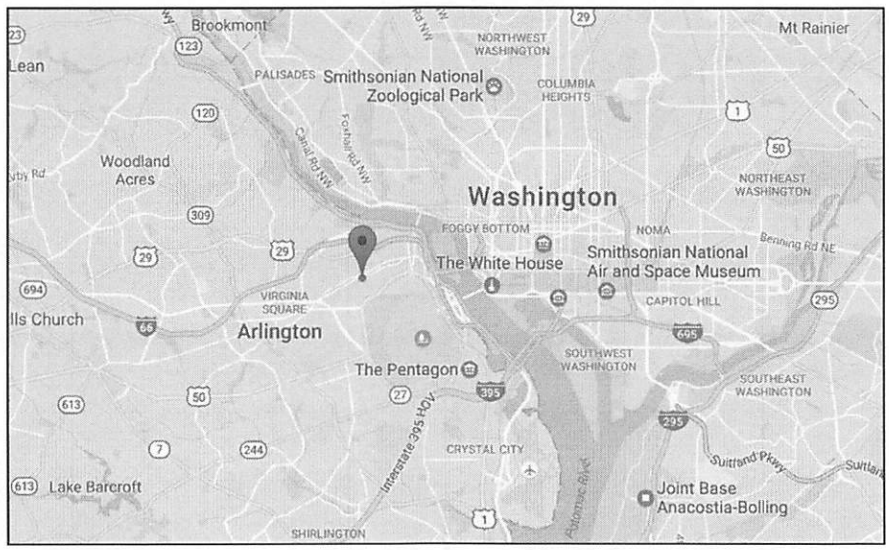


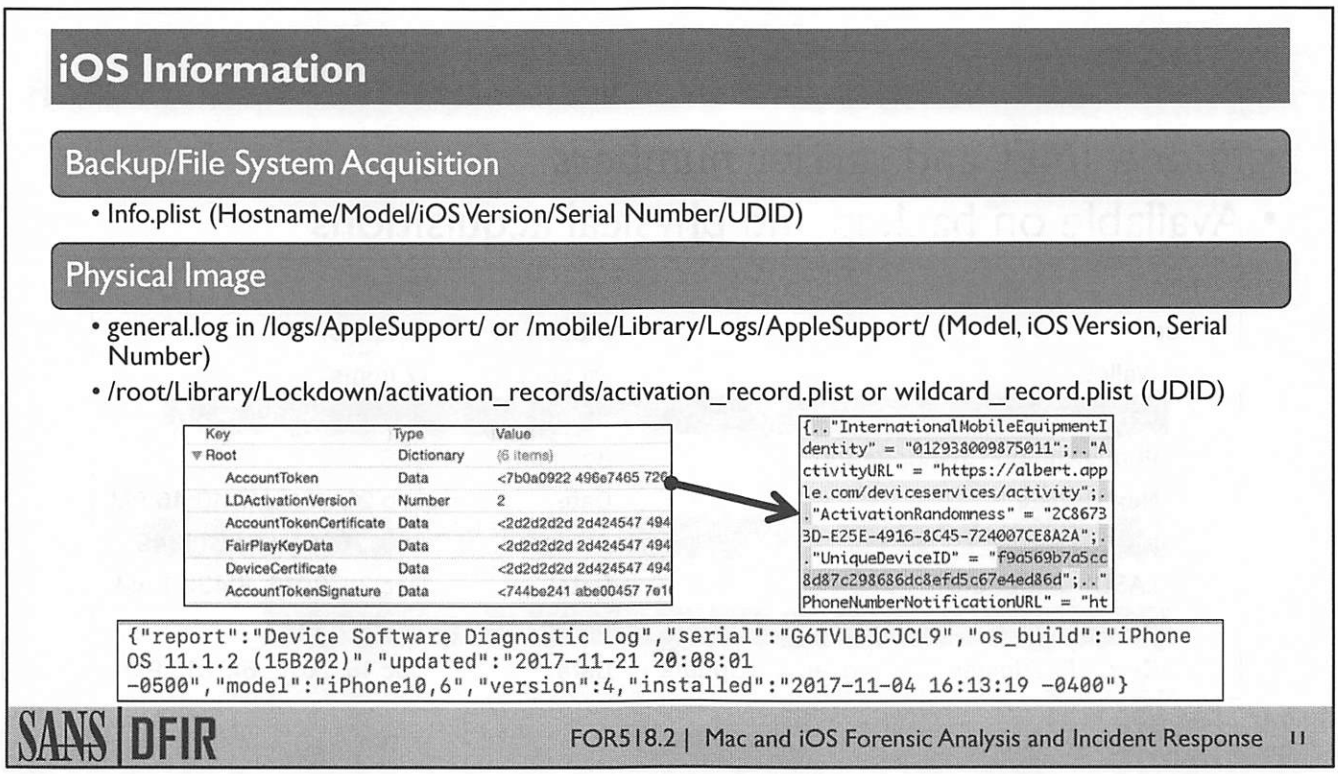
com.apple.TimeZonePref.Last_Selected_City	Array	(10 items)
Item 0	String	38.89076
Item 1	String	-77.08475
Item 2	String	0
Item 3	String	America/New_York
Item 4	String	US
Item 5	String	Arlington
Item 6	String	U.S.A.
Item 7	String	Arlington
Item 8	String	U.S.A.
Item 9	String	DEPRECATED IN 10.6

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 10

If the user has the time zone set to the current location, the system will attempt to get as close as possible to where the system's IP is coming from. The `.GlobalPreferences.plist` file will show a latitude and longitude as well as the textual representation of this location.

While this may be close to where the system was, it is unlikely it will be an exact location. The example used above was about 2 miles off.





Simple device information such as model, iOS version, device serial number, UDID, and hostname can be found in a variety of locations, depending on whether you have a file system/backup acquisition or a full physical image.

On backups or file system acquisitions, this information can easily be found in the `Info.plist` that is created. Some acquisition tools do not necessarily call this the `Info.plist`, but they should have a similar file containing the identifying information for the device.

On physical images, this information is stored in many files:

- The model, iOS version, and serial number can be found in the `general.log` file located in `/logs/AppleSupport/` or `/mobile/Library/Logs/AppleSupport/` as the header of the log file.
- The UDID can be found in the `activation_record.plist` or `wildcard_record.plist` file located in the `/root/Library/Lockdown/` directory as embedded data in the `AccountToken` key.

Other files on the device contain similar information:

- Model: `/preferences/SystemConfiguration/NetworkInterfaces.plist`
- Model, Hostname: `/preferences/SystemConfiguration/preferences.plist`
- Hostname: `/preferences/SystemConfiguration/com.apple.mobilegestalt.plist`

iOS Information: com.apple.commcenter.plist

- Phone, IMEI, and carrier numbers
- Available on backup and physical acquisitions

Key	Type	Value
▼ Root	Dictionary	(8 items)
▶ Wallet	Dictionary	(2 items)
CarrierBundleName	String	310260_ID-89012601
PhoneNumberChangeReport	Boolean	YES
NextUpdate	Date	Sep 21, 2015, 12:10:46 PM
ICCID	String	8901260122559914849
LASDNextUpdate	Date	Dec 28, 2016, 8:43:01 AM
PhoneNumber	String	15713158868
CarrierEntitlementsControllerNextUpdate	Date	Dec 15, 2016, 6:01:13 PM

Physical: `/private/var/wireless/Library/Preferences/com.apple.commcenter.plist`
Backup: `/wireless/Library/Preferences/com.apple.commcenter.plist`

The `com.apple.commcenter.plist` can be viewed to determine the phone number and ICCID information. This plist file may also contain carrier information. The "CarrierBundleName" can be used to search for the carrier on https://www.imei.info/carriers/. For example, 310260 belongs to the T-Mobile network. See the screenshot.

iOS Device Setup Data: com.apple.purplebuddy.plist

- Locale, Potential Hardware, Setup Install Time
- Available on backup and physical acquisitions

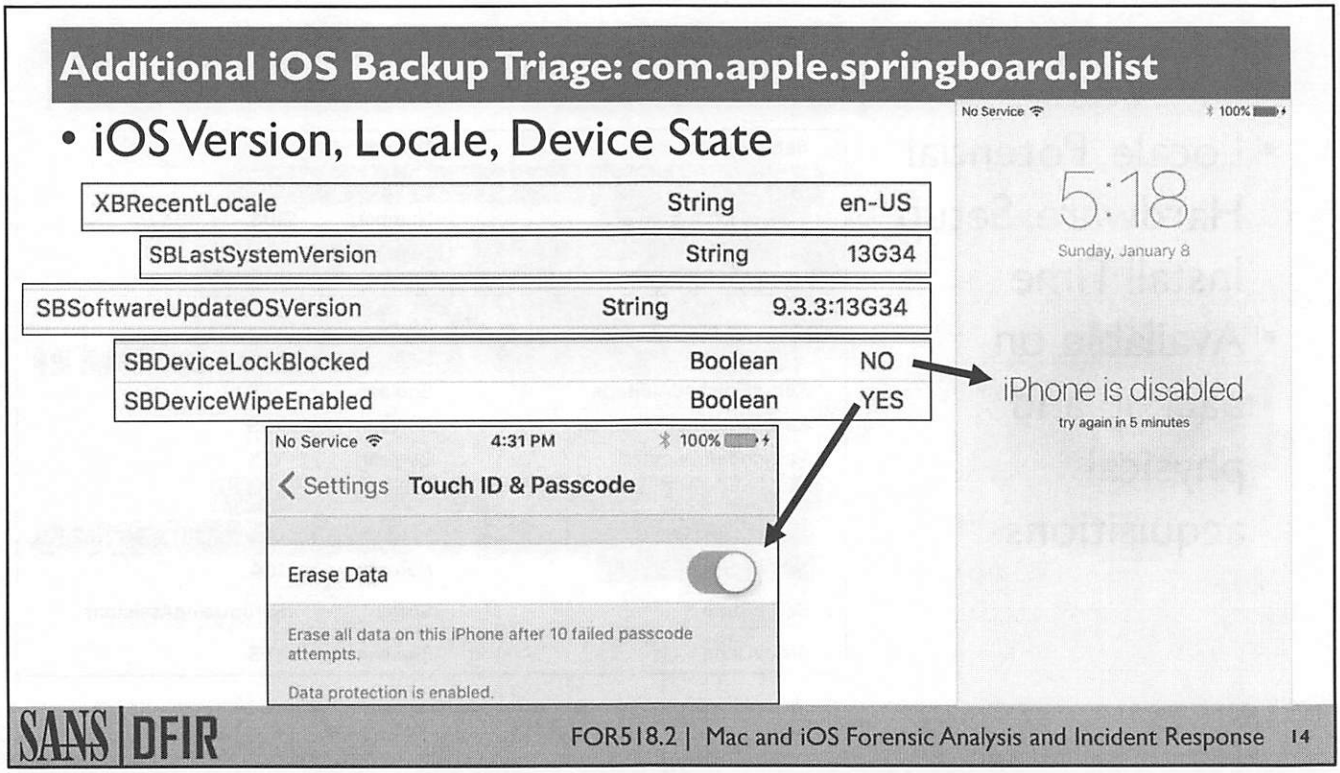
SetupVersion	Number	10
Locale	String	en_US
SSDeviceType_J85AP	Number	103
▼ GuessedCountry	Dictionary	(2 items)
▼ countries	Array	(1 item)
Item 0	String	US
at	Date	May 18, 2016, 9:19:31 PM
SetupFinishedAllSteps	Boolean	YES
KeychainSyncPresented	Boolean	YES
PasscodePresented	Boolean	YES
Language	String	en-US
SetupLastExit	Date	May 18, 2016, 9:22:32 PM
SSDeviceType_J86AP	Number	104
SetupState	String	SetupUsingAssistant
SetupDone	Boolean	YES

Physical: `/private/var/mobile/Library/Preferences/com.apple.purplebuddy.plist`
Backup: `/mobile/Library/Preferences/com.apple.purplebuddy.plist`

The `com.apple.purplebuddy.plist` can be viewed to determine the device setup information, including the original locale, setup time, and device hardware model.

The example above performed the "Purple Buddy" device setup on 05/18/2016. This is where the user configures all items about their devices, including their passcode, locale, keychain, Wi-Fi, etc.

The hardware can be determined from the "`SSDeviceType`" keys. The model string '`J85AP`' or '`J86AP`' can be searched for to determine the human-readable model information such as iPad Mini 2. On many devices, there are multiple DeviceTypes represented. They are not necessarily the device currently being used; however, it may indicate previous devices the user has used, since these multiple device type artifacts likely come from previous iOS backups.

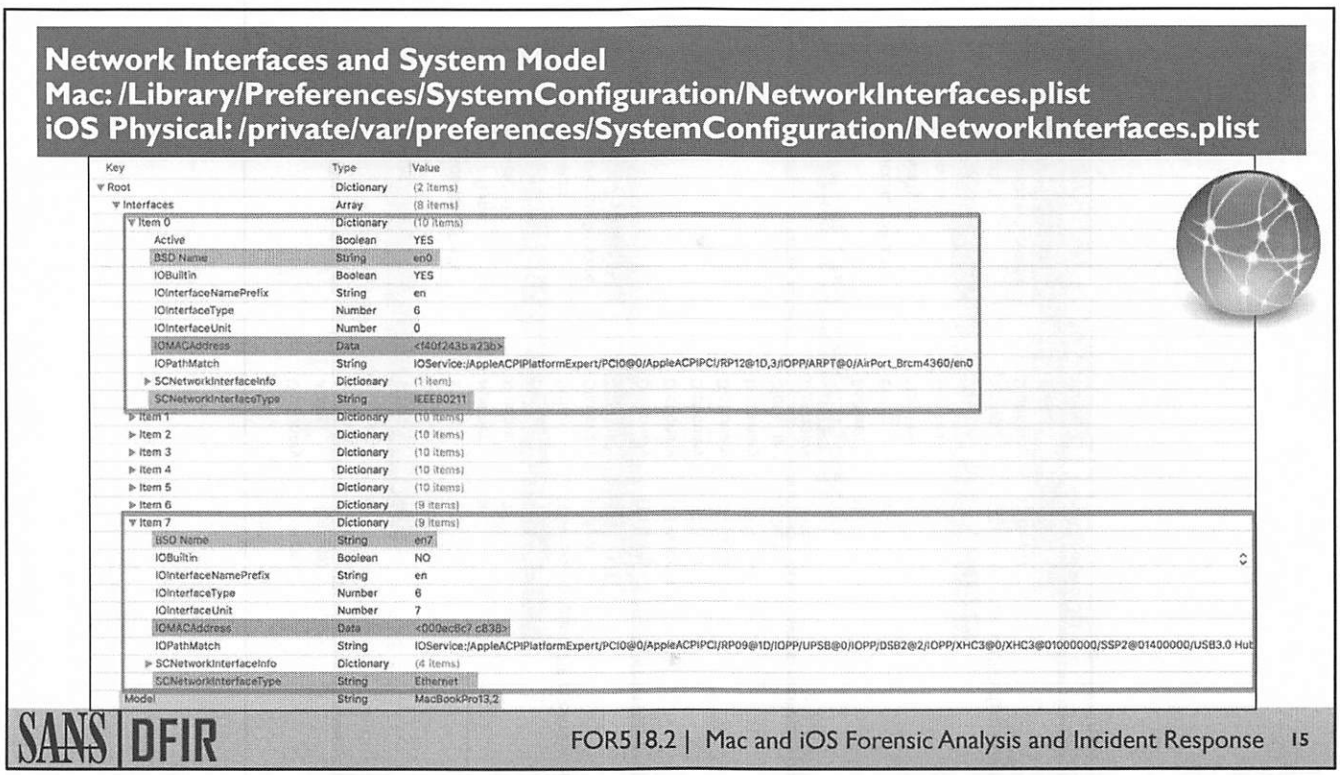


Location: `[/private/var]/mobile/Library/Preferences/com.apple.springboard.plist`

Various Springboard and interface-related items are stored in this plist file.

Additional iOS triage information can be found in the `com.apple.springboard.plist` file. The locale setting (`XBRecentLocale`) can provide native location-based information. In this example, it is "`en-US`". The OS version (`SBLastSystemVersion/SBSoftwareUpdateOSVersion`) is stored as a build number—"`13G34`". This number can be Google searched to determine the numerical OS version (9.3.3).

The `SBDeviceLockBlocked` key keeps track of whether or not the device is disabled (it is not in this case), while the `SBDeviceWipeEnabled` key determines if the user set the "Erase Data" option in their "Touch ID and Passcode" settings to wipe the device after 10 failed attempts at the passcode.



Mac: `/Library/Preferences/SystemConfiguration/NetworkInterfaces.plist`
iOS Physical: `/private/var/preferences/SystemConfiguration/NetworkInterfaces.plist`

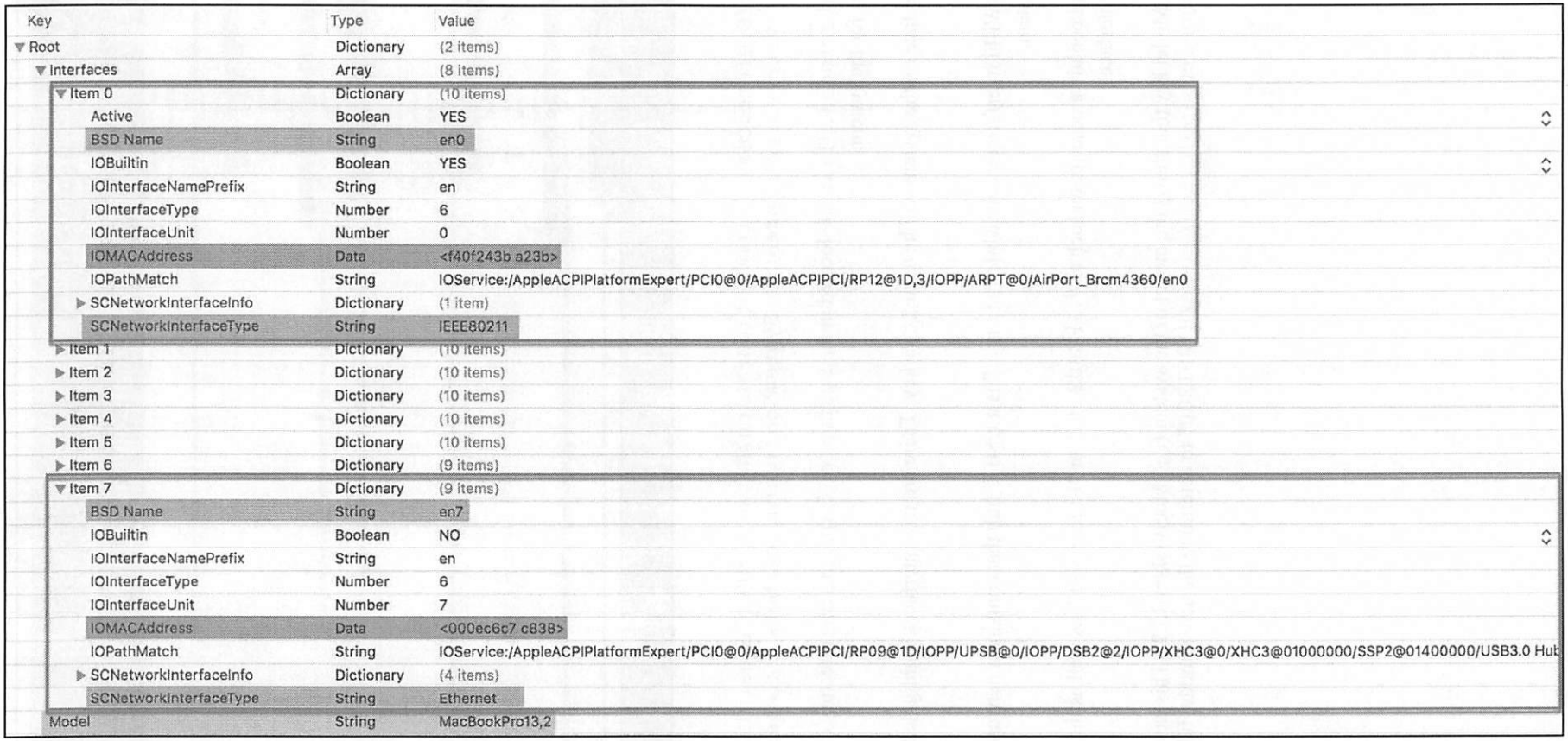
The `NetworkInterfaces.plist` file located in the `SystemConfiguration` directory contains the network interfaces for the system.

Each network interface on the system will have an `Item` key. There are two interfaces highlighted in the above screenshot.

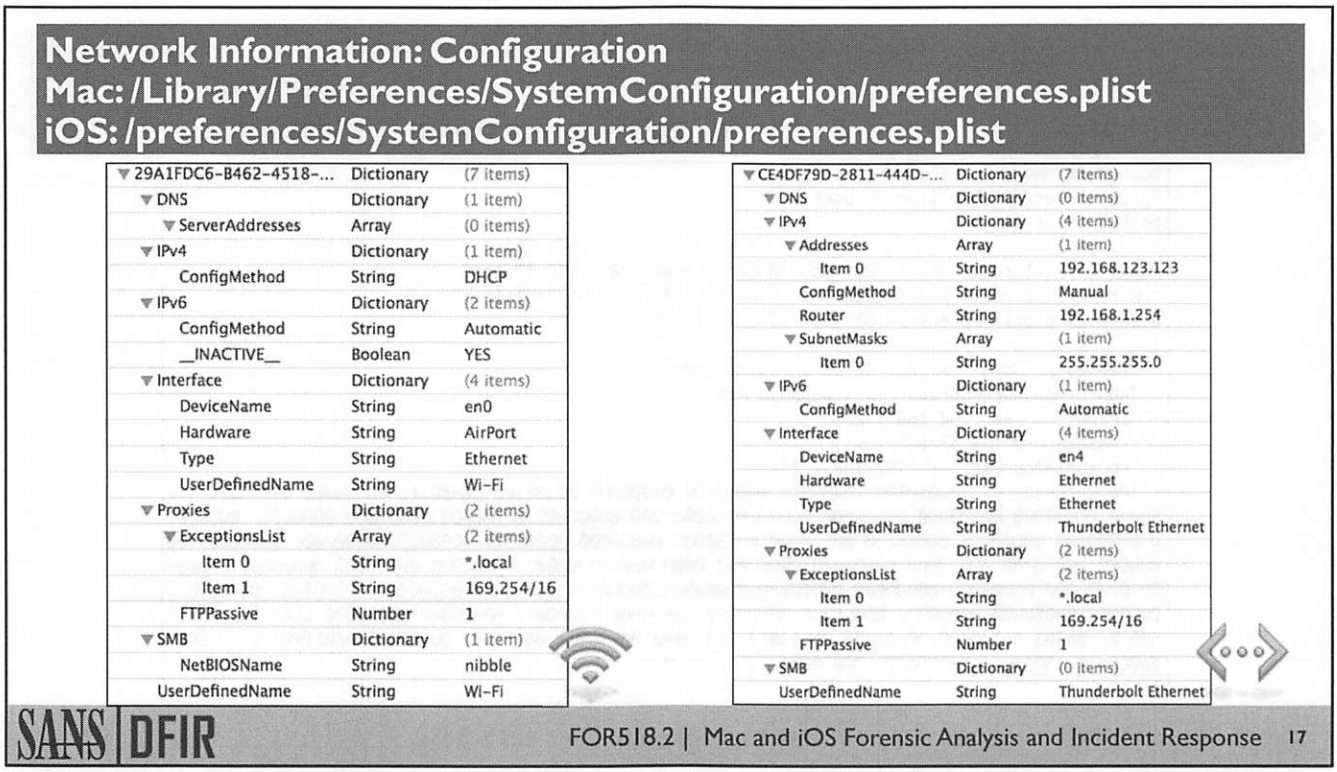
"`Item 0`" is the Wi-Fi interface on en0, while "`Item 7`" is a USB-C hub that contains a network port that was attached to the system.

Each interface will contain a description such as "`IEEE802.11`" or "`Ethernet`", as well as the unique MAC address for each interface.

Finally, there is a Model key that shows the model of the system (i.e., MacBookPro13,2). This can be searched for on "`https://support.apple.com/en-us/HT201300`" to determine the actual model information.



Key	Type	Value
▼ Root	Dictionary	(2 items)
▼ Interfaces	Array	(8 items)
▼ Item 0	Dictionary	(10 items)
Active	Boolean	YES
BSD Name	String	en0
IOBuiltin	Boolean	YES
IOInterfaceNamePrefix	String	en
IOInterfaceType	Number	6
IOInterfaceUnit	Number	0
IOMACAddress	Data	<f40f243b a23b>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/RP12@1D,3/IOPP/ARPT@0/AirPort_Brcm4360/en0
▶ SCNetworkInterfaceInfo	Dictionary	(1 item)
SCNetworkInterfaceType	String	IEEE80211
▶ Item 1	Dictionary	(10 items)
▶ Item 2	Dictionary	(10 items)
▶ Item 3	Dictionary	(10 items)
▶ Item 4	Dictionary	(10 items)
▶ Item 5	Dictionary	(10 items)
▶ Item 6	Dictionary	(9 items)
▼ Item 7	Dictionary	(9 items)
BSD Name	String	en7
IOBuiltin	Boolean	NO
IOInterfaceNamePrefix	String	en
IOInterfaceType	Number	6
IOInterfaceUnit	Number	7
IOMACAddress	Data	<000ec6c7 c838>
IOPathMatch	String	IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/RP09@1D/IOPP/UPSB@0/IOPP/DSB2@2/IOPP/XHC3@0/XHC3@01000000/SSP2@01400000/USB3.0 Hub
▶ SCNetworkInterfaceInfo	Dictionary	(4 items)
SCNetworkInterfaceType	String	Ethernet
Model	String	MacBookPro13,2

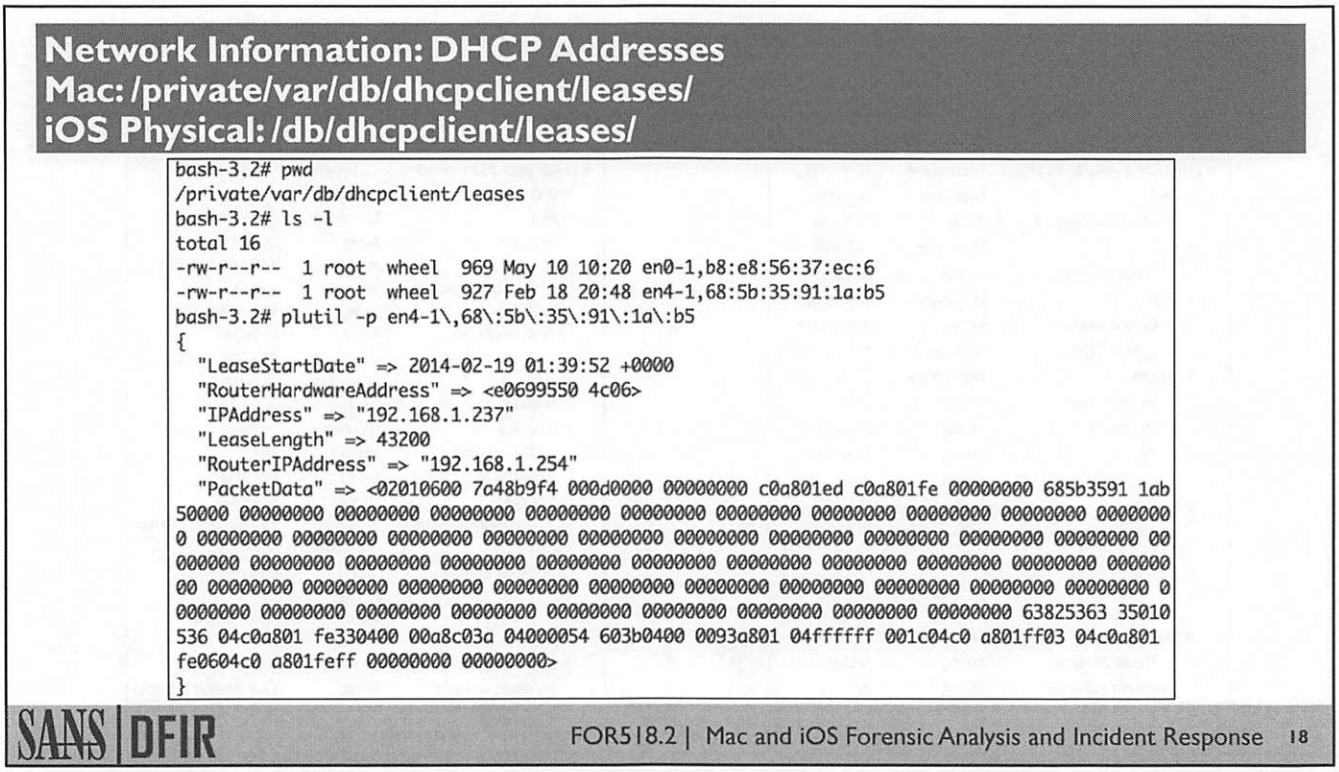


Mac: `/Library/Preferences/SystemConfiguration/preferences.plist`
iOS Backup: `/preferences/SystemConfiguration/preferences.plist`
iOS Physical: `/private/var/preferences/SystemConfiguration/preferences.plist`

The `NetworkServices` key contains the configuration for each network interface. Two examples are shown above.

The example on the left contains a Wi-Fi interface (`UserDefinedName` key). This interface uses DHCP (rather than a static IP), and it has a `NetBIOSName` key that contains the NetBIOS name of the system. The network interface device is `en0`.

The example on the right contains a Thunderbolt Ethernet interface (`UserDefinedName` key). This interface has a static IP configured (as well as router and subnet mask). The network interface device is `en4`.



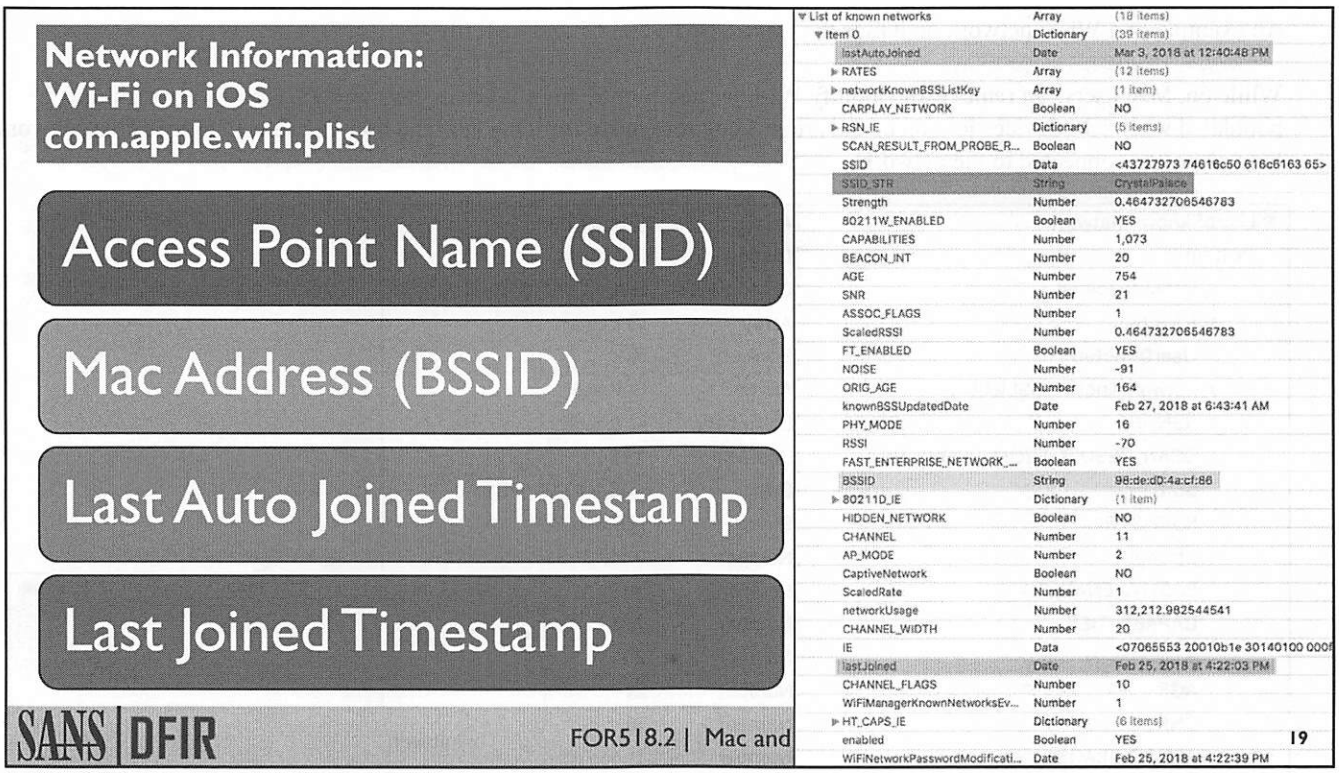
Mac: `/private/var/db/dhcpclient/leases/`
iOS Physical: `/db/dhcpclient/leases/`

The files located in the directories above contain configurations for DHCP network settings. These contain the settings for the latest connection on the specified interface.

The example above shows two DHCP configurations: One for the en0 adapter and another for the en4 adapter. Each adapter has an associated MAC address in the filename.

Each file contains:

- Lease Start Date
- Router MAC Address
- Assigned IP Address
- SSID of Access Point (Wi-Fi only)
- DHCP Lease Length (in minutes)
- Router IP Address
- Packet Data



Mac:
`/Library/Preferences/SystemConfiguration/com.apple.airport.preferences.plist (10.9-)`
iOS Backup: `/preferences/SystemConfiguration/com.apple.wifi.plist`
iOS Physical: `/private/var/preferences/SystemConfiguration/com.apple.wifi.plist`

These plists contain network information of the "remembered" or "saved" networks. Each network is stored in its own `Item` key.

The "remembered" networks are a list of networks the system has previously established a connection with. These items do not appear to be purged unless performed by the user. If removed using the GUI shown, the items will be removed from the property list.

Some of the attributes available for each network include:

- Captive (that pop-up screen you get in hotels and restaurants)
- When the system last connected to the network, stored in local system time
- Network SSID
- Automatic logon

These networks can help an investigator determine where a system might have traveled to if the network SSIDs are unique, such as a restaurant or hotel name.

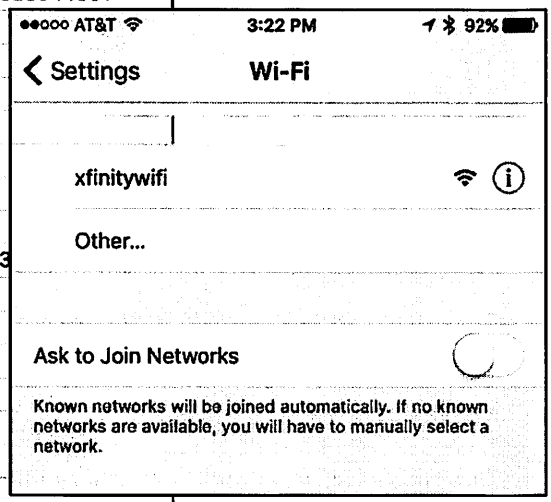
Note: This file will only be available if the system has a wireless network card available.

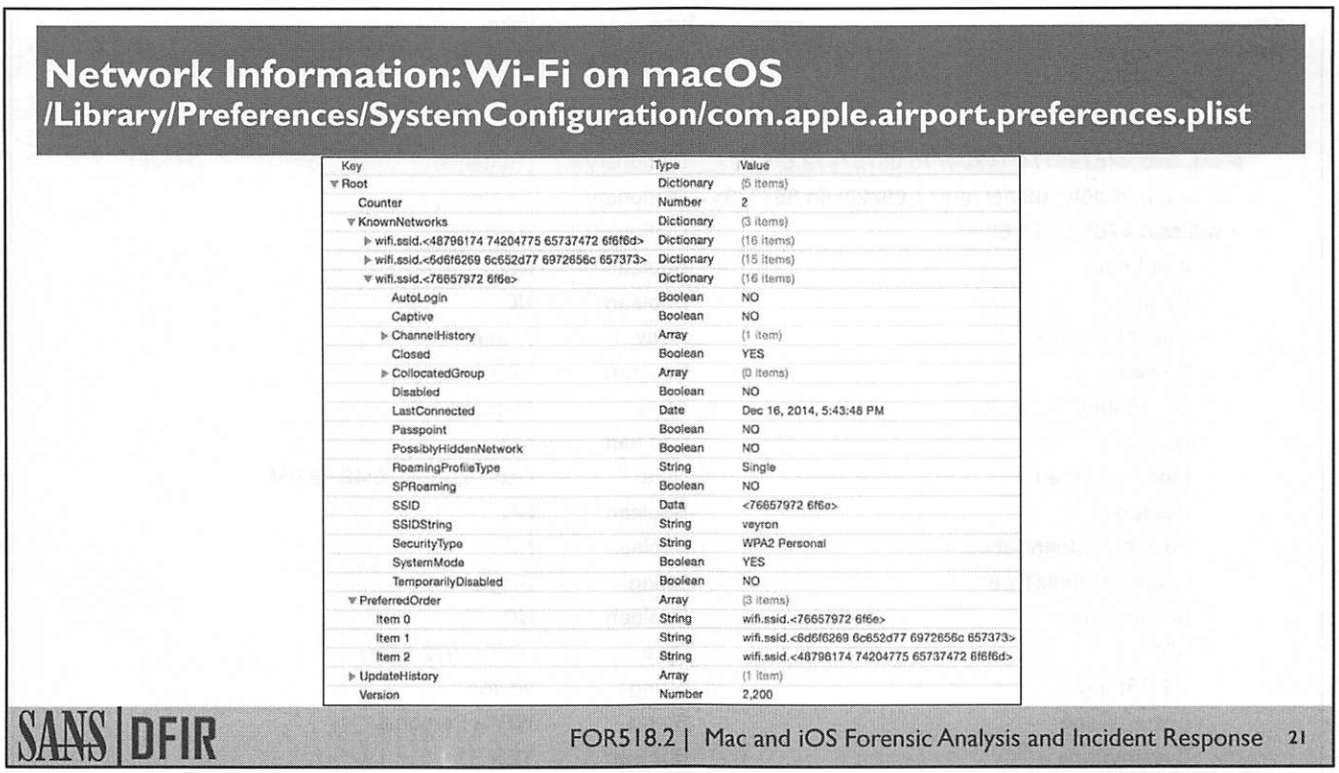
By default, these are stored in a chronological format (Item 0 is the oldest); however, the user can use the GUI to change the order in which they attempt to connect, which will change the order in the property list file.

An example of a Wi-Fi network on iOS is the `com.apple.wifi.plist` file.

While on, Mac users can remove and modify Wi-Fi access points, but it is not as easy on iOS. Known networks are populated when iOS devices join them and are difficult to remove from the list. The user can select "Forget This Network", but only when connected to the network.

▽ List of known networks	Array	(20 items)
▽ Item 0	Dictionary	(37 items)
lastAutoJoined	Date	Dec 14, 2016, 10:50:00 AM
▷ RATES	Array	(8 items)
UserDirected	Boolean	NO
▷ networkKnownBSSListKey	Array	(2 items)
▷ RSN_IE	Dictionary	(4 items)
SCAN_RESULT_FROM_PROBE_RSP	Boolean	YES
SSID	Data	<464f5235 3138>
SSID_STR	String	FOR518
Strength	Number	0.581395328044891
80211W_ENABLED	Boolean	YES
CAPABILITIES	Number	273
BEACON_INT	Number	20
AGE	Number	21
SNR	Number	14
ASSOC_FLAGS	Number	1
▷ EXT_CAPS	Dictionary	(1 item)
ScaledRSSI	Number	0.5813953
FT_ENABLED	Boolean	YES
NOISE	Number	-88
ORIG_AGE	Number	21
PHY_MODE	Number	144
RSSI	Number	-74
FAST_ENTERPRISE_NETWORK_SUPPO...	Boolean	YES
▷ QBSS_LOAD_IE	Dictionary	(3 items)
BSSID	String	1c:b9:c4:3c:1b:c
▷ 80211D_IE	Dictionary	(1 item)
CHANNEL	Number	108
AP_MODE	Number	2
▷ VHT_CAPS_IE	Dictionary	(2 items)
ScaledRate	Number	1
networkUsage	Number	30,510.0310333371
CHANNEL_WIDTH	Number	80
IE	Data	<07125553 20240424 340
CHANNEL_FLAGS	Number	1,040
▷ HT_CAPS_IE	Dictionary	(6 items)
enabled	Boolean	YES
WiFiNetworkPasswordModificationDate	Date	Dec 9, 2016, 5:58:29 PM





On 10.10+ systems, the `com.apple.airport.preferences.plist` property list file contains similar information; however, it is organized a bit differently.

The keys under `KnownNetworks` contain keys named `wifi.ssid.<hex>`. The `<hex>` is the hex representation of the network's SSID name.

The `PreferredOrder` key contains the order in which access points should be used, in order of preference. Item 0 is the most preferred.

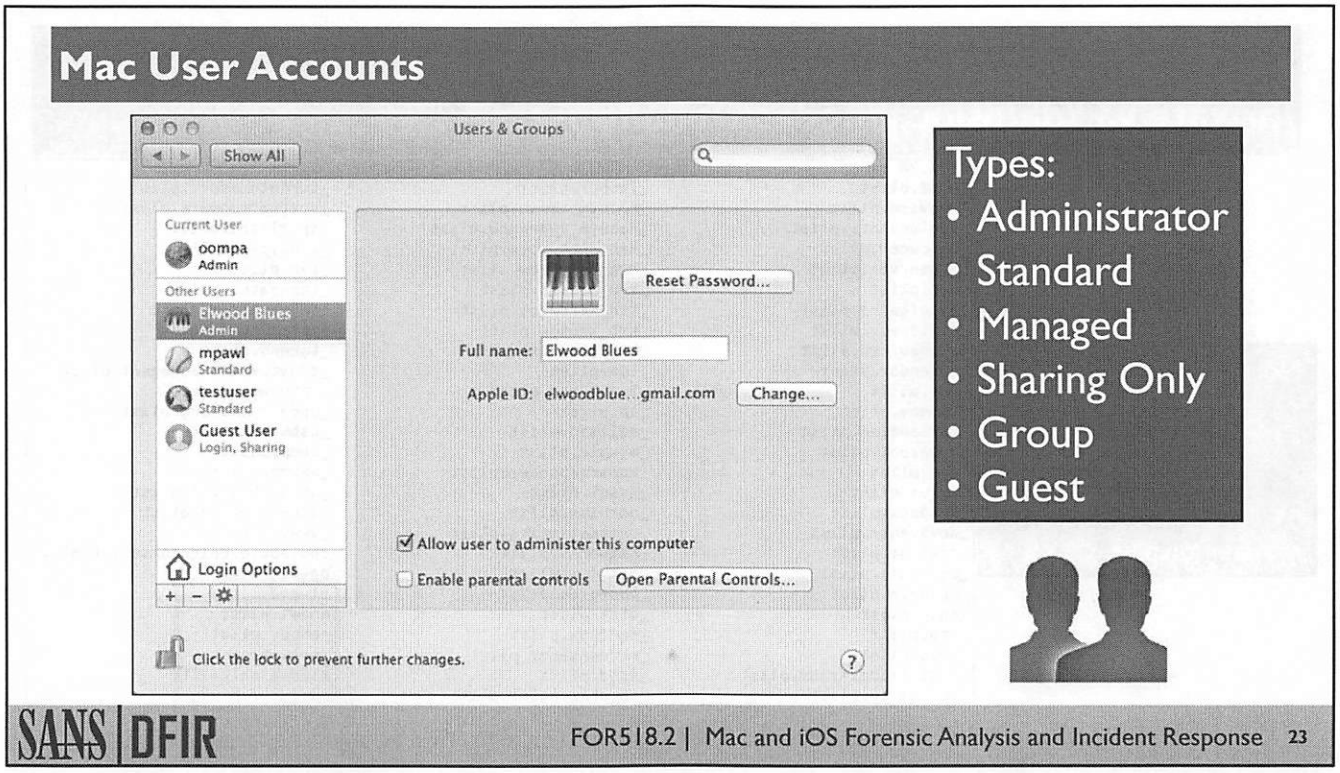
To determine which one of the listed networks is the "home" network, such as the system owner's home access point or the company enterprise Wi-Fi, analysis can be done on the airport property list file `com.apple.airport.preferences.plist` or the `system.log`. This is assuming the user is using Wi-Fi versus a wired connection.

The airport preferences file should at least have one `Item` key in the list if the wireless network adapter was used. When a system is set up, the first item, or `Item 0,` is populated. Most users set up their new laptop or device in their own homes and offices. Security of these access points should (hopefully) be secured and not listed as "`OPEN`", as with most Wi-Fi hotspots, such as the local coffee shop.

The `system.log` file can be searched for the "`airportd`" term to determine where a device has established the most connections. Many users tend to use the device in one location more than any other, such as their home or office.

This is not exactly a scientific method to determine a home network but can help in determining the user profile of a system user.

Key	Type	Value
▽ Root	Dictionary	(5 items)
Counter	Number	2
▽ KnownNetworks	Dictionary	(3 items)
▷ wifi.ssid.<48796174 74204775 65737472 6f6f6d>	Dictionary	(16 items)
▷ wifi.ssid.<6d6f6269 6c652d77 6972656c 657373>	Dictionary	(15 items)
▽ wifi.ssid.<76657972 6f6e>	Dictionary	(16 items)
AutoLogin	Boolean	NO
Captive	Boolean	NO
▷ ChannelHistory	Array	(1 item)
Closed	Boolean	YES
▷ CollocatedGroup	Array	(0 items)
Disabled	Boolean	NO
LastConnected	Date	Dec 16, 2014, 5:43:48 PM
Passpoint	Boolean	NO
PossiblyHiddenNetwork	Boolean	NO
RoamingProfileType	String	Single
SPRoaming	Boolean	NO
SSID	Data	<76657972 6f6e>
SSIDString	String	veyron
SecurityType	String	WPA2 Personal
SystemMode	Boolean	YES
TemporarilyDisabled	Boolean	NO
▽ PreferredOrder	Array	(3 items)
Item 0	String	wifi.ssid.<76657972 6f6e>
Item 1	String	wifi.ssid.<6d6f6269 6c652d77 6972656c 657373>
Item 2	String	wifi.ssid.<48796174 74204775 65737472 6f6f6d>
▷ UpdateHistory	Array	(1 item)
Version	Number	2,200



Information about the user accounts is useful when trying to determine which user did what. The 'Users & Groups' preference panel shown above allows a user to add or remove users, change passwords, change their information, enable parental controls, and perform administrative functions.

Each user is associated with a particular account type:

- Administrator
- Standard
- Managed with Parental Controls
- Sharing Only
- Group
- Guest

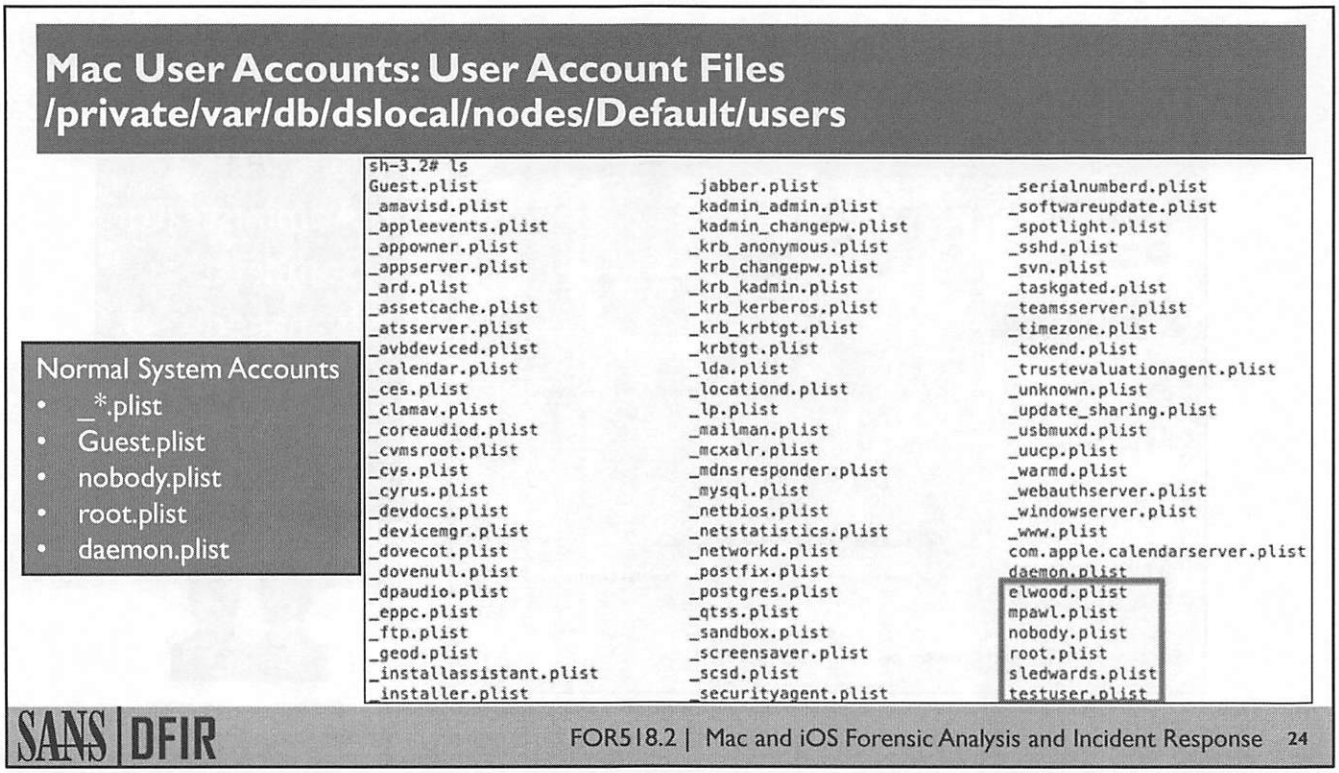
The **Administrator** has full administrative access to the system, while a **Standard** user has limited administrative privileges to the system; they can install software and change their own account preferences.

An account that is **Managed with Parental Controls** can be restricted from using certain applications, may have limited exposure to inappropriate content, or may have time usage restrictions.

A **Sharing Only** account can be used by networked users to access shared files.

A **Group** can be created to keep track of more complex user types, such as in an enterprise environment.

A **Guest user** can log in (without a password) and use the computer temporarily. If configured, a Guest user may be able to connect to shared folders or have parental controls enabled. If the system uses FileVault, the Guest user will only be able to access Safari. After a Guest user logs out, all data in the Guest home directory is deleted. Guest users on 10.7 and below are similar. One exception using 10.7: when FileVault is enabled, Guest users cannot log on.



Each user and group has a property list file containing data about their respective user. The users are located in the `/private/var/db/dslocal/nodes/Default/users` directory, while the groups are detailed in the `/private/var/db/dslocal/nodes/Default/groups` directory.

The property list files may be binary or XML, depending on the OS version.

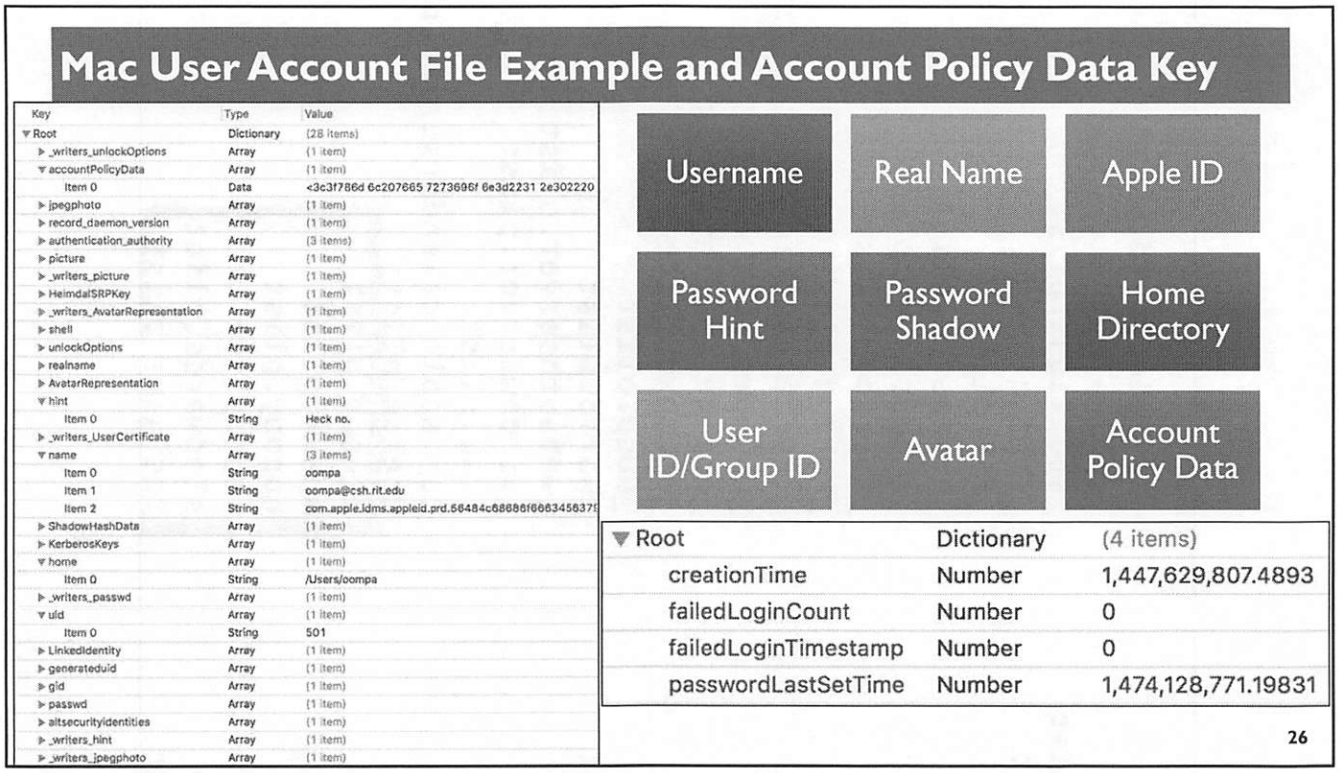
- 10.6: XML
- 10.7+: binary

Access to this directory requires root privileges.

It is worth noting that users who use Open Directory (similar to Active Directory) will not have a user plist in this directory.

The `_*.plist` are for services and groups, while `daemon.plist`, `root.plist`, and `nobody.plist` are also default accounts.

```
sh-3.2# ls
Guest.plist                     _jabber.plist                   _serialnumberd.plist
_amavisd.plist                  _kadmin_admin.plist             _softwareupdate.plist
_appleevents.plist              _kadmin_changepw.plist          _spotlight.plist
_appowner.plist                 _krb_anonymous.plist            _sshd.plist
_appserver.plist                _krb_changepw.plist             _svn.plist
_ard.plist                      _krb_kadmin.plist               _taskgated.plist
_assetcache.plist               _krb_kerberos.plist             _teamsserver.plist
_atsserver.plist                _krb_krbtgt.plist               _timezone.plist
_avbdeviced.plist               _krbtgt.plist                   _tokend.plist
_calendar.plist                 _lda.plist                      _trustevaluationagent.plist
_ces.plist                      _locationd.plist                _unknown.plist
_clamav.plist                   _lp.plist                       _update_sharing.plist
_coreaudiod.plist               _mailman.plist                  _usbmuxd.plist
_cvmsroot.plist                 _mcxalr.plist                   _uucp.plist
_cvs.plist                      _mdnsresponder.plist            _warmd.plist
_cyrus.plist                    _mysql.plist                    _webauthserver.plist
_devdocs.plist                  _netbios.plist                  _windowserver.plist
_devicemgr.plist                _netstatistics.plist            _www.plist
_dovecot.plist                  _networkd.plist                 com.apple.calendarserver.plist
_dovenull.plist                 _postfix.plist                  daemon.plist
_dpaudio.plist                  _postgres.plist                 elwood.plist
_eppc.plist                     _qtss.plist                     mpawl.plist
_ftp.plist                      _sandbox.plist                  nobody.plist
_geod.plist                     _screensaver.plist              root.plist
_installassistant.plist         _scsd.plist                     sledwards.plist
_installer.plist                _securityagent.plist            testuser.plist
```



Each user property list contains data about the specific user account. It can contain a variety of different information and may sometimes be different between different versions of macOS. This is an example from a system on 10.13, where the account has been associated with an Apple ID.

The bottom screenshot shows the account policy data from the user's account file, which was extracted from the accountPolicyData key. This plist contains the account creation time and when the password was last set in Unix epoch time. The Failed Login Count and Failed Login Timestamp do not appear to be used or updated.

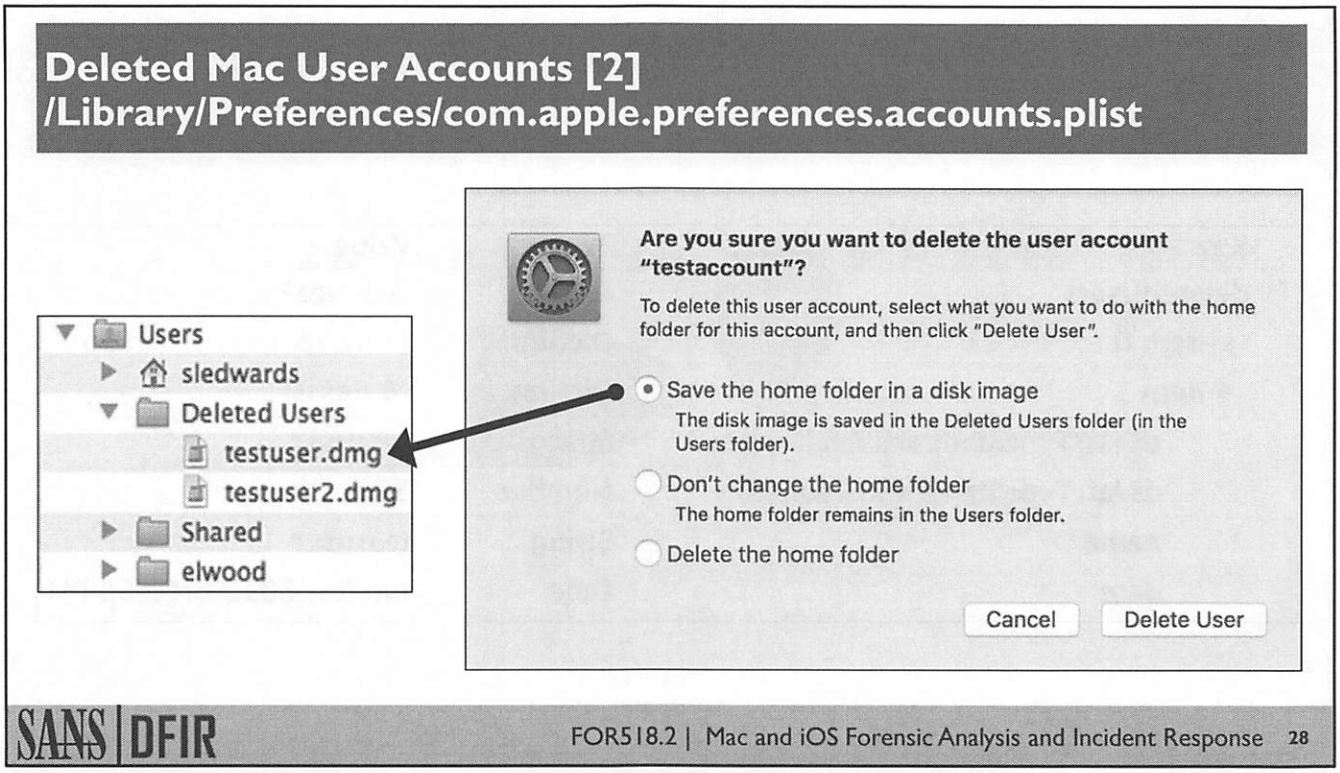
Deleted Mac User Accounts [1]
/Library/Preferences/com.apple.preferences.accounts.plist

Key	Type	Value
▼ deletedUsers	Array	(2 items)
▶ Item 0	Diction...	(4 items)
▼ Item 1	Diction...	(4 items)
dsAttrTypeStandard:RealName	String	testuser
dsAttrTypeStandard:UniqueID	Number	502
name	String	testuser
date	Date	Jun 13, 2012 8:41:58 PM

The deleted users are shown in the `com.apple.preferences.accounts.plist` file under the `deletedUsers` key. This property list is located in the `/Library/Preferences/` directory.

This key contains:

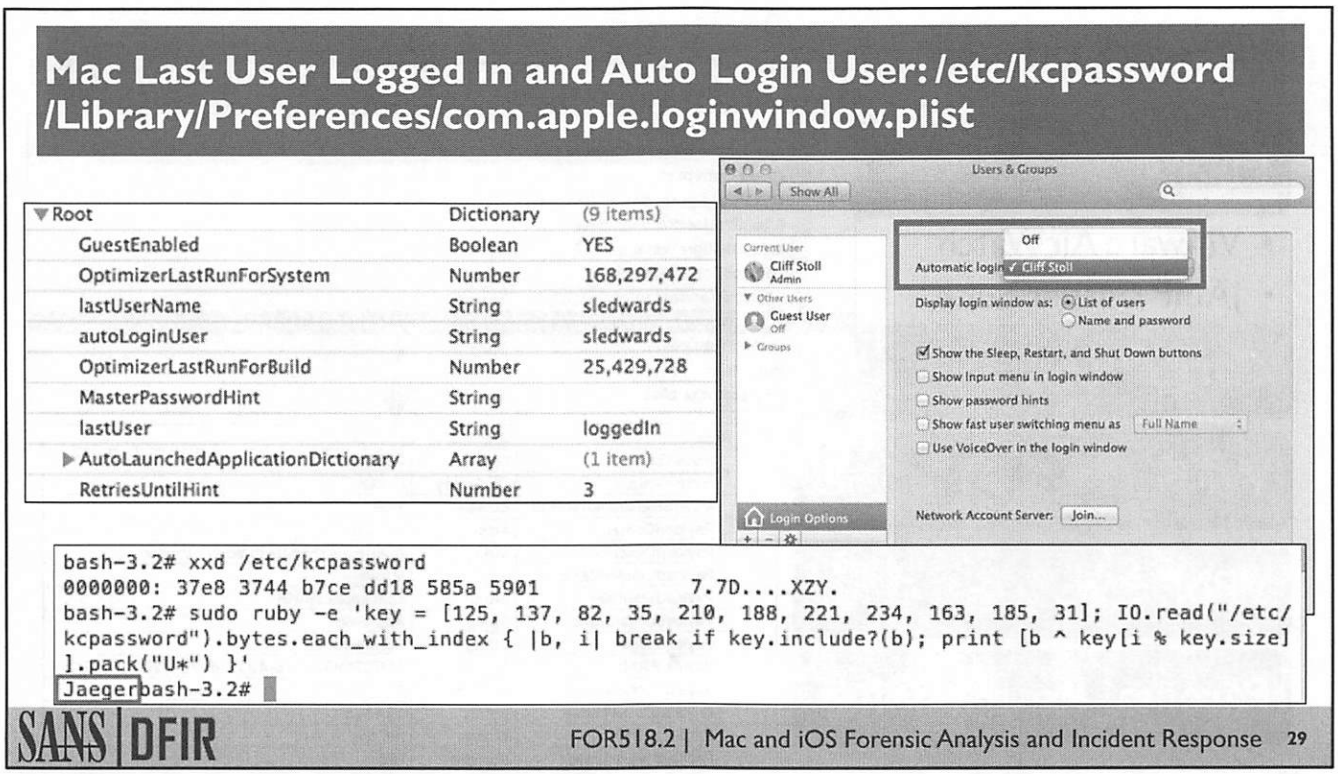
- Deleted user's "Real Name"
- UID
- Username
- Deletion date (local system time)



There are three options available when a user account is deleted:

1. "`Save the home folder in a disk image`": This default option archives the user's home directory and saves it in a disk image file (DMG). The inset screenshot shows a couple of deleted user accounts, saved to DMG files, which are then moved to the `/Users/Deleted Users/` directory.
2. "`Don't change the home folder`": The home folder does not get archived; it stays in place.
3. "`Delete the home folder`": The user's home directory is removed.

When the user accounts are deleted, the user's plist in the `/private/var/db/dslocal/nodes/Default/users/` directory is also removed.



The `com.apple.loginwindow.plist` property list located in the `/Library/Preferences/` directory contains:

- `lastUser`: If the user is currently logged in (assuming the system was imaged live).
- `autoLoginUser`: The Auto Login user (if configured).
- `lastUserName`: Last logged in user.
- `RetriesUntilHint`: The number of times before a password hint is given.
- `GuestEnabled`: Guest Account Status.
- `MasterPasswordHint`: If a master password has been configured, a hint may have also been configured.

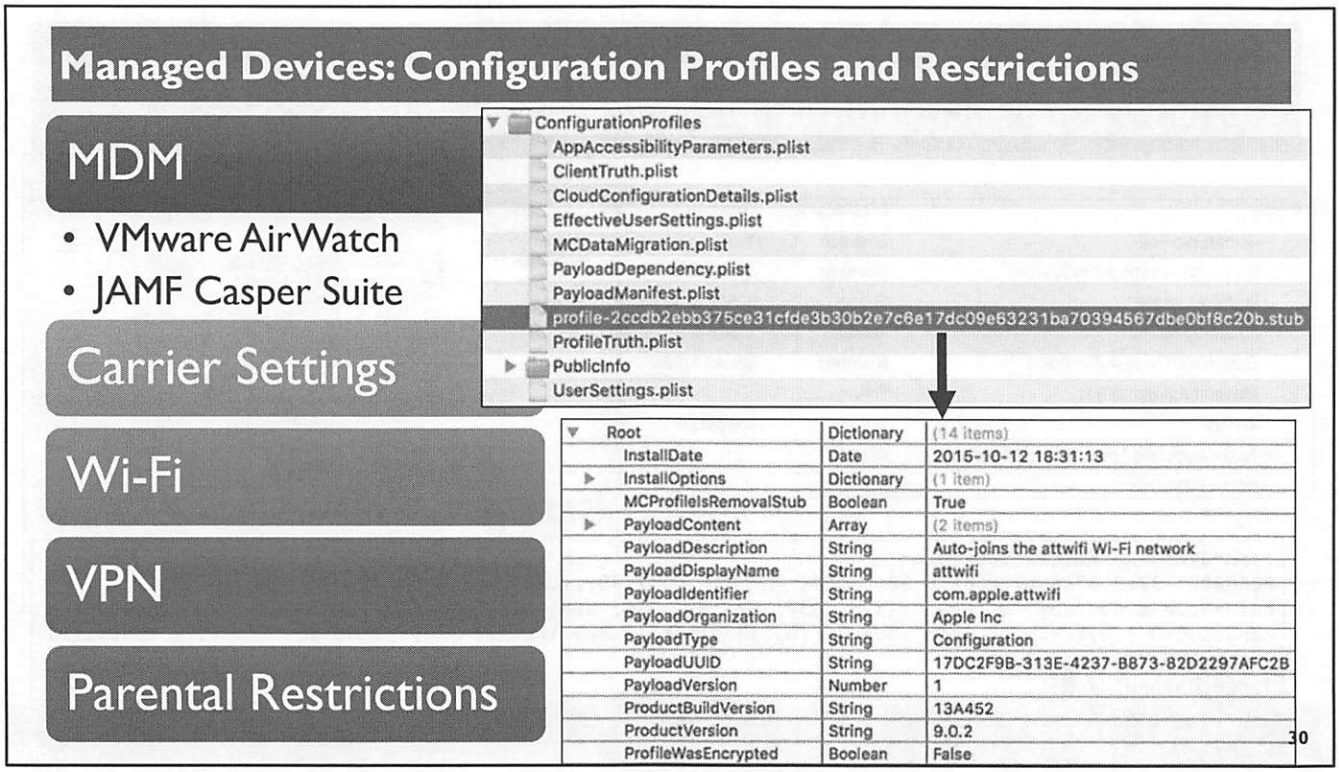
A user may choose the ability to have the system automatically log them on using the "`Automatic login`" selection window in the 'Users & Groups' preferences pane.

The user's password is then XOR'd with a multi-byte key and stored in `/etc/kcpassword`. To decode this password, use the Ruby script below from https://gist.github.com/opshope/32f65875d45215c3677d.

```
sudo ruby -e 'key = [125, 137, 82, 35, 210, 188, 221, 234, 163, 185, 31];
IO.read("/etc/kcpassword").bytes.each_with_index { |b, i| break if
key.include?(b); print [b ^ key[i % key.size]].pack("U*") }'
```

Automatic login is not available for a system's User FileVault or accounts that use iCloud credentials to log in. It may be possible to find this file on the system even after Automatic login was turned off, however.

References:
`http://www.brock-family.org/gavin/perl/kcpassword.html`
`https://apple.stackexchange.com/questions/50652/does-activating-auto-login-compromise-secure-password-storage`
`https://support.apple.com/en-us/HT201476`



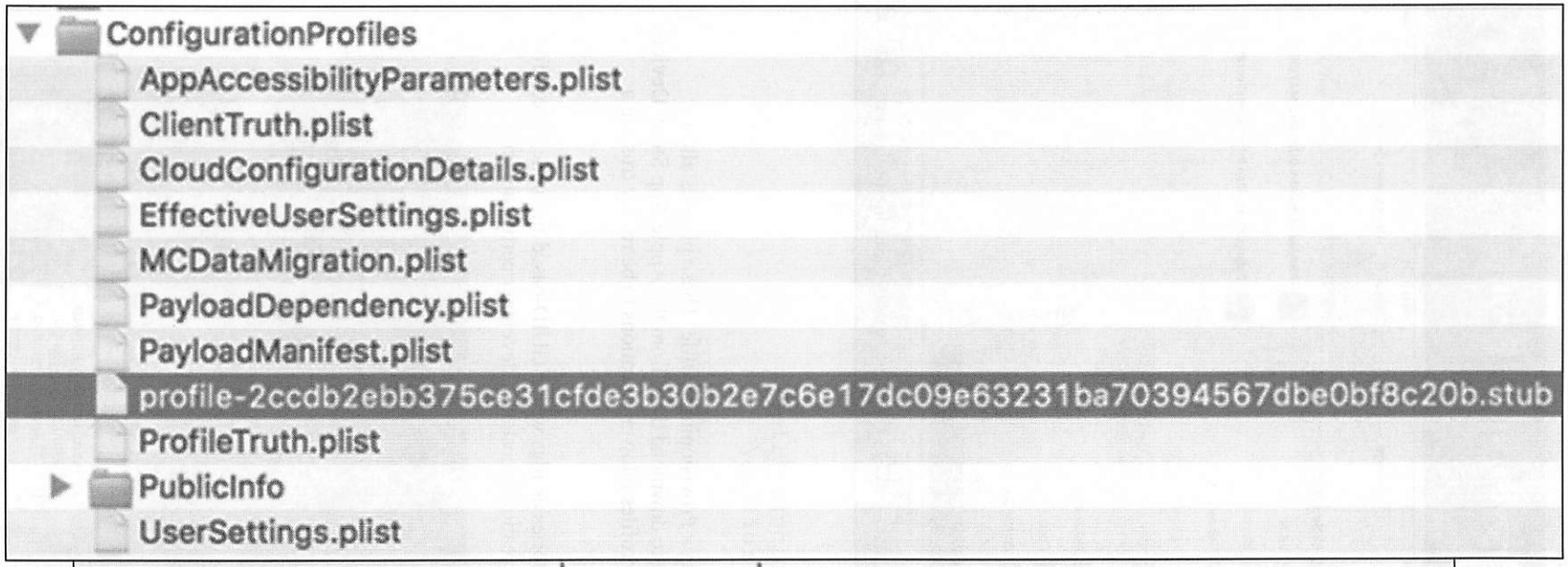
macOS: /private/var/db/ConfigurationProfiles/
Physical: `/private/var/mobile/Library/ConfigurationProfiles/`
Backup/File System: `/mobile/Library/ConfigurationProfiles/` (or `/UnknownDomain/Library/ConfigurationProfiles/`)

Devices can be managed (or supervised). They may be managed through your enterprise Mobile Device Management system or settings pushed to you by your corporation or carrier (as well as many other ways).

Managed devices will have a "profile" installed on the device. This profile can be reviewed to see what actions/limitations it can use. These profiles are found in the ConfigurationProfiles directory with a filename similar to "`profile-<some_hash>.stub`". The example above shows a standard profile on AT&T devices that allows these devices to auto-join "attwifi" access points. These profiles can be accessed through the GUI in the following path: `Settings | General | Profiles` (or `Device Management`). Note: Hidden profiles, like the AT&T one above, will not show up, as it is configured as a "hidden" profile. If no "non-hidden" profiles are installed, the menu option does not exist.

Restrictions (`Settings | General | Restrictions`) may also be used to limit certain applications, services, app purchases, media content, privacy restrictions, etc. This is similar to parental-type controls found on other devices and systems. Devices that have restrictions may have those restrictions recorded in the `UserSettings.plist`, `EffectiveUserSettings.plist`, and `PublicEffectiveUserSettings.plist` files in the `ConfigurationProfiles` directory.

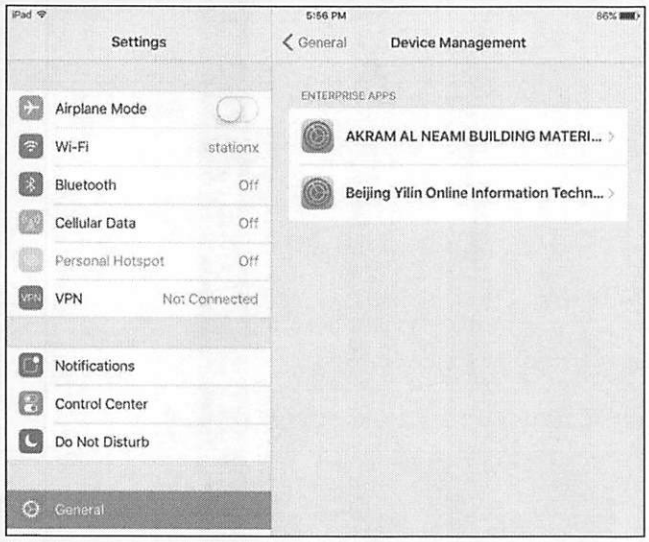
Reference:
`https://developer.apple.com/business/documentation/Configuration-Profile-Reference.pdf`



Root	Dictionary	(14 items)
InstallDate	Date	2015-10-12 18:31:13
InstallOptions	Dictionary	(1 item)
MCProfileIsRemovalStub	Boolean	True
PayloadContent	Array	(2 items)
PayloadDescription	String	Auto-joins the attwifi Wi-Fi network
PayloadDisplayName	String	attwifi
PayloadIdentifier	String	com.apple.attwifi
PayloadOrganization	String	Apple Inc
PayloadType	String	Configuration
PayloadUUID	String	17DC2F9B-313E-4237-B873-82D2297AFC2B
PayloadVersion	Number	1
ProductBuildVersion	String	13A452
ProductVersion	String	9.0.2
ProfileWasEncrypted	Boolean	False

iOS Managed Devices: Enterprise Provisioning Profiles /MobileDevice/ProvisioningProfiles/

- Enterprise deployments
- Allows apps to be run without downloading from Apple App Store
- Malware on non-jailbroken devices (WireLurker)
- Jailbroken devices (Pangu)



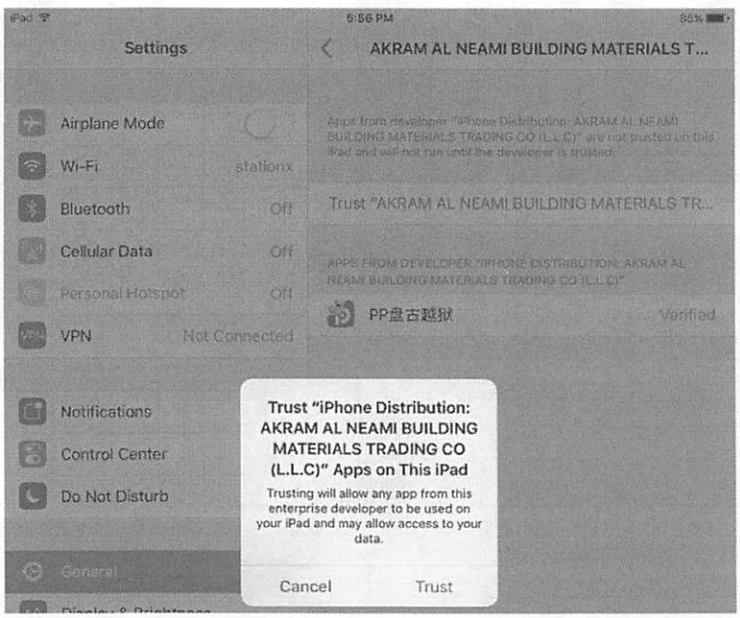
Physical: `/private/var/MobileDevice/ProvisioningProfiles/`
Backup/File System: `/MobileDevice/ProvisioningProfiles/`

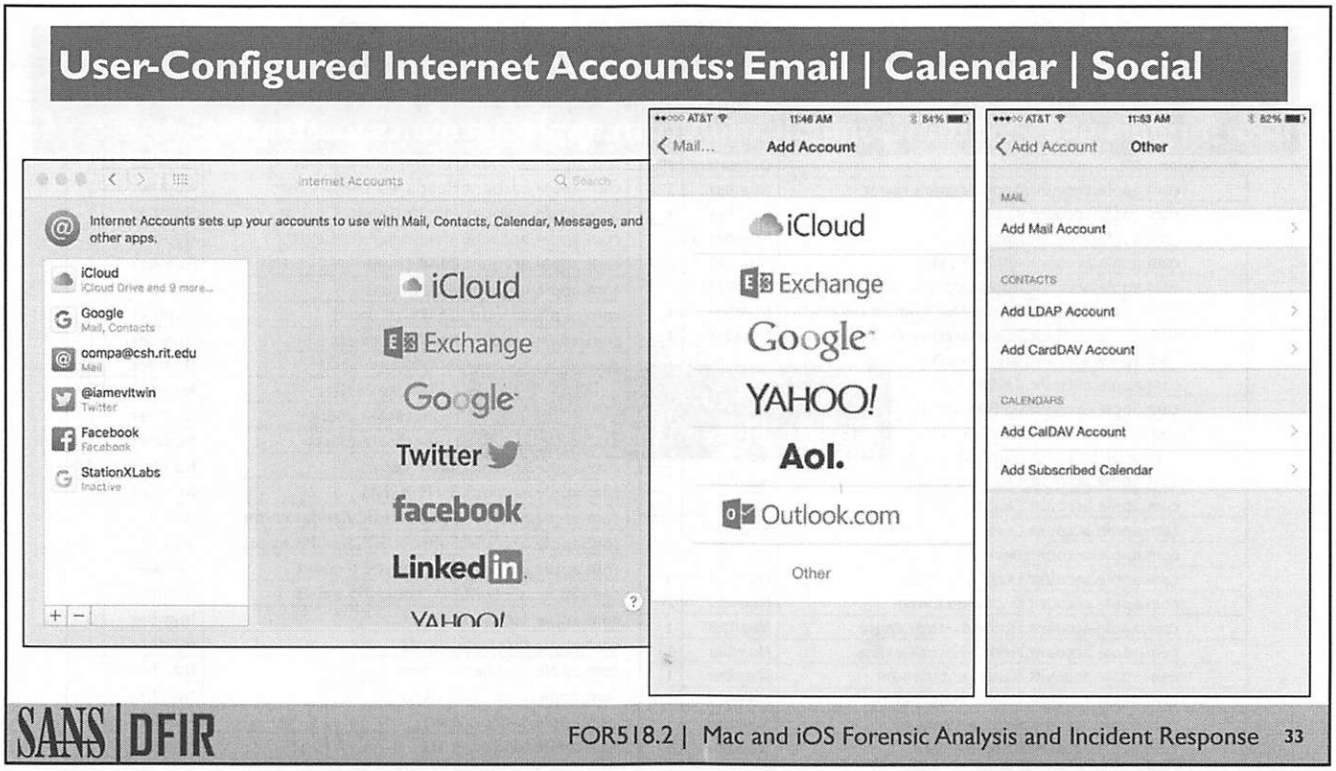
Devices can run applications distributed by an enterprise using a Provisioning Profile. These profiles allow applications to be distributed and run by users without having to download them from the Apple App Store. Devices do not need to be jailbroken to run these applications. These profiles allow applications to be run by trusting a certificate that is downloaded by the user onto the device.

These certificates are stored in the ProvisioningProfiles directories as files with a GUID-based filename. These files contain information such as the enterprise name, bundle ID, creation timestamp, developer certificate, and entitlements.

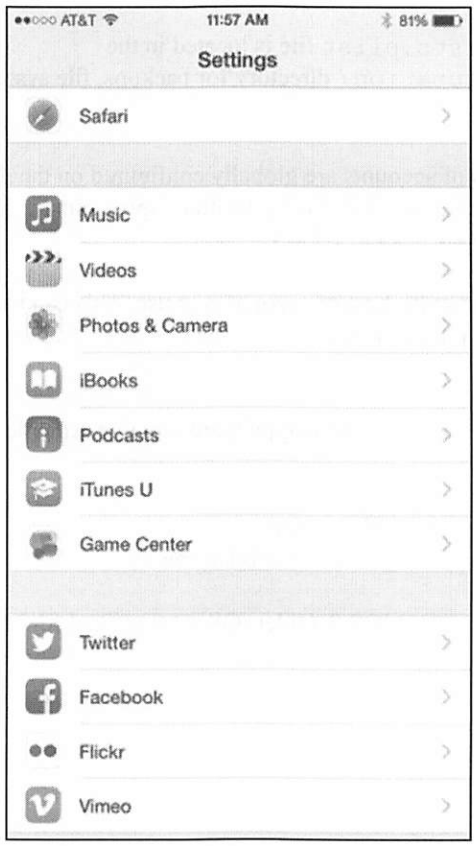
References:
`https://www.paloaltonetworks.com/content/dam/pan/en_US/assets/pdf/reports/Unit_42/unit42-wirelurker.pdf`

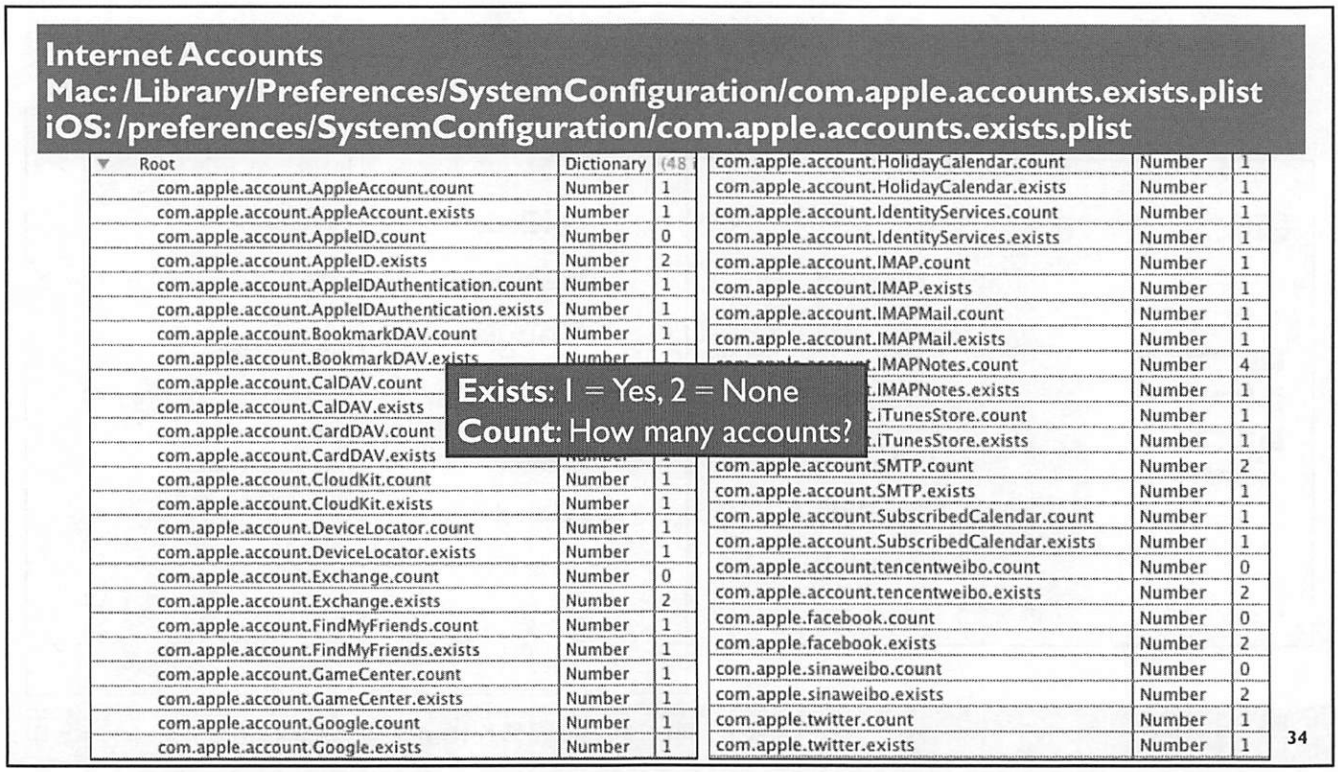
(Dev Account Required)
`https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingProfiles/MaintainingProfiles.html`





Knowing what accounts a user has configured with their device can be good investigative information to know. Mac and iOS devices both have a single area of configured account information that may include email, calendar, and/or social media accounts.





Root	Dictionary	(48
com.apple.account.AppleAccount.count	Number	1
com.apple.account.AppleAccount.exists	Number	1
com.apple.account.AppleID.count	Number	0
com.apple.account.AppleID.exists	Number	2
com.apple.account.AppleIDAuthentication.count	Number	1
com.apple.account.AppleIDAuthentication.exists	Number	1
com.apple.account.BookmarkDAV.count	Number	1
com.apple.account.BookmarkDAV.exists	Number	1
com.apple.account.CalDAV.count		
com.apple.account.CalDAV.exists		
com.apple.account.CardDAV.count		
com.apple.account.CardDAV.exists		
com.apple.account.CloudKit.count	Number	1
com.apple.account.CloudKit.exists	Number	1
com.apple.account.DeviceLocator.count	Number	1
com.apple.account.DeviceLocator.exists	Number	1
com.apple.account.Exchange.count	Number	0
com.apple.account.Exchange.exists	Number	2
com.apple.account.FindMyFriends.count	Number	1
com.apple.account.FindMyFriends.exists	Number	1
com.apple.account.GameCenter.count	Number	1
com.apple.account.GameCenter.exists	Number	1
com.apple.account.Google.count	Number	1
com.apple.account.Google.exists	Number	1

com.apple.account.HolidayCalendar.count	Number	1
com.apple.account.HolidayCalendar.exists	Number	1
com.apple.account.IdentityServices.count	Number	1
com.apple.account.IdentityServices.exists	Number	1
com.apple.account.IMAP.count	Number	1
com.apple.account.IMAP.exists	Number	1
com.apple.account.IMAPMail.count	Number	1
com.apple.account.IMAPMail.exists	Number	1
.IMAPNotes.count	Number	4
.IMAPNotes.exists	Number	1
.iTunesStore.count	Number	1
.iTunesStore.exists	Number	1
com.apple.account.SMTP.count	Number	2
com.apple.account.SMTP.exists	Number	1
com.apple.account.SubscribedCalendar.count	Number	1
com.apple.account.SubscribedCalendar.exists	Number	1
com.apple.account.tencentweibo.count	Number	0
com.apple.account.tencentweibo.exists	Number	2
com.apple.facebook.count	Number	0
com.apple.facebook.exists	Number	2
com.apple.sinaweibo.count	Number	0
com.apple.sinaweibo.exists	Number	2
com.apple.twitter.count	Number	1
com.apple.twitter.exists	Number	1

Mac: `/Library/Preferences/SystemConfiguration/com.apple.accounts.exists.plist`
iOS:
`[/private/var]/preferences/SystemConfiguration/com.apple.accounts.exists.plist`

The `com.apple.accounts.exists.plist` file is located in the `/preferences/SystemConfiguration/` directory for backups, file system extractions, and physical images.

This property list shows which types of accounts are globally configured on the device. These can include email (IMAP/SMTP/Exchange), Google, Calendar, Facebook, Twitter, Apple, Sinaweibo, etc. Each account has two associated keys: a "`Count`" key and an "`Exists`" key.

The "`Exists`" key shows if that particular type of account is in use. This can have a few different options:

1. At least one account is set up for this type
2. No accounts of this type

The "`Count`" key shows how many accounts of this type there are. For example, in the screenshot above, this device has:

- Two SMTP email accounts
- One Apple, Google, Twitter, and IMAP account
- No Exchange, Facebook, Sinaweibo, Tencentweibo accounts

The example above shows a device running iOS 8. Older iOS versions had fewer accounts listed—mainly AppleID, Facebook, and Sinaweibo.

Root	Dictionary	(48
com.apple.account.AppleAccount.count	Number	1
com.apple.account.AppleAccount.exists	Number	1
com.apple.account.AppleID.count	Number	0
com.apple.account.AppleID.exists	Number	2
com.apple.account.AppleIDAuthentication.count	Number	1
com.apple.account.AppleIDAuthentication.exists	Number	1
com.apple.account.BookmarkDAV.count	Number	1
com.apple.account.BookmarkDAV.exists	Number	1
com.apple.account.CalDAV.count	Number	2
com.apple.account.CalDAV.exists	Number	1
com.apple.account.CardDAV.count	Number	2
com.apple.account.CardDAV.exists	Number	1
com.apple.account.CloudKit.count	Number	1
com.apple.account.CloudKit.exists	Number	1
com.apple.account.DeviceLocator.count	Number	1
com.apple.account.DeviceLocator.exists	Number	1
com.apple.account.Exchange.count	Number	0
com.apple.account.Exchange.exists	Number	2
com.apple.account.FindMyFriends.count	Number	1
com.apple.account.FindMyFriends.exists	Number	1
com.apple.account.GameCenter.count	Number	1
com.apple.account.GameCenter.exists	Number	1
com.apple.account.Google.count	Number	1
com.apple.account.Google.exists	Number	1
com.apple.account.HolidayCalendar.count	Number	1
com.apple.account.HolidayCalendar.exists	Number	1
com.apple.account.IdentityServices.count	Number	1
com.apple.account.IdentityServices.exists	Number	1
com.apple.account.IMAP.count	Number	1
com.apple.account.IMAP.exists	Number	1
com.apple.account.IMAPMail.count	Number	1
com.apple.account.IMAPMail.exists	Number	1
com.apple.account.IMAPNotes.count	Number	4
com.apple.account.IMAPNotes.exists	Number	1
com.apple.account.iTunesStore.count	Number	1
com.apple.account.iTunesStore.exists	Number	1
com.apple.account.SMTP.count	Number	2
com.apple.account.SMTP.exists	Number	1
com.apple.account.SubscribedCalendar.count	Number	1
com.apple.account.SubscribedCalendar.exists	Number	1
com.apple.account.tencentweibo.count	Number	0
com.apple.account.tencentweibo.exists	Number	2
com.apple.facebook.count	Number	0
com.apple.facebook.exists	Number	2
com.apple.sinaweibo.count	Number	0
com.apple.sinaweibo.exists	Number	2
com.apple.twitter.count	Number	1
com.apple.twitter.exists	Number	1

Configured Internet Accounts [iOS 7+, OS X 11+]
Accounts3.sqlite | Accounts4.sqlite

```
1  select
2  zaccount.z_pk,
3  zaccounttype.zaccounttypedescription,
4  zaccount.zusername,
5  zaccount.zaccountdescription,
6  zaccount.zparentaccount,
7  datetime(zaccount.zdate+978307200,'UNIXEPOCH','UTC') as Timestamp,
8  zaccount.zidentifier
9  from zaccount
10 left join zaccounttype on zaccounttype == zaccounttype.z_pk
```

	Z_PK	ZACCOUNTTYPEDESCRIPTION	ZUSERNAME	ZACCOUNTDESCRIPTION	ZPARENTACCOUNT	Timestamp	ZIDENTIFIER
1	1	Twitter	iamevltwin	@iamevltwin	NULL	NULL	0FF46713-0EDD-478B-A992-1DB3C234F0D4
2	3	iCloud	oompa@csh.rit.edu	iCloud	NULL	2013-09-21 00:38:58	921FF895-8377-4ADB-B012-31ABB4761243
3	4	Messages	oompa@csh.rit.edu	NULL	NULL	2013-09-21 00:38:58	E9C73074-2B21-4CC4-A5A9-FAB6882CABEB
4	5	Device Locator	oompa@csh.rit.edu	NULL	3	2013-09-21 00:38:59	E319CBAA-3FD9-4773-81AD-A94CF4E7FA75
5	6	IMAPMail	NULL	NULL	3	2013-09-21 00:38:59	B3ED629B-5D07-442A-8088-D9D5B8C7EEF4
6	7	SMTP	oompa	SMTP:oompa@mail.csh.rit.edu	NULL	2013-09-21 00:38:59	CB6B7B0A-874A-403E-AD3F-59389FDFFCF5
7	8	IMAP	oompa	CSH	NULL	2013-09-21 00:38:59	AD3C55BC-6C88-454E-8CC5-CB70FF40891F

Mac:

- 10.11: `~/Library/Accounts/Accounts3.sqlite`
- 10.12+: `~/Library/Accounts/Accounts4.sqlite`

iOS: `[/private/var]/mobile/Library/Accounts/Accounts3.sqlite`

The `Accounts3.sqlite or Accounts4.sqlite` database keeps track of which accounts (Mail, Messages, Communication, Calendar, Social Media, etc.) are integrated into the Mac or iOS system.

The `ZACCOUNTTYPE` table contains the types of accounts that can be configured globally on the device. The `Z_PK` column contains the identification number for each account. This table also includes the account description, credential types, and the identifier for the account.

The `ZACCOUNT` table contains specific user account information. We can match up the `Z_PK` key from the `ZACCOUNTTYPE` table with the `ZACCOUNTTYPE` column of this table to determine what account this information is associated with.

The `ZACCOUNT` table in the example above shows the following columns of interest:

- `Z_PK` = Primary Key
- `ZACCOUNTTYPEDESCRIPTION` from `ZACCOUNTTYPE` Table = Type of Account
- `ZUSERNAME` = Account Username
- `ZACCOUNTDESCRIPTION` from `ZACCOUNT` Table = Account Description, more specific
- `ZPARENTACCOUNT` = Parent Account Type
- `ZDATE` = Account Setup Timestamp, Mac Epoch timestamp format (labeled in the query as "Timestamp")
- `ZIDENTIFIER` = Account GUID

```
1  select
2  zaccount.z_pk,
3  zaccounttype.zaccounttypedescription,
4  zaccount.zusername,
5  zaccount.zaccountdescription,
6  zaccount.zparentaccount,
7  datetime(zaccount.zdate+978307200,'UNIXEPOCH','UTC') as Timestamp,
8  zaccount.zidentifier
9  from zaccount
10 left join zaccounttype on zaccounttype == zaccounttype.z_pk
```

	Z_PK	ZACCOUNTTYPEDESCRIPTION	ZUSERNAME	ZACCOUNTDESCRIPTION	ZPARENTACCOUNT	Timestamp	ZIDENTIFIER
1	1	Twitter	iamevitwin	@iamevitwin	*NULL*	*NULL*	0FF46713-0EDD-478B-A992-1DB3C234F0D4
2	3	iCloud	oompa@csh.rit.edu	iCloud	*NULL*	2013-09-21 00:38:58	921FF895-8377-4ADB-B012-31ABB4761243
3	4	Messages	oompa@csh.rit.edu	*NULL*	*NULL*	2013-09-21 00:38:58	E9C73074-2B21-4CC4-A5A9-FAB6882CABEB
4	5	Device Locator	oompa@csh.rit.edu	*NULL*	3	2013-09-21 00:38:59	E319CBAA-3FD9-4773-81AD-A94CF4E7FA75
5	6	IMAPMail	*NULL*	*NULL*	3	2013-09-21 00:38:59	B3ED629B-5D07-442A-8088-D9D5B8C7EEF4
6	7	SMTP	oompa	SMTP:oompa@mail.csh.rit.edu	*NULL*	2013-09-21 00:38:59	CB6B7B0A-874A-403E-AD3F-59389FDFFCF5
7	8	IMAP	oompa	CSH	*NULL*	2013-09-21 00:38:59	AD3C55BC-6C88-454E-8CC5-CB70FF40891F

Internet Accounts: Configuration Details
Accounts3.sqlite | Accounts4.sqlite

```
1   select
2   zaccount.z_pk,
3   zaccounttype.zaccounttypedescription,
4   zaccount.zusername,
5   zaccount.zaccountdescription,
6   zaccountproperty.zkey,
7   zaccountproperty.zvalue
8   from zaccountproperty
9   left join zaccount on zaccount.z_pk == zaccountproperty.zowner
10  left join zaccounttype on zaccount.zaccounttype == zaccounttype.z_pk
```

```
0000 62 70 6c 69 73 74 30 30 d4 01 02 03 04 05 06 09   bplist00........
0010 0a 58 24 76 65 72 73 69 6f 6e 58 24 6f 62 6a 65   .X$versionX$obje
0020 63 74 73 59 24 61 72 63 68 69 76 65 72 54 24 74   ctsY$archiverT$t
0030 6f 70 12 00 01 86 a0 a2 07 08 55 24 6e 75 6c 6c   op........U$null
0040 5d 53 61 72 61 68 20 45 64 77 61 72 64 73 5f 10   ]Sarah Edwards_.
0050 0f 4e 53 4b 65 79 65 64 41 72 63 68 69 76 65 72   .NSKeyedArchiver
0060 d1 0b 0c 54 72 6f 6f 74 80 01 08 11 1a 23 2d 32   ...Troot.....#-2
0070 37 3a 40 4e 60 63 68 00 00 00 00 00 00 01 01 00   7:@N`ch.........
0080 00 00 00 00 00 00 0d 00 00 00 00 00 00 00 00 00   ................
0090 00 00 00 00 00 00 6a                              ......j
```

	Z_PK	ZACCOUNTTYPEDESCRIPTION	ZUSERNAME	ZACCOUNTDESCRIPTION	ZKEY	ZVALUE
1	1	Twitter	iamevltwin	@iamevltwin	user_id	BLOB
2	1	Twitter	iamevltwin	@iamevltwin	fullName	BLOB
3	3	iCloud	oompa@csh.rit.edu	iCloud	mobileMeStatus	BLOB
4	3	iCloud	oompa@csh.rit.edu	iCloud	protocolVersion	BLOB
5	3	iCloud	oompa@csh.rit.edu	iCloud	Type	BLOB
6	3	iCloud	oompa@csh.rit.edu	iCloud	Class	BLOB
7	3	iCloud	oompa@csh.rit.edu	iCloud	primaryEmail	BLOB

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 38

The specific details on each account configuration can be viewed using the SQL query above. This can include things like the email server, username, and port information in the `ZVALUE` column's BLOB information. The example above shows a binary plist file that contains the "fullName" for the twitter account `@iamevltwin`.

Different accounts will have different types of configuration data that is named by the `ZKEY` column.

```
1   select
2   zaccount.z_pk,
3   zaccounttype.zaccounttypedescription,
4   zaccount.zusername,
5   zaccount.zaccountdescription,
6   zaccountproperty.zkey,
7   zaccountproperty.zvalue
8   from zaccountproperty
9   left join zaccount on zaccount.z_pk == zaccountproperty.zowner
10  left join zaccounttype on zaccount.zaccounttype == zaccounttype.z_pk
```

	Z_PK	ZACCOUNTTYPEDESCRIPTION	ZUSERNAME	ZACCOUNTDESCRIPTION	ZKEY	ZVALUE
1	1	Twitter	iamevltwin	@iamevltwin	user_id	*BLOB*
2	1	Twitter	iamevltwin	@iamevltwin	fullName	*BLOB*
3	3	iCloud	oompa@csh.rit.edu	iCloud	mobileMeStatus	*BLOB*
4	3	iCloud	oompa@csh.rit.edu	iCloud	protocolVersion	*BLOB*
5	3	iCloud	oompa@csh.rit.edu	iCloud	Type	*BLOB*
6	3	iCloud	oompa@csh.rit.edu	iCloud	Class	*BLOB*
7	3	iCloud	oompa@csh.rit.edu	iCloud	primaryEmail	*BLOB*

```
0000  62 70 6c 69 73 74 30 30 d4 01 02 03 04 05 06 09   bplist00........
0010  0a 58 24 76 65 72 73 69 6f 6e 58 24 6f 62 6a 65   .X$versionX$obje
0020  63 74 73 59 24 61 72 63 68 69 76 65 72 54 24 74   ctsY$archiverT$t
0030  6f 70 12 00 01 86 a0 a2 07 08 55 24 6e 75 6c 6c   op........U$null
0040  5d 53 61 72 61 68 20 45 64 77 61 72 64 73 5f 10   ]Sarah Edwards_.
0050  0f 4e 53 4b 65 79 65 64 41 72 63 68 69 76 65 72   .NSKeyedArchiver
0060  d1 0b 0c 54 72 6f 6f 74 80 01 08 11 1a 23 2d 32   ...Troot.....#-2
0070  37 3a 40 4e 60 63 68 00 00 00 00 00 00 01 01 00   7:@N`ch.........
0080  00 00 00 00 00 00 0d 00 00 00 00 00 00 00 00 00   ................
0090  00 00 00 00 00 00 6a                              ......j
```

Internet Accounts: Account Authorizations (iOS)
Accounts3.sqlite | Accounts4.sqlite

- Account authorizations with applications

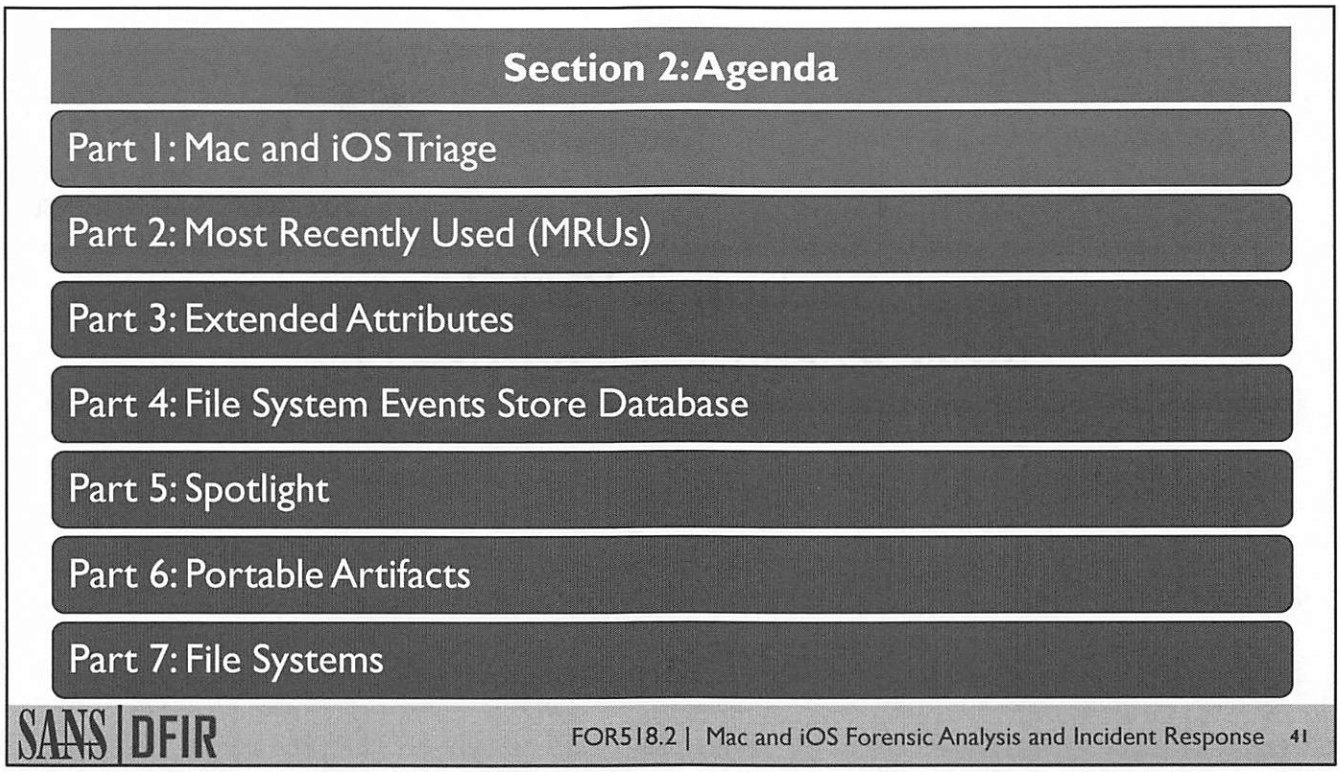
```
1  select Z_PK,zaccounttype, zbundleid,zgrantedpermissions from zauthorization
```

	Z_PK	ZACCOUNTTYPE	ZBUNDLEID	ZGRANTEDPERMISSIONS
1	1	1	com.atebits.Tweetie2	*NULL*
2	2	1	com.hootsuite.hootsuite	
3	3	3	com.facebook.Facebook	user_about_me,read_stream
4	4	1	com.zenlabs.c25k	

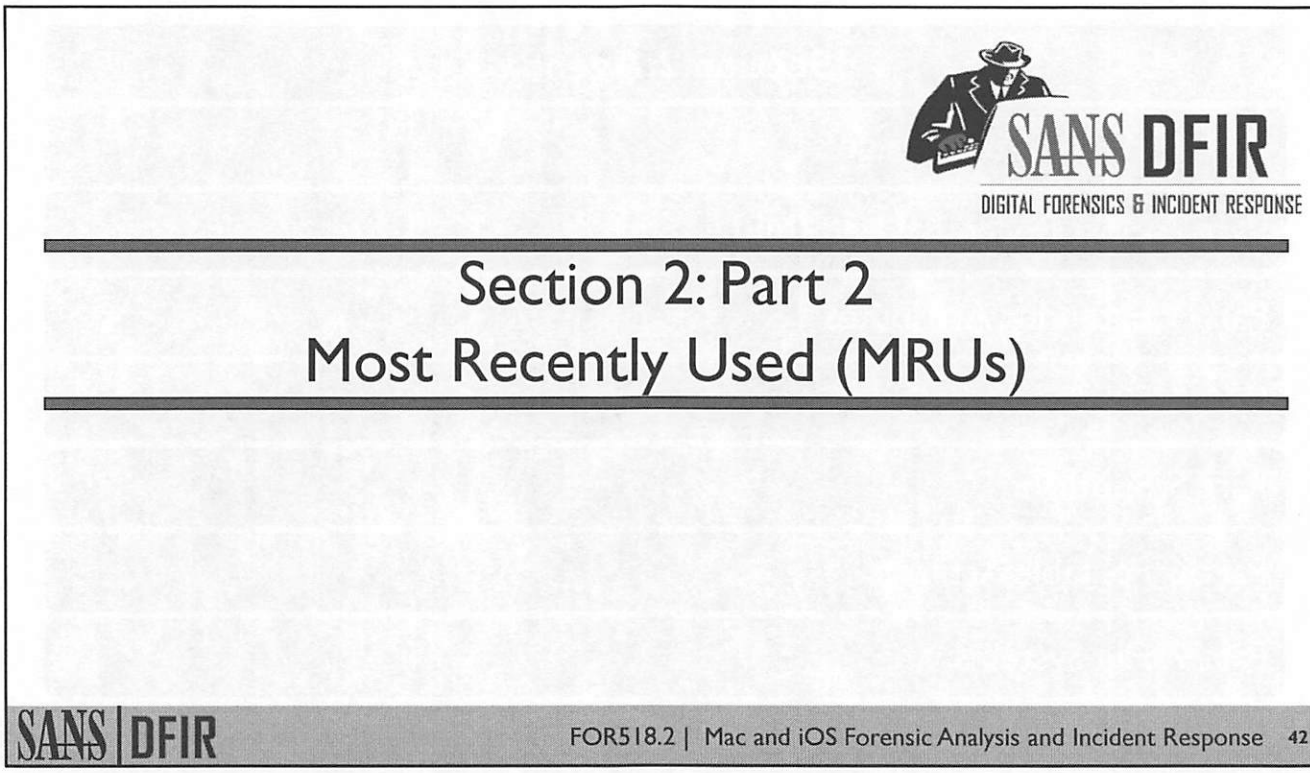
The `ZAUTHORIZATION` table contains information about accounts that have access to specific applications and their permissions.

In the example above, the Twitter account has authorization to use these three applications:

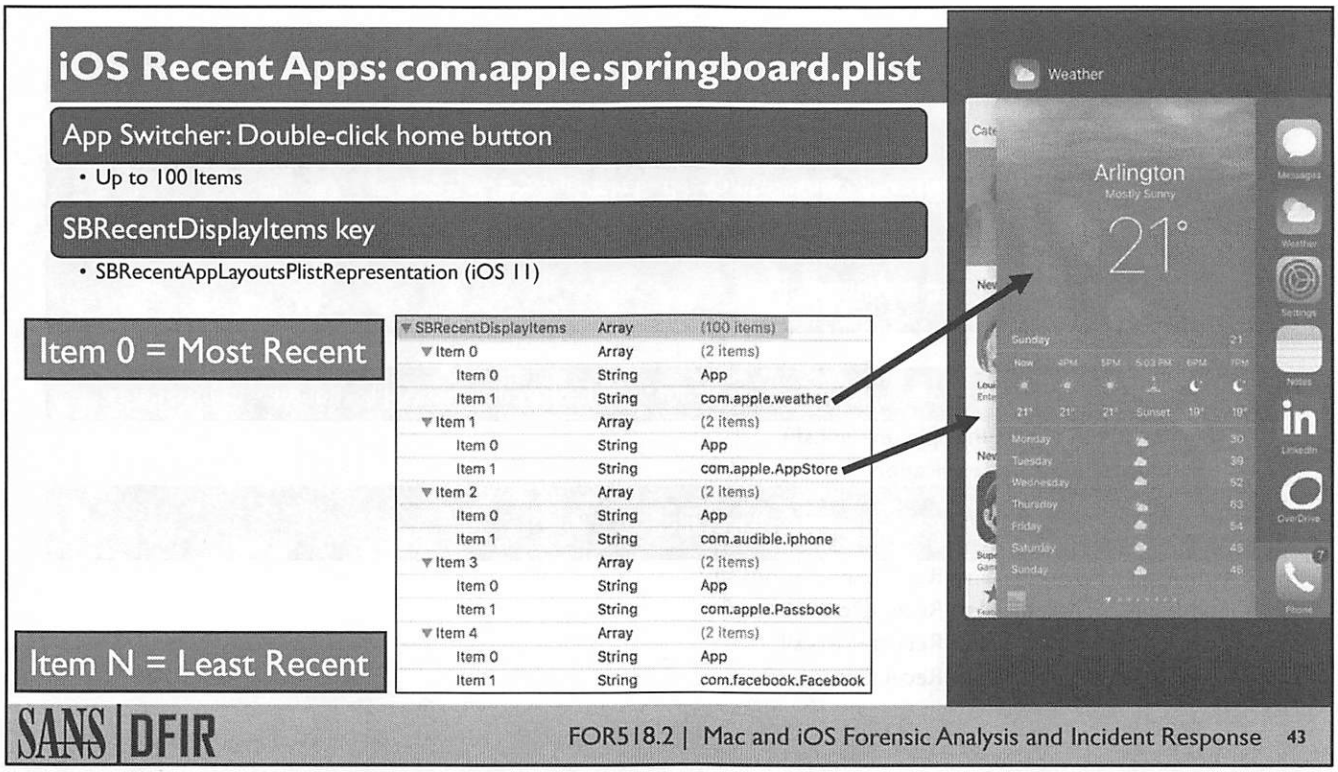
- `com.atebits.Tweetie2`: Twitter App
- `com.hootsuite.hootsuite`: Hootsuite App
- `com.zenlabs.c25k`: Couch to 5k Application



This page intentionally left blank.



This page intentionally left blank.



Location: [/private/var]/mobile/Library/Preferences/com.apple.springboard.plist

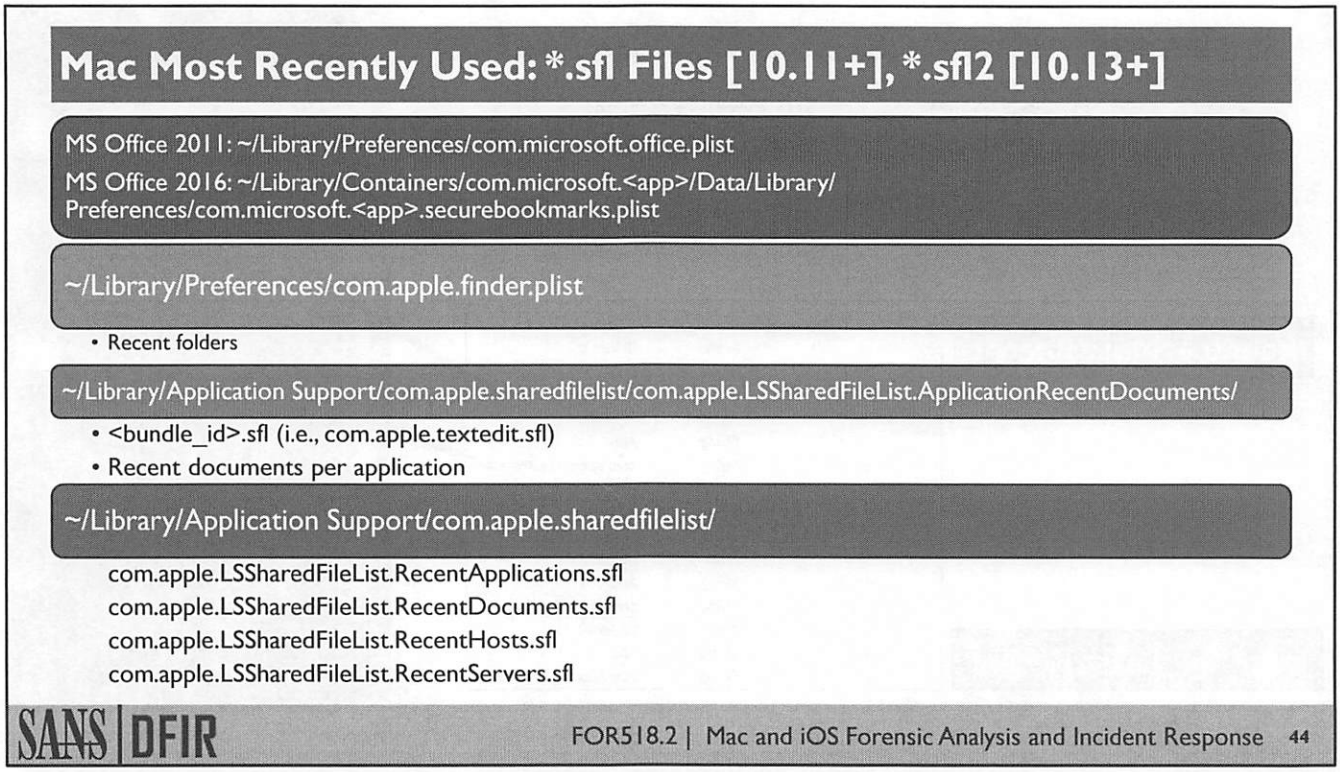
Recently used iOS applications can help an investigation by allowing us to determine which applications have been used and the most recent order of usage.

In the screenshot to the right, you see what the user might see when they are flipping through their apps. You can get to this view by double-clicking the home button on iDevices. The most recent application is to the left. In this example, the user was on the Springboard but has most recently used the Weather application and the App Store before that (although it is hard to see in this view.)

The "SBRecentDisplayItems" key in the springboard plist stores up to the last 100 applications that have been used. Item 0 is the most recent application, while Item N is the least recently used application. Unfortunately, this plist does not keep track of usage timestamps. In the plist example above, it shows that this particular device has 100 recent applications; however, if the user "swiped up" and removed one, it will also be removed from this plist. On iOS 11, the key used is "ValueSBRecentAppLayoutsPlistRepresentation".

The example has 5 (out of 100!) applications shown, and each one is identified by its bundle ID, from most recent to least recent:

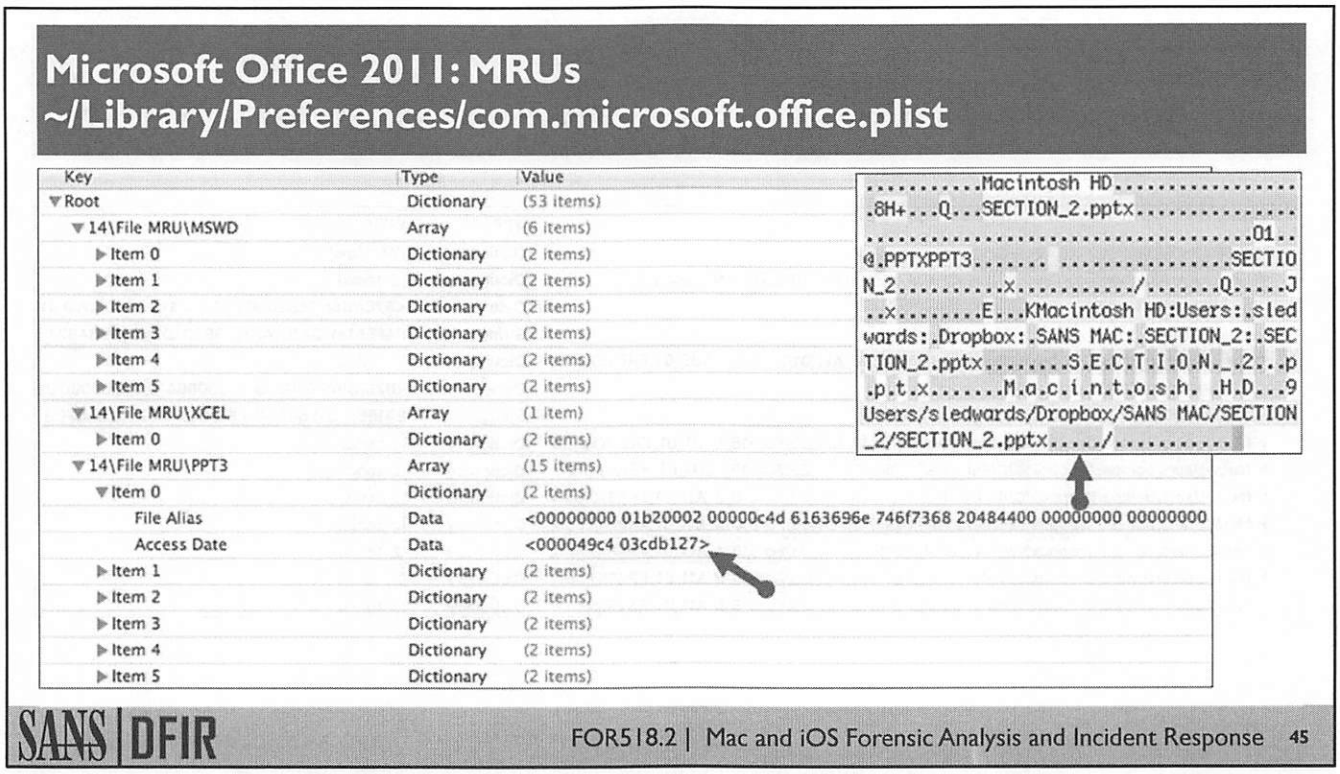
- com.apple.weather: Apple Weather
- com.apple.AppStore: Apple App Store
- com.audible.iphone: Audible
- com.apple.Passbook: Apple Passes/Wallet
- com.facebook.Facebook: Facebook



10.11 changed some of the MRUs slightly. The recent documents for each application, and other recent items, now use the "SFL" format. This format uses the NSKeyedArchiver format. These are binary plist files that have the file extension "`.sfl`".

In 10.13, this format changed slightly with the *.sfl2 file extension.

The items found in com.apple.finder.plist are still in the same format found on 10.10 and older.



The always ubiquitous Microsoft Office uses the `com.microsoft.office.plist` property list to store its MRU files. Each application in Office (Word, Excel, PowerPoint, etc.) has a separate MRU key containing the most recently used files for that application.

Each MRU has Alias data that can be extracted to view additional data (shown above) and an access date in Mac OS format.

The access date can be interpreted as follows:

- Select the middle four bytes (i.e., `0x49C403CD`)
- Put these bytes into the Calculator.app
- Select "Byte Flip" to change it to little endian (`0xCD03C449`)
- Calculate the numeric Mac OS date (i.e., `3439576137`)
- Use Epoch Converter to make the date human readable (`2012-12-28 21:48:57 Fri UTC`)

Microsoft Office 2016: MRUs
~/Library/Containers/com.microsoft.<app>/Data/Library/Preferences/com.microsoft.<app>.securebookmarks.plist

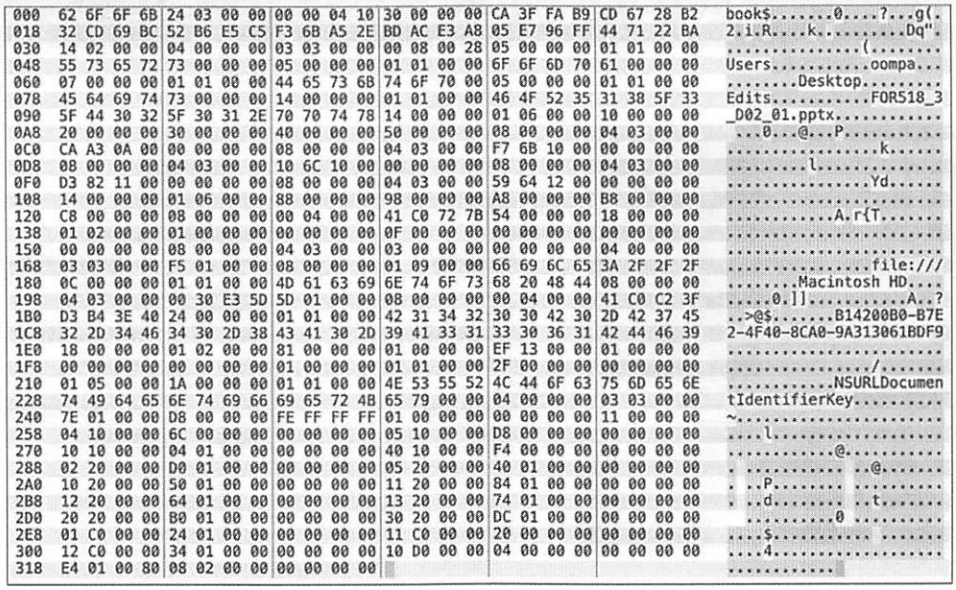
Key	Type	Value
▼ Root	Dictionary	(27 items)
▼ file:///Users/oompa/Dropbox%20(Personal)/Conferences/CEIC2014/MacLogs.pptx	Dictionary	(2 items)
kBookmarkDataKey	Data	<626f6f6b 18030000 00000410 30000000 41
kUUIDKey	String	8AFEF131-877C-4673-8B2D-DC2D1926A877
▼ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_01/For518_1_A05_01_PRESIO.pptx	Dictionary	(2 items)
kBookmarkDataKey	Data	<626f6f6b 04030000 00000410 30000000 09
kUUIDKey	String	39166053-F648-4BE5-8D67-0D22C13ACFD9
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_1_A11_01_PRESIO.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_1_A11_01_RSTR.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_2_A11_01_PRESIO.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_3_A11_01_PRESIO.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_4_A11_01_PRESIO.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_5_A11_01_PRESIO.pptx	Dictionary	(2 items)
▶ file:///Users/oompa/Dropbox%20(Personal)/FOR518_A11_03/For518_5_A11_01_RSTR.pptx	Dictionary	(2 items)

SANS DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 46

Microsoft 2016 changed quite a bit; for example, the MRUs are now being stored in a different property list file. This list is populated in an alphabetical format, so we do not know which document was opened first or last. We also do not get timestamps as we did before. Each key only has bookmark data and a UUID associated with it.

We do, however, get more than the macOS MRU default count of 10!

Bookmark Data



```
000  62 6F 6F 6B 24 03 00 00 00 00 04 10 30 00 00 00 CA 3F FA B9 CD 67 28 B2   book$.......0....?...g(.
018  32 CD 69 BC 52 B6 E5 C5 F3 6B A5 2E BD AC E3 A8 05 E7 96 FF 44 71 22 BA   2.i.R....k.........Dq".
030  14 02 00 00 04 00 00 00 03 03 00 00 00 08 00 28 05 00 00 00 01 01 00 00   ...............(........
048  55 73 65 72 73 00 00 00 05 00 00 00 01 01 00 00 6F 6F 6D 70 61 00 00 00   Users...........oompa...
060  07 00 00 00 01 01 00 00 44 65 73 6B 74 6F 70 00 05 00 00 00 01 01 00 00   ........Desktop.........
078  45 64 69 74 73 00 00 00 14 00 00 00 01 01 00 00 46 4F 52 35 31 38 5F 33   Edits...........FOR518_3
090  5F 44 30 32 5F 30 31 2E 70 70 74 78 14 00 00 00 01 06 00 00 10 00 00 00   _D02_01.pptx............
0A8  20 00 00 00 30 00 00 00 40 00 00 00 50 00 00 00 08 00 00 00 04 03 00 00    ...0...@...P...........
0C0  CA A3 0A 00 00 00 00 00 08 00 00 00 04 03 00 00 F7 6B 10 00 00 00 00 00   ..................k.....
0D8  08 00 00 00 04 03 00 00 10 6C 10 00 00 00 00 00 08 00 00 00 04 03 00 00   .........l..............
0F0  D3 82 11 00 00 00 00 00 08 00 00 00 04 03 00 00 59 64 12 00 00 00 00 00   ................Yd......
108  14 00 00 00 01 06 00 00 88 00 00 00 98 00 00 00 A8 00 00 00 B8 00 00 00   ........................
120  C8 00 00 00 08 00 00 00 00 04 00 00 41 C0 72 7B 54 00 00 00 18 00 00 00   ............A.r{T.......
138  01 02 00 00 01 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 00 00 00 00   ........................
150  00 00 00 00 08 00 00 00 04 03 00 00 03 00 00 00 00 00 00 00 04 00 00 00   ........................
168  03 03 00 00 F5 01 00 00 08 00 00 00 01 09 00 00 66 69 6C 65 3A 2F 2F 2F   ................file:///
180  0C 00 00 00 01 01 00 00 4D 61 63 69 6E 74 6F 73 68 20 48 44 08 00 00 00   ........Macintosh HD....
198  04 03 00 00 00 30 E3 5D 5D 01 00 00 08 00 00 00 00 04 00 00 41 C0 C2 3F   .....0.]]...........A..?
1B0  D3 B4 3E 40 24 00 00 00 01 01 00 00 42 31 34 32 30 30 42 30 2D 42 37 45   ..>@$.......B14200B0-B7E
1C8  32 2D 34 46 34 30 2D 38 43 41 30 2D 39 41 33 31 33 30 36 31 42 44 46 39   2-4F40-8CA0-9A313061BDF9
1E0  18 00 00 00 01 02 00 00 81 00 00 00 01 00 00 00 EF 13 00 00 01 00 00 00   ........................
1F8  00 00 00 00 00 00 00 00 01 00 00 00 01 01 00 00 2F 00 00 00 00 00 00 00   ................/.......
210  01 05 00 00 1A 00 00 00 01 01 00 00 4E 53 55 52 4C 44 6F 63 75 6D 65 6E   ............NSURLDocumen
228  74 49 64 65 6E 74 69 66 69 65 72 4B 65 79 00 00 04 00 00 00 03 03 00 00   tIdentifierKey..........
240  7E 01 00 00 D8 00 00 00 FE FF FF FF 01 00 00 00 00 00 00 00 11 00 00 00   ~.......................
258  04 10 00 00 6C 00 00 00 00 00 00 00 05 10 00 00 D8 00 00 00 00 00 00 00   ....l...................
270  10 10 00 00 04 01 00 00 00 00 00 00 40 10 00 00 F4 00 00 00 00 00 00 00   ............@...........
288  02 20 00 00 D0 01 00 00 00 00 00 00 05 20 00 00 40 01 00 00 00 00 00 00   . ........... ..@.......
2A0  10 20 00 00 50 01 00 00 00 00 00 00 11 20 00 00 84 01 00 00 00 00 00 00   . ..P........ ..........
2B8  12 20 00 00 64 01 00 00 00 00 00 00 13 20 00 00 74 01 00 00 00 00 00 00   . ..d........ ..t.......
2D0  20 20 00 00 B0 01 00 00 00 00 00 00 30 20 00 00 DC 01 00 00 00 00 00 00    ...........0 ..........
2E8  01 C0 00 00 24 01 00 00 00 00 00 00 11 C0 00 00 20 00 00 00 00 00 00 00   ....$........... .......
300  12 C0 00 00 34 01 00 00 00 00 00 00 10 D0 00 00 04 00 00 00 00 00 00 00   ....4...................
318  E4 01 00 80 08 02 00 00 00 00 00 00                                       ............
```


The `bookmark` data can be extracted and viewed in a hex editor. A `bookmark` is a reference mechanism that is used to find a file or directory, similar to Windows Link files.

Each bookmark starts off with the header "`book`".

In the screenshot above, various items of interest can be viewed:

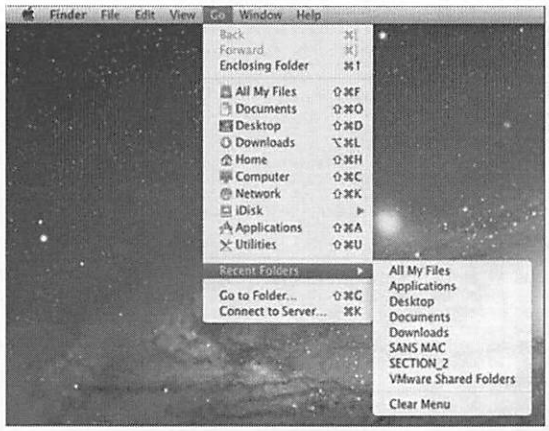
- Strings making up the file path (`/Users/oompa/Desktop/Edits/FOR518_3_D02_01.pptx`)
- Volume Name (`Macintosh HD`)
- Volume GUID (`B14200B0-B7E2-4F40-8CA0-9A313061BDF9`)

Bookmark data may also contain PNG icons that can be extracted.

Reference:

CFURL:
`https://developer.apple.com/documentation/corefoundation/cfurl-rd7`

Recent Folders: FXRecentFolders ~/Library/Preferences/com.apple.finder.plist



FXRecentFolders	Array	(8 items)
Item 0	Dictionary	(2 items)
file-bookmark	Data	<626f6f6b a0020000 00000110 10000000
name	String	Documents
Item 1	Dictionary	(2 items)
file-bookmark	Data	<626f6f6b a0020000 00000110 10000000
name	String	Downloads
Item 2	Dictionary	(2 items)
file-bookmark	Data	<626f6f6b 9c020000 00000110 10000000
name	String	Desktop

The Recent Folders are located in the `com.apple.finder.plist` in the `Preferences` directory.

Shown above are two examples of the `com.apple.finder.plist` file. Each recent Item contains a folder name and a reference link.

On top is an example from a 10.7+ system. The `file-bookmark` is a file reference using the same bookmark format as before.

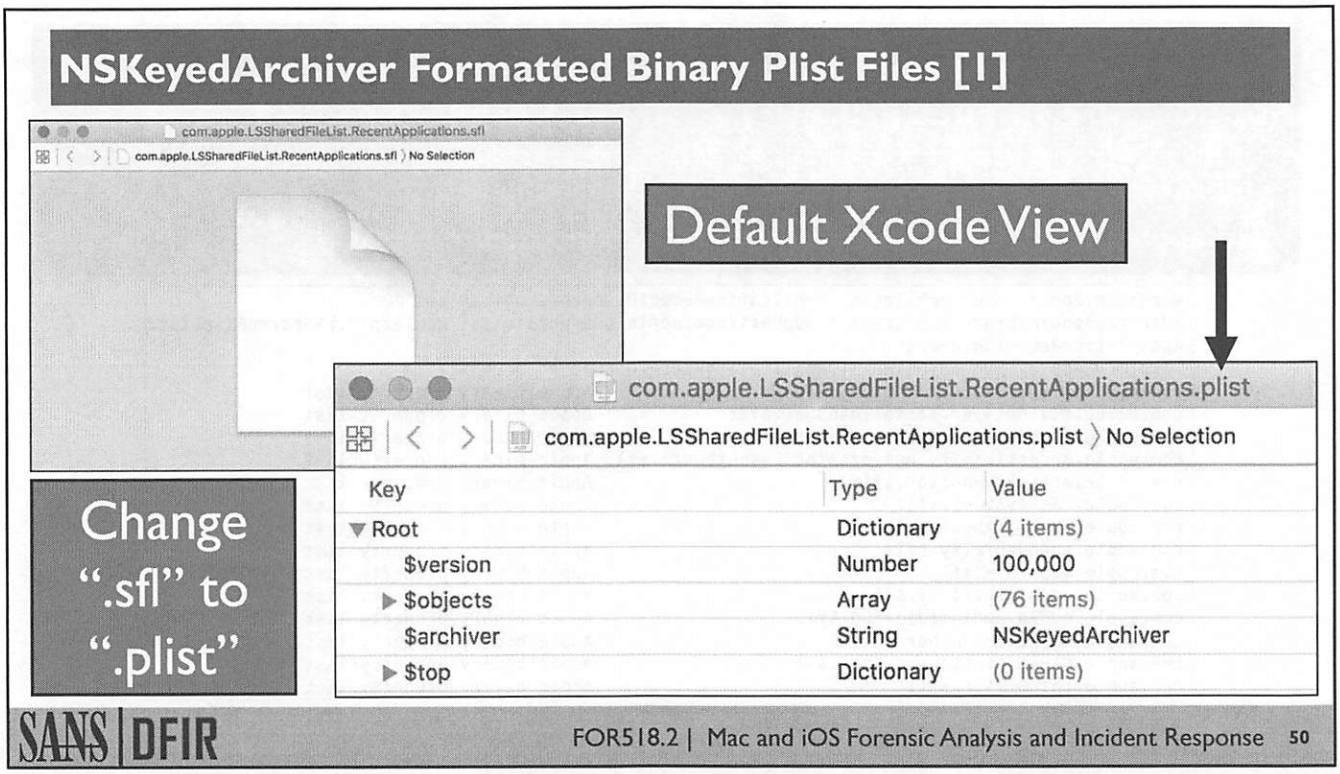
Recent Documents per Application <bundle_id>.sfl Files [10.11+] , *.sfl2 [10.13+]

~/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.ApplicationRecentDocuments/

```
[word:com.apple.LSSharedFileList.ApplicationRecentDocuments oompa$ pwd
/Users/oompa/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.
[ApplicationRecentDocuments
word:com.apple.LSSharedFileList.ApplicationRecentDocuments oompa$ file *
:no-bundle:????.sfl:                                Apple binary property list
com.adobe.flashplayer.installmanager.sfl:           Apple binary property list
com.adobe.installadobeflashplayer.sfl:              Apple binary property list
com.apple.accessibility.universalaccessauthwarn.sfl: Apple binary property list
com.apple.activitymonitor.sfl:                      Apple binary property list
com.apple.addressbook.sfl:                          Apple binary property list
com.apple.aosheartbeat.sfl:                         Apple binary property list
com.apple.aospushrelay.sfl:                         Apple binary property list
com.apple.appstore.sfl:                             Apple binary property list
com.apple.archiveutility.sfl:                       Apple binary property list
com.apple.audio.audiomidisetup.sfl:                 Apple binary property list
com.apple.backup.launcher.sfl:                      Apple binary property list
com.apple.bluetoothfileexchange.sfl:                Apple binary property list
com.apple.calculator.sfl:                           Apple binary property list
```


Each application will have its own "SFL" file. Each of these is named by the "reverse DNS" format application bundle ID.

This list will contain both native Apple applications and third-party applications.



While there are many plists that use the NSKeyedArchiver format, the Shared File List "SFL" files make a good example of how to parse them.

If we take one of these files and open its directory in Xcode, you will notice it does not show you the plist structure—instead, it shows a blank page icon. This is due to the "`*.sfl`" file extension.

If we change the file extension to "`*.plist`", we can view the contents of the plist; however, due to how this plist is structured, some of the entries are not shown. Depending on the plist viewer, these values may or may not be shown. Newer versions of BlackLight do show them!

When viewing many plists at a time, you can easily tell you have one of these NSKeyedArchiver formatted plists by the first few keys:

- `$version`
- `$objects`
- `$archiver` with the value "`NSKeyedArchiver`"
- `$top`

NSKeyedArchiver Formatted Binary Plist Files [2]

No Context for Keys and Values

Item 16	String	file:///Applications/Safari.app/
▼ Item 17	Dictionary	(2 items)
$classname	String	NSURL
▼ $classes	Array	(2 items)
Item 0	String	NSURL
Item 1	String	NSObject
Item 18	Data	<626f6f6b cc020000 00000410 30000000 0
▼ Item 19	Dictionary	(2 items)
▼ NS.keys	Array	(0 items)
▼ NS.objects	Array	(0 items)
▼ Item 20	Dictionary	(2 items)
$classname	String	SFLListItem
▼ $classes	Array	(2 items)
Item 0	String	SFLListItem
Item 1	String	NSObject
▼ Item 21	Dictionary	(1 item)
order	Number	-319
▼ Item 22	Dictionary	(1 item)
NS.uuidbytes	Data	<cffa6b33 6b46476e b93a3384 d4c48242>
Item 23	String	Terminal
▼ Item 24	Dictionary	(0 items)
Item 25	String	file:///Applications/Utilities/Terminal.app/
Item 26	Data	<626f6f6b 04030000 00000410 30000000 0
▼ Item 27	Dictionary	(1 item)
order	Number	-320
▼ Item 28	Dictionary	(1 item)
NS.uuidbytes	Data	<6229995c 87f9435a b6012fdd a7cf1852>
Item 29	String	System Preferences

If we take a look at one these files, we can see items that appear to be of interest. The problem with this format of plist files is that there is no outright context put into the keys and values.

Item 16	String	file:///Applications/Safari.app/
▼ Item 17	Dictionary	(2 items)
$classname	String	NSURL
▼ $classes	Array	(2 items)
Item 0	String	NSURL
Item 1	String	NSObject
Item 18	Data	<626f6f6b cc020000 00000410 30000000 0(
▼ Item 19	Dictionary	(2 items)
▼ NS.keys	Array	(0 items)
▼ NS.objects	Array	(0 items)
▼ Item 20	Dictionary	(2 items)
$classname	String	SFLListItem
▼ $classes	Array	(2 items)
Item 0	String	SFLListItem
Item 1	String	NSObject
▼ Item 21	Dictionary	(1 item)
order	Number	-319
▼ Item 22	Dictionary	(1 item)
NS.uuidbytes	Data	<cffa6b33 6b46476e b93a3384 d4c48242>
Item 23	String	Terminal
▼ Item 24	Dictionary	(0 items)
Item 25	String	file:///Applications/Utilities/Terminal.app/
Item 26	Data	<626f6f6b 04030000 00000410 30000000 0
▼ Item 27	Dictionary	(1 item)
order	Number	-320
▼ Item 28	Dictionary	(1 item)
NS.uuidbytes	Data	<6229995c 87f9435a b6012fdd a7cf1852>
Item 29	String	System Preferences

NSKeyedArchiver Formatted Binary Plist Files [3]

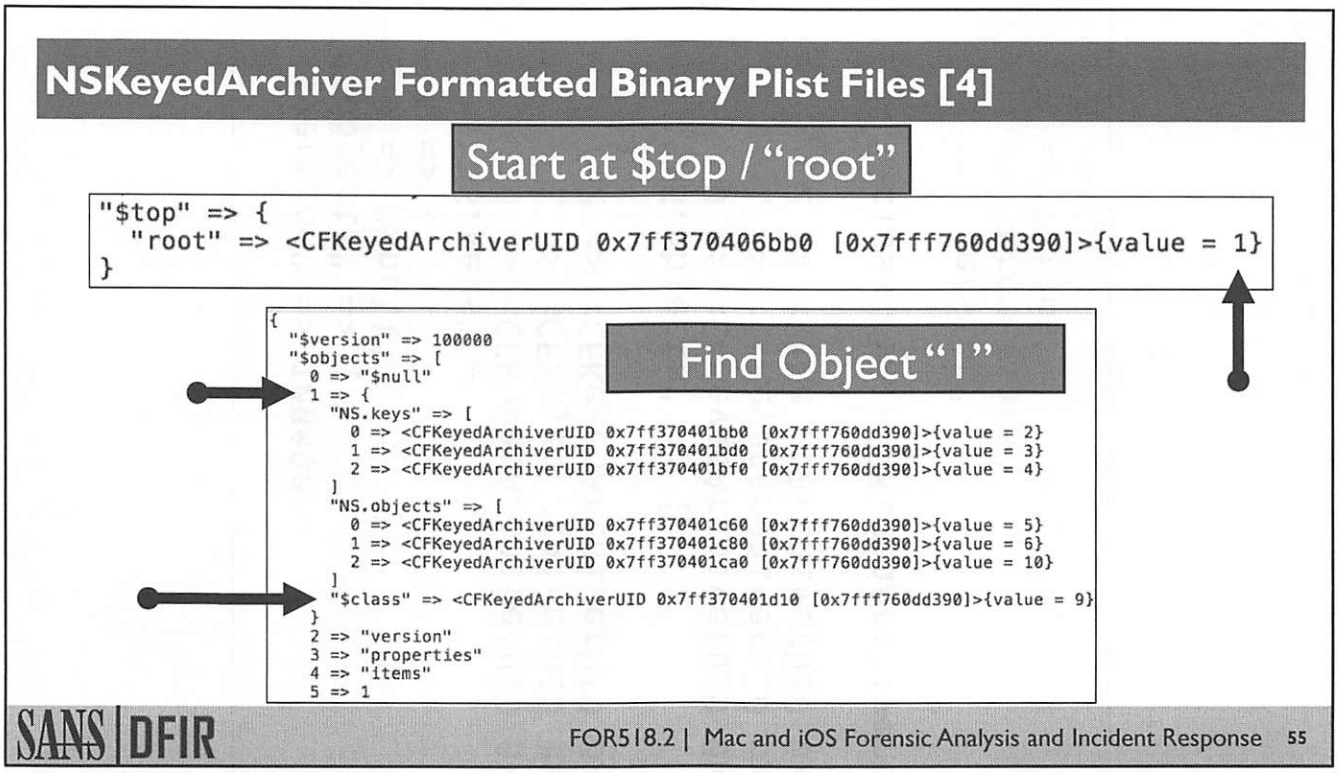
```
{
  "$version" => 100000
  "$objects" => [
    0 => "$null"
    1 => {
      "NS.keys" => [
        0 => <CFKeyedArchiverUID 0x7ff370401bb0 [0x7fff760dd390]>{value = 2}
        1 => <CFKeyedArchiverUID 0x7ff370401bd0 [0x7fff760dd390]>{value = 3}
        2 => <CFKeyedArchiverUID 0x7ff370401bf0 [0x7fff760dd390]>{value = 4}
      ]
      "NS.objects" => [
        0 => <CFKeyedArchiverUID 0x7ff370401c60 [0x7fff760dd390]>{value = 5}
        1 => <CFKeyedArchiverUID 0x7ff370401c80 [0x7fff760dd390]>{value = 6}
        2 => <CFKeyedArchiverUID 0x7ff370401ca0 [0x7fff760dd390]>{value = 10}
      ]
      "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
    }
    2 => "version"
    3 => "properties"
    4 => "items"
    5 => 1
    6 => {
      "NS.keys" => [
        0 => <CFKeyedArchiverUID 0x7ff370401db0 [0x7fff760dd390]>{value = 7}
      ]
      "NS.objects" => [
        0 => <CFKeyedArchiverUID 0x7ff370401e10 [0x7fff760dd390]>{value = 8}
      ]
      "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
    }
    7 => "com.apple.LSSharedFileList.MaxAmount"
    8 => 10
    9 => {
      "$classname" => "NSDictionary"
      "$classes" => [
        0 => "NSDictionary"
        1 => "NSObject"
      ]
    }
```

Output from plutil -p

If your plist viewer shows all the values needed, feel free to use it! If it doesn't, we can use the output from the "`plutil -p <file>`" command.

This output shows the values in a somewhat human readable format in JSON style output.

```
{
  "$version" => 100000
  "$objects" => [
    0 => "$null"
    1 => {
      "NS.keys" => [
        0 => <CFKeyedArchiverUID 0x7ff370401bb0 [0x7fff760dd390]>{value = 2}
        1 => <CFKeyedArchiverUID 0x7ff370401bd0 [0x7fff760dd390]>{value = 3}
        2 => <CFKeyedArchiverUID 0x7ff370401bf0 [0x7fff760dd390]>{value = 4}
      ]
      "NS.objects" => [
        0 => <CFKeyedArchiverUID 0x7ff370401c60 [0x7fff760dd390]>{value = 5}
        1 => <CFKeyedArchiverUID 0x7ff370401c80 [0x7fff760dd390]>{value = 6}
        2 => <CFKeyedArchiverUID 0x7ff370401ca0 [0x7fff760dd390]>{value = 10}
      ]
      "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
    }
    2 => "version"
    3 => "properties"
    4 => "items"
    5 => 1
    6 => {
      "NS.keys" => [
        0 => <CFKeyedArchiverUID 0x7ff370401db0 [0x7fff760dd390]>{value = 7}
      ]
      "NS.objects" => [
        0 => <CFKeyedArchiverUID 0x7ff370401e10 [0x7fff760dd390]>{value = 8}
      ]
      "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
    }
    7 => "com.apple.LSSharedFileList.MaxAmount"
    8 => 10
    9 => {
      "$classname" => "NSDictionary"
      "$classes" => [
        0 => "NSDictionary"
        1 => "NSObject"
      ]
    }
```

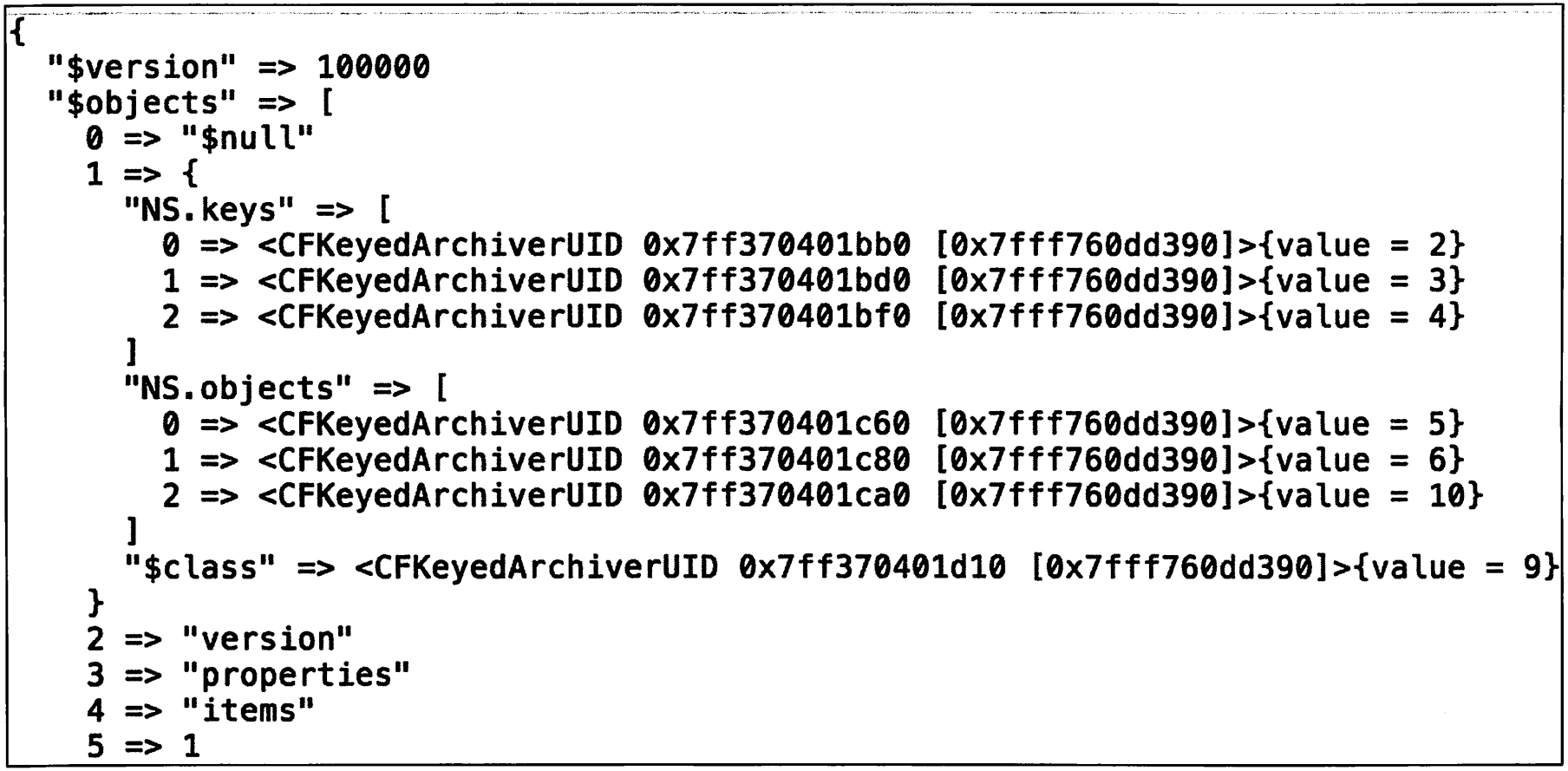


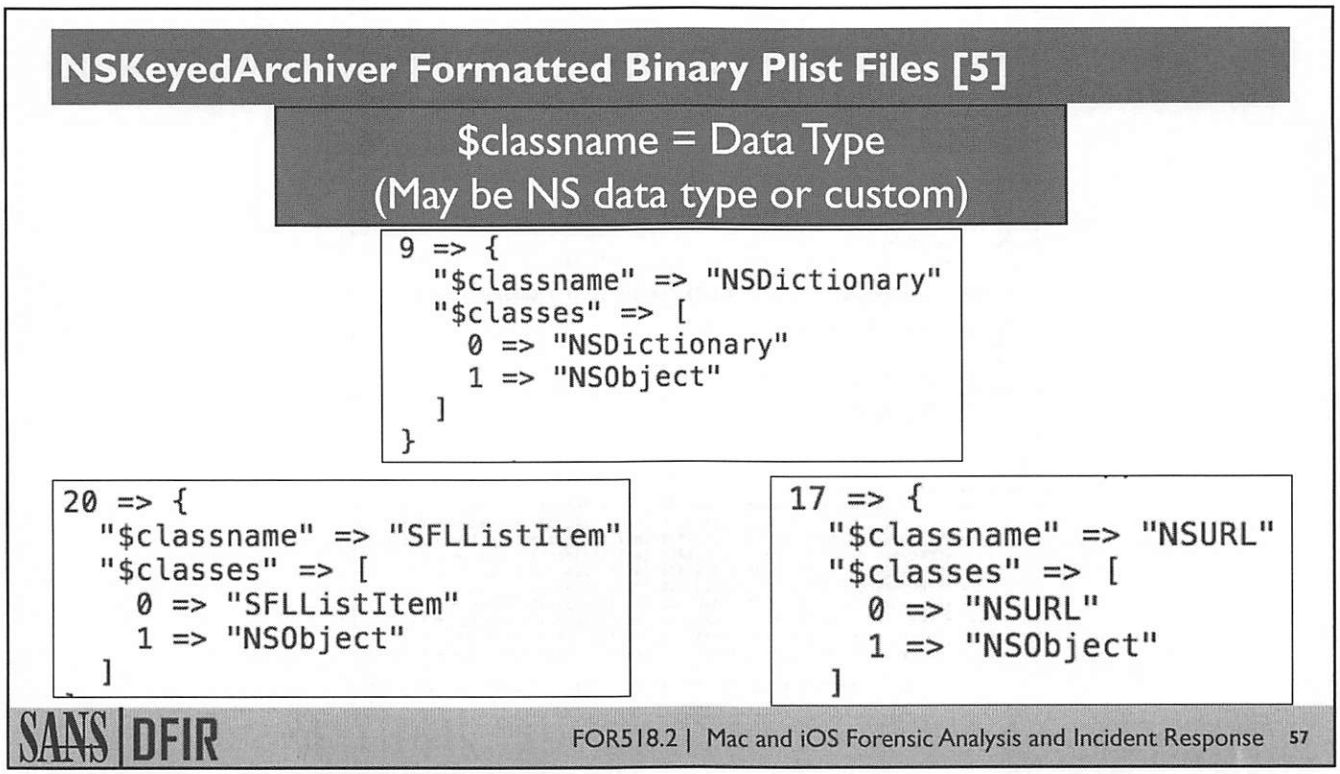
To start parsing these plist files, we need to start at the `$top` key. This should hold the "`root`" of the plist tree. In the example above, "`root`" points to the value "`1`".

In the `$objects` section, we will find the "`1`" value, as noted by the arrow above. This key holds (sub)keys and objects. This particular format has a key = value format as shown below.

```
Key = Object
2 = 5
3 = 6
4 = 10
```

You will notice that certain "`objects`" have a `$class` key. This describes a bit about the values in this object. Some examples are in the next slide.





In the previous example, the `$class` referred to object "9", or an `NSDictionary` data type.

Other examples may include other `NSData` types, like `NSURL`, `NSDate`, `NSUUID`, or `NSArray`.

Other "custom" types may also be available. In the example plist, we have a `$classname` "`SFLListItem`" describing the recent item.

NSKeyedArchiver Formatted Binary Plist Files [6]

```
2 => "version"
3 => "properties"
4 => "items"
5 => 1
6 => {
  "NS.keys" => [
    0 => <CFKeyedArchiverUID 0x7ff370401db0 [0x7fff760dd390]>{value = 7}
  ]
  "NS.objects" => [
    0 => <CFKeyedArchiverUID 0x7ff370401e10 [0x7fff760dd390]>{value = 8}
  ]
  "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
}
7 => "com.apple.LSSharedFileList.MaxAmount"
8 => 10
9 => {
  "$classname" => "NSDictionary"
  "$classes" => [
    0 => "NSDictionary"
    1 => "NSObject"
  ]
}
10 => {
  "NS.objects" => [
    0 => <CFKeyedArchiverUID 0x7ff370401ff0 [0x7fff760dd390]>{value = 11}
    1 => <CFKeyedArchiverUID 0x7ff370402010 [0x7fff760dd390]>{value = 21}
    2 => <CFKeyedArchiverUID 0x7ff370402030 [0x7fff760dd390]>{value = 27}
    3 => <CFKeyedArchiverUID 0x7ff370402050 [0x7fff760dd390]>{value = 33}
    4 => <CFKeyedArchiverUID 0x7ff370402070 [0x7fff760dd390]>{value = 39}
    5 => <CFKeyedArchiverUID 0x7ff370402090 [0x7fff760dd390]>{value = 45}
    6 => <CFKeyedArchiverUID 0x7ff3704020b0 [0x7fff760dd390]>{value = 51}
    7 => <CFKeyedArchiverUID 0x7ff3704020d0 [0x7fff760dd390]>{value = 57}
    8 => <CFKeyedArchiverUID 0x7ff3704020f0 [0x7fff760dd390]>{value = 63}
    9 => <CFKeyedArchiverUID 0x7ff370402110 [0x7fff760dd390]>{value = 69}
  ]
  "$class" => <CFKeyedArchiverUID 0x7ff3704021b0 [0x7fff760dd390]>{value = 75}
}
```


Continuing with our example, we can see that keys "2", "3", and "4" are associated with "version", "properties", and "items", respectively. The "items" key sounds most promising. In this case, let's follow this one down the line.

Previously, we saw that value "4" is associated with value "10". Value "10" shown above contains an array of objects: 11, 21, 27, 33, 39, 45, 51, 57, 63, 69; 10 items! This sounds like our Recent Applications.

Let's look at recent item "11" …

```
2 => "version"
3 => "properties"
4 => "items"
5 => 1
6 => {
  "NS.keys" => [
    0 => <CFKeyedArchiverUID 0x7ff370401db0 [0x7fff760dd390]>{value = 7}
  ]
  "NS.objects" => [
    0 => <CFKeyedArchiverUID 0x7ff370401e10 [0x7fff760dd390]>{value = 8}
  ]
  "$class" => <CFKeyedArchiverUID 0x7ff370401d10 [0x7fff760dd390]>{value = 9}
}
7 => "com.apple.LSSharedFileList.MaxAmount"
8 => 10
9 => {
  "$classname" => "NSDictionary"
  "$classes" => [
    0 => "NSDictionary"
    1 => "NSObject"
  ]
}
10 => {
  "NS.objects" => [
    0 => <CFKeyedArchiverUID 0x7ff370401ff0 [0x7fff760dd390]>{value = 11}
    1 => <CFKeyedArchiverUID 0x7ff370402010 [0x7fff760dd390]>{value = 21}
    2 => <CFKeyedArchiverUID 0x7ff370402030 [0x7fff760dd390]>{value = 27}
    3 => <CFKeyedArchiverUID 0x7ff370402050 [0x7fff760dd390]>{value = 33}
    4 => <CFKeyedArchiverUID 0x7ff370402070 [0x7fff760dd390]>{value = 39}
    5 => <CFKeyedArchiverUID 0x7ff370402090 [0x7fff760dd390]>{value = 45}
    6 => <CFKeyedArchiverUID 0x7ff3704020b0 [0x7fff760dd390]>{value = 51}
    7 => <CFKeyedArchiverUID 0x7ff3704020d0 [0x7fff760dd390]>{value = 57}
    8 => <CFKeyedArchiverUID 0x7ff3704020f0 [0x7fff760dd390]>{value = 63}
    9 => <CFKeyedArchiverUID 0x7ff370402110 [0x7fff760dd390]>{value = 69}
  ]
  "$class" => <CFKeyedArchiverUID 0x7ff3704021b0 [0x7fff760dd390]>{value = 75}
}
```

Item "11" contains a variety of keys:

- `$class` = `value` = 20 (not shown: "SFLListItem")
- `order` = -146
  - Research is ongoing in this area, as some plists have both positive numbers as well as negative. It does not appear consistent which items are older or newer.
- `bookmark` = `value 18`: This is the binary BLOB shown in the example. We can extract this to look at the bookmark data for this recent application.
- `uniqueIdentifier` = value 12: This is an `NS.uuidbytes`, or GUID, for this recent application.
- `properties` = value 19: (now shown, but empty)
- URL = value 15: The path to the application (`file:///Applications/System%20Preferences.app/`)
- `name` = value `14`: Application Name "`System Preferences`"

Most of this information is the same data we found in previous MRU plist files—just in a slightly different, more convoluted format.

```
    11 => {
      "$class" => <CFKeyedArchiverUID 0x7ff370402280 [0x7fff760dd390]>{value = 20}
      "order" => -146
      "bookmark" => <CFKeyedArchiverUID 0x7ff3704022a0 [0x7fff760dd390]>{value = 18}
      "uniqueIdentifier" => <CFKeyedArchiverUID 0x7ff3704022c0 [0x7fff760dd390]>{value = 12}
      "properties" => <CFKeyedArchiverUID 0x7ff3704022e0 [0x7fff760dd390]>{value = 19}
      "URL" => <CFKeyedArchiverUID 0x7ff370402300 [0x7fff760dd390]>{value = 15}
      "name" => <CFKeyedArchiverUID 0x7ff370402320 [0x7fff760dd390]>{value = 14}
    }
    12 => {
      "NS.uuidbytes" => <63c56a16 3b014cdf 9de42fab 8a616942>
      "$class" => <CFKeyedArchiverUID 0x7ff3704024d0 [0x7fff760dd390]>{value = 13}
    }
    13 => {
      "$classname" => "NSUUID"
      "$classes" => [
        0 => "NSUUID"
        1 => "NSObject"
      ]
    }
    14 => "System Preferences"
    15 => {
      "NS.base" => <CFKeyedArchiverUID 0x7ff370402650 [0x7fff760dd390]>{value = 0}
      "NS.relative" => <CFKeyedArchiverUID 0x7ff370402690 [0x7fff760dd390]>{value = 16}
      "$class" => <CFKeyedArchiverUID 0x7ff370402670 [0x7fff760dd390]>{value = 17}
    }
    16 => "file:///Applications/System%20Preferences.app/"
    17 => {
      "$classname" => "NSURL"
      "$classes" => [
        0 => "NSURL"
        1 => "NSObject"
      ]
    }
    18 => <626f6f6b e4020000 00000410 30000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
04020000 0c000000 01010000 4170706c 69636174 696f6e73 16000000 01010000 53797374 656d2050 72656665 72656e63 65732e61
70700000 08000000 01060000 04000000 18000000 08000000 04030000 47000000 00000000 08000000 04030000 2ca70000 00000000
08000000 01060000 48000000 58000000 08000000 00040000 41bb8965 13000000 18000000 01020000 02000000 00000000 0f000000
00000000 00000000 00000000 08000000 01090000 66696c65 3a2f2f2f 0a000000 01010000 456c2043 61707469 616e0000 08000000
04030000 00000032 74000000 08000000 00040000 41bbf944 dc000000 24000000 01010000 44363942 44313038 2d463934 322d3330
46372d42 4535422d 34413644 43343134 35463336 18000000 01020000 81000000 01000000 ef130000 01000000 00000000 00000000
01000000 01010000 2f000000 00000000 01050000 af000000 01020000 65376461 65303539 39643635 31353263 37366266 33643266
35373966 38643166 32633137 66343231 3b303030 30303030 303b3030 30303030 30303b30 30303030 30303030 30303030 3031613b
636f6d2e 6170706c 652e6170 702d7361 6e64626f 782e7265 61643b30 30303030 3030313b 30313030 30303034 3b303030 30303030
30303030 30613732 633b2f61 70706c69 63617469 6f6e732f 73797374 656d2070 72656665 72656e63 65732e61 70700000 a8000000
feffffff 01000000 00000000 0d000000 04100000 38000000 00000000 05100000 68000000 00000000 10100000 88000000 00000000
40100000 78000000 00000000 02200000 38010000 00000000 05200000 a8000000 00000000 10200000 b8000000 00000000 11200000
ec000000 00000000 12200000 cc000000 00000000 13200000 dc000000 00000000 20200000 18010000 00000000 30200000 44010000
00000000 81f00000 4c010000 00000000>
```

MacMRU Python Script: github.com/mac4n6/macMRU-Parser

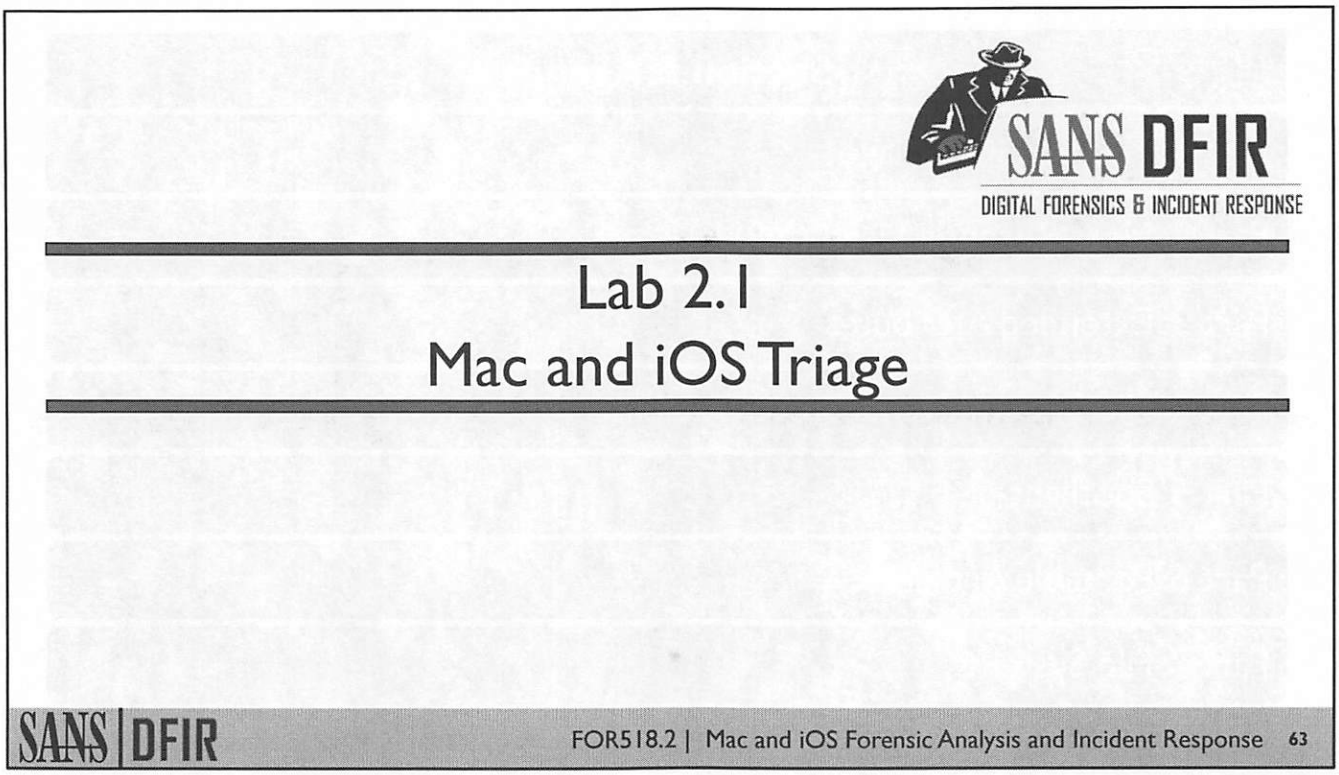
- Older Plists, SFL* Files, Microsoft Office Files, and Spotlight Shortcuts
- Run on live system or mounted image
- Parses Bookmark and Alias BLOBs

```
bit:macMRU-Parser-master oompa$ python macMRU.py ~/Library/
###### MacMRU Parser v1.1 ######
================================================================================
Parsing: /Users/oompa/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.iCloudItems.sfl
Cannot open file: /Users/oompa/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.iCloudItems.sfl
================================================================================
================================================================================
Parsing: /Users/oompa/Library/Application Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.RecentApplications.sfl
Max number of recent items in this plist: 10
    [Item Number: 0 | Order: -134.0] Name:'Microsoft Word.app' (URL:'file:///Applications/Microsoft%20Word.app/')
    [Item Number: 1 | Order: -136.0] Name:'Microsoft Error Reporting.app' (URL:'file:///Applications/Microsoft%20PowerPoint.app/
    [Item Number: 2 | Order: -140.0] Name:'Console.app' (URL:'file:///Applications/Utilities/Console.app/')
    [Item Number: 3 | Order: -130.0] Name:'osxpmem.app' (URL:'file:///private/var/folders/n7/vnfzc155443_qg0zp2cwz1880000gn/T/Ap
    [Item Number: 4 | Order: -137.0] Name:'Microsoft PowerPoint.app' (URL:'file:///Applications/Microsoft%20PowerPoint.app/')
    [Item Number: 5 | Order: -131.0] Name:'Microsoft Excel.app' (URL:'file:///Applications/Microsoft%20Excel.app/')
    [Item Number: 6 | Order: -138.0] Name:'Preview.app' (URL:'file:///Applications/Preview.app/')
    [Item Number: 7 | Order: -129.0] Name:'osxpmem.app' (URL:'file:///Users/oompa/Downloads/osxpmem.app/')
    [Item Number: 8 | Order: -133.0] Name:'Epoch Converter.app' (URL:'file:///Applications/Epoch%20Converter.app/')
    [Item Number: 9 | Order: -132.0] Name:'System Preferences.app' (URL:'file:///Applications/System%20Preferences.app/')
================================================================================
```

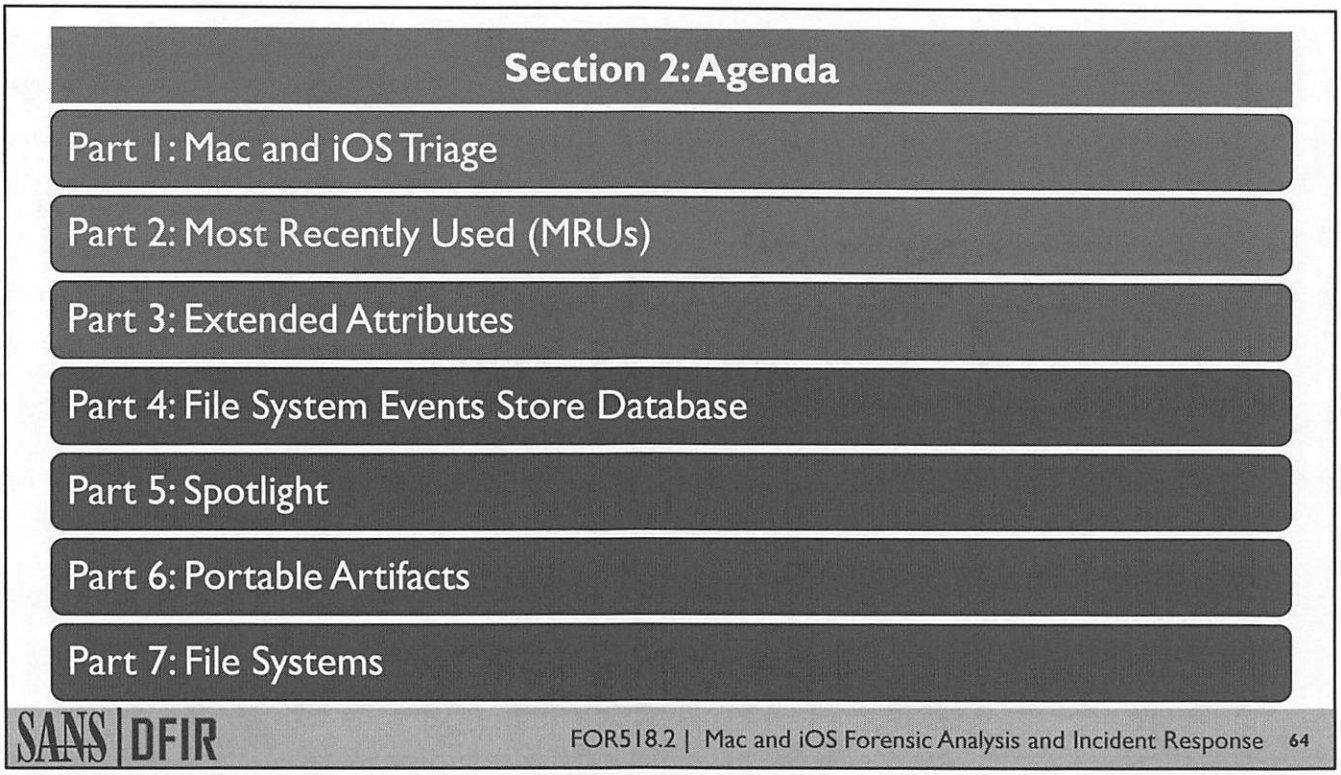

A script was created by the course author to deal with these pesky new Mac MRU plist files. It should make an analysis of these files easier on the analyst.

This Python script can be downloaded from `https://github.com/mac4n6/macMRU-Parser`

References:
`https://www.mac4n6.com/blog/2016/7/10/new-script-macmru-most-recently-used-plist-parser`
`https://www.mac4n6.com/blog/2016/8/15/update-to-macmru-parser-now-with-microsoft-office-support`
`https://www.mac4n6.com/blog/2017/7/19/script-update-mac-mru-parser-spotlight-shortcuts-blob-parsing`



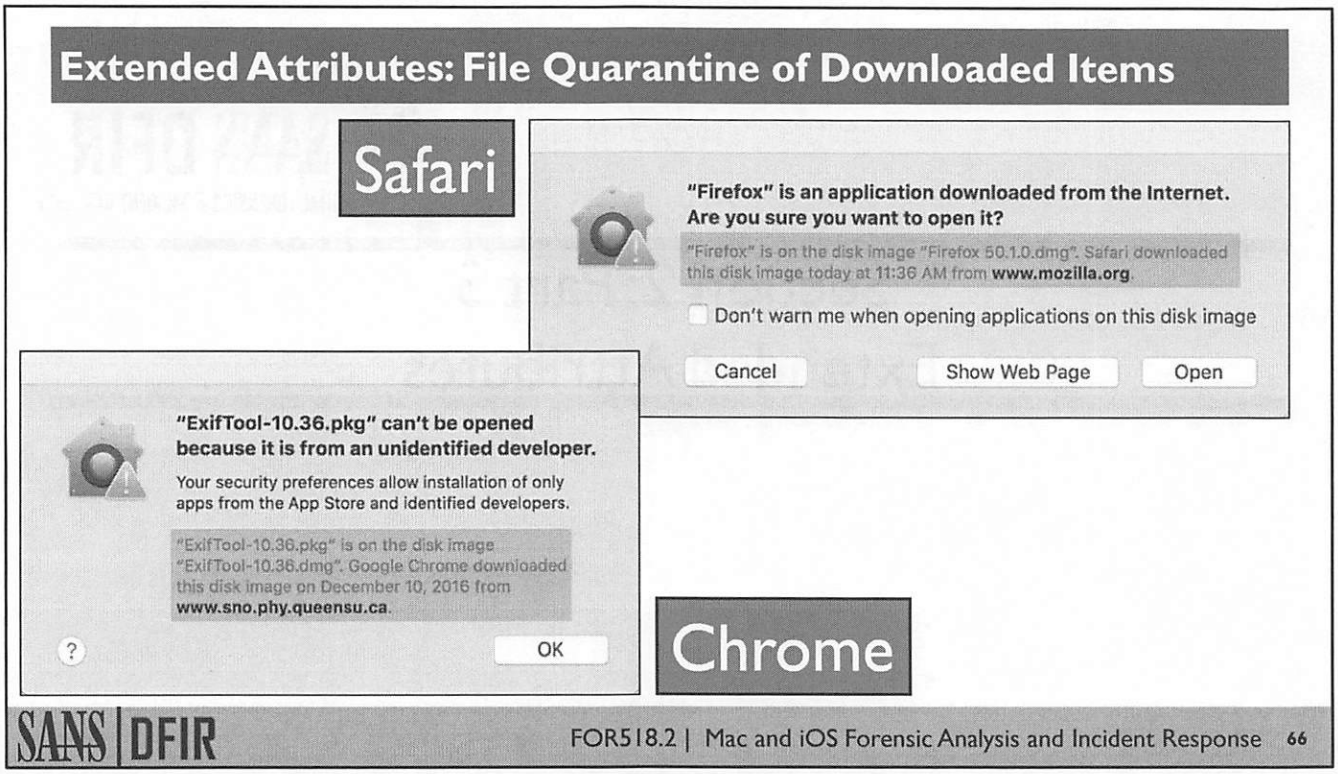
This page intentionally left blank.



This page intentionally left blank.

Section 2: Part 3
Extended Attributes

This page intentionally left blank.

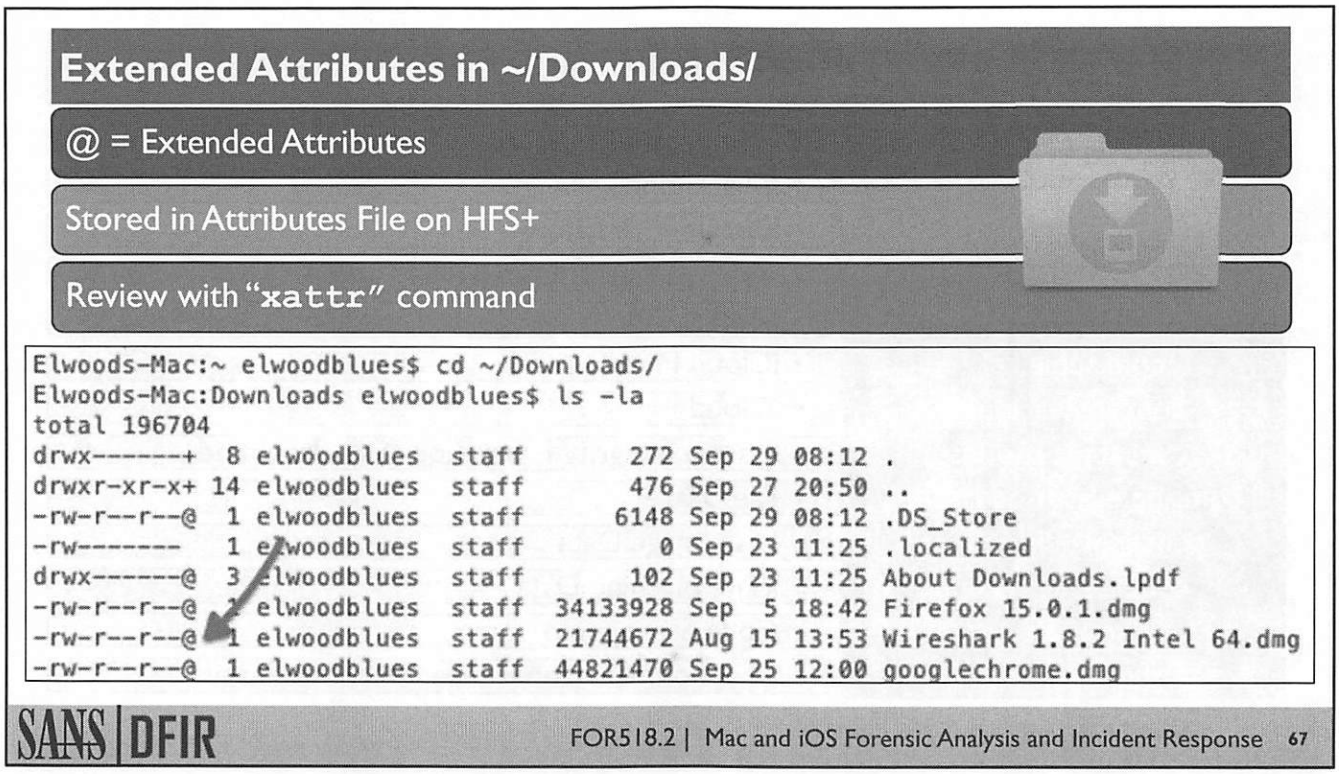


Apple uses file quarantine to protect the system by checking files against a malicious file database (more on this later). It is also used to inform the user when and where a file was downloaded from when the user attempts to open the file.

Two examples are shown above. The top example shows a file downloaded using the Safari web browser. The file downloaded was "`Firefox 50.1.0.dmg`", which was downloaded at 11:36 AM from mozilla.org.

The bottom example shows that "`ExifTool-10.36.pkg`" was downloaded using the Chrome web browser on December 10, 2016, from `www.sno.phy.queensu.ca`.

How does the Mac know when and where these files were downloaded from? … Extended attributes!

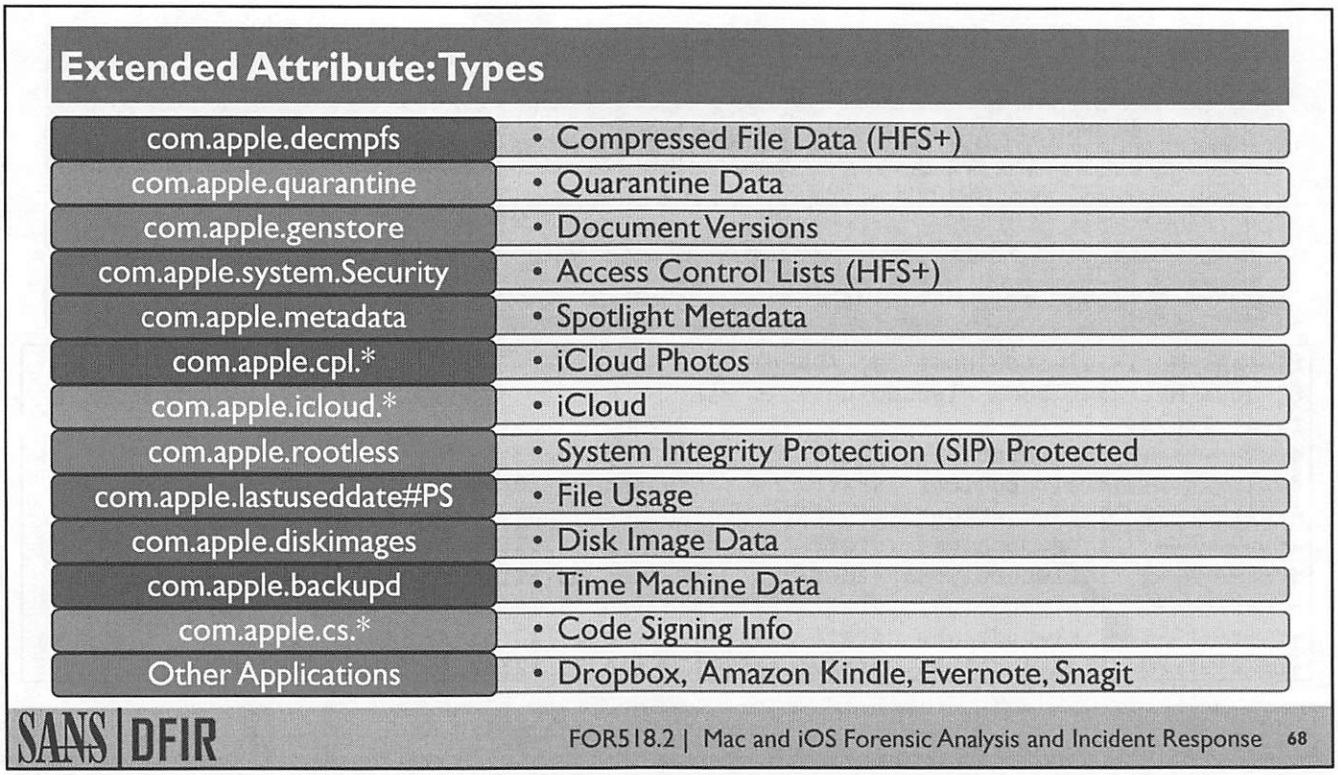


Each user has their own `Downloads` directory. This directory contains all of the downloads from applications, such as web browsers and email that use this folder as their default download location. Most applications will have a preference setting that users can change; however, by default, this directory will be used.

On a well-used system, this directory will contain months, even years' worth of downloads.

Each item that was downloaded will likely have extended attributes showing metadata of the download. To see if a file contains extended attributes, you can run the command "`ls -la`".

Files with extended attributes will have the "`@`" at the end of the permissions. You can even try running the command "`ls -l@`" to see the names of the attributes.

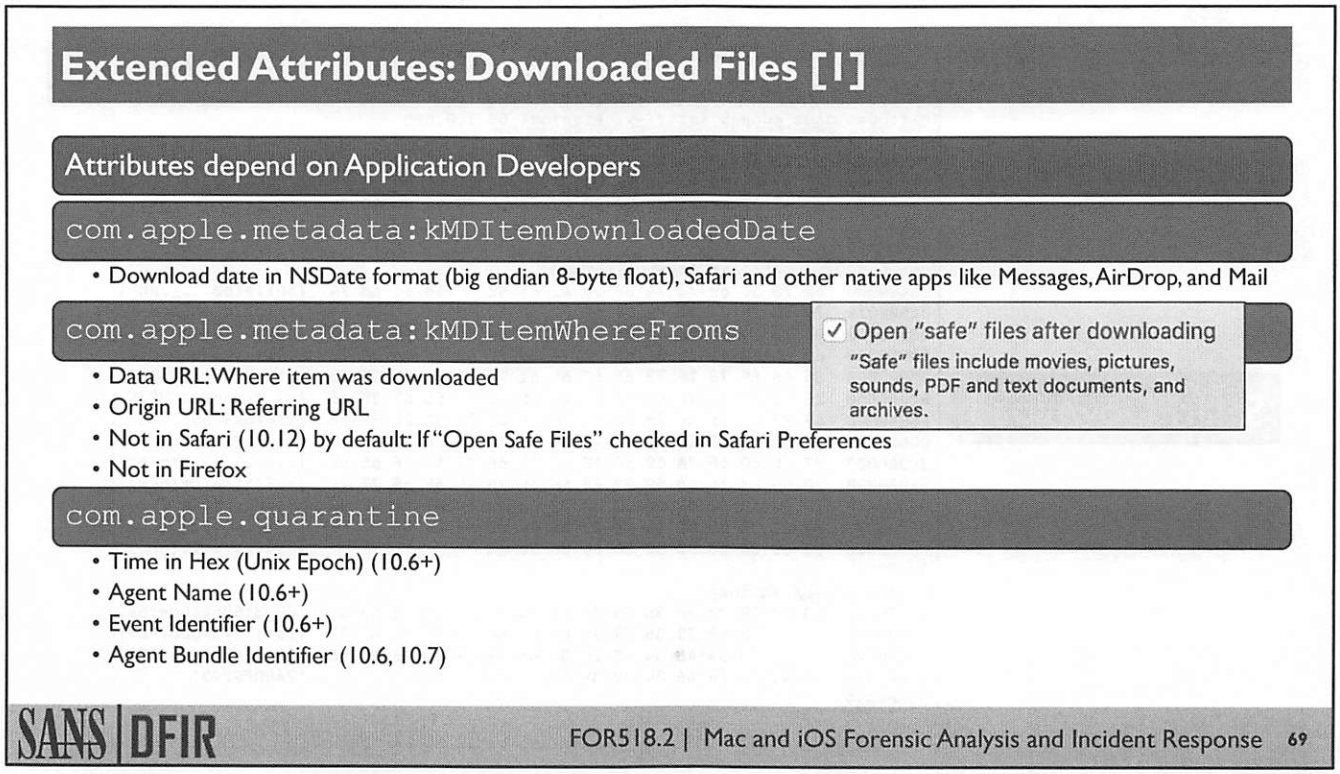


Extended attributes come in many types, some of which are listed above. The two most common are `com.apple.decmpfs` and `com.apple.quarantine`. Each attribute contains data specific to its purpose and does not follow a static format. An investigator's intuition and reverse engineering skills may be needed to decode the contents of these attributes.

The `com.apple.decmpfs` attribute is particularly interesting because it means this file is using the per-file compression available in the HFS+ file system. Files such as those found in `/bin`, like `ls`, `echo`, or `dd` are compressed.

Ever wonder why some versions of EnCase show these files as having 0 bytes? This is why.

Note: The `xattr` command filters out `com.apple.decmpfs` and `com.apple.system.Security` attributes. An investigator may need additional tools to review these two attributes.



The contents of extended attribute data can vary by operating system or application. Each browser saves different extended attributes.

Safari: All in slide
Chrome: Does not use `kMDItemDownloadedDate`

The extended attributes can contain useful information, such as:

- Download dates
- URLS: The Data URL shows where the data was actually downloaded from. The Origin URL shows where the download was referred from. This extended attribute will not get written for some of Safari's downloaded files on 10.12 if the "Open Safe Files" option is checked in Safari Preferences.
- Quarantine Data: Contains information related to Apple's file quarantine system.

Extended Attributes: Downloaded Files [2]

Safari Download

```
bit:Downloads oompa$ xattr -xl Firefox\ 50.1.0.dmg
com.apple.metadata:kMDItemDownloadedDate:
00000000  62 70 6C 69 73 74 30 30 A1 01 33 41 BE 0E E3 62  |bplist00..3A...b|
00000010  AC 63 5A 08 0A 00 00 00 00 00 00 01 01 00 00 00  |.cZ.............|
00000020  00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 13                                   |.....|
00000035
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 65 68 74  |bplist00..._.eht|
00000010  74 70 73 3A 2F 2F 64 6F 77 6E 6C 6F 61 64 2D 69  |tps://download-i|
00000020  6E 73 74 61 6C 6C 65 72 2E 63 64 6E 2E 6D 6F 7A  |nstaller.cdn.moz|
00000030  69 6C 6C 61 2E 6E 65 74 2F 70 75 62 2F 66 69 72  |illa.net/pub/fir|
00000040  65 66 6F 78 2F 72 65 6C 65 61 73 65 73 2F 35 30  |efox/releases/50|
00000050  2E 31 2E 30 2F 6D 61 63 2F 65 6E 2D 55 53 2F 46  |.1.0/mac/en-US/F|
00000060  69 72 65 66 6F 78 25 32 30 35 30 2E 31 2E 30 2E  |irefox%2050.1.0.|
00000070  64 6D 67 5F 10 32 68 74 74 70 73 3A 2F 2F 77 77  |dmg_.2https://ww|
00000080  77 2E 6D 6F 7A 69 6C 6C 61 2E 6F 72 67 2F 65 6E  |w.mozilla.org/en|
00000090  2D 55 53 2F 66 69 72 65 66 6F 78 2F 6E 65 77 2F  |-US/firefox/new/|
000000A0  3F 73 63 65 6E 65 3D 32 08 0B 73 00 00 00 00 00  |?scene=2..s.....|
000000B0  00 01 01 00 00 00 00 00 00 00 03 00 00 00 00 00  |................|
000000C0  00 00 00 00 00 00 00 00 00 00 A8                 |...........|
000000cb
com.apple.quarantine:
00000000  30 30 38 33 3B 35 38 35 65 61 62 64 65 3B 53 61  |0083;585eabde;Sa|
00000010  66 61 72 69 3B 36 39 35 41 44 44 35 37 2D 45 37  |fari;695ADD57-E7|
00000020  32 32 2D 34 46 34 45 2D 38 46 39 39 2D 35 45 46  |22-4F4E-8F99-5EF|
00000030  37 41 38 44 46 39 32 39 33                       |7A8DF9293|
00000039
```


Similar to the previous example, this shows one more extended attribute that gets written when a file is downloaded using the Safari browser.

The "`com.apple.metadata:kMDItemDownloadedDate`" attribute contains a binary plist file that holds the download date.

Another browser, Firefox, will only show the "`com.apple.quarantine`" attribute information; it does not write the other two. Different applications can use different extended attributes.

Extended Attributes: Downloaded Files [3]

Chrome Download

```
bit:Downloads oompa$ xattr -xl HexFiend.zip
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 35 68 74  |bplist00..._.5ht|
00000010  74 70 3A 2F 2F 72 69 64 69 63 75 6C 6F 75 73 66  |tp://ridiculousf|
00000020  69 73 68 2E 63 6F 6D 2F 68 65 78 66 69 65 6E 64  |ish.com/hexfiend|
00000030  2F 66 69 6C 65 73 2F 48 65 78 46 69 65 6E 64 2E  |/files/HexFiend.|
00000040  7A 69 70 5F 10 23 68 74 74 70 3A 2F 2F 72 69 64  |zip_.#http://rid|
00000050  69 63 75 6C 6F 75 73 66 69 73 68 2E 63 6F 6D 2F  |iculousfish.com/|
00000060  68 65 78 66 69 65 6E 64 2F 08 0B 43 00 00 00 00  |hexfiend/..C....|
00000070  00 00 01 01 00 00 00 00 00 00 00 03 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00 00 00 00 69              |...........i|
0000008c
com.apple.quarantine:
00000000  30 30 38 31 3B 35 38 34 38 63 37 65 35 3B 47 6F  |0081;5848c7e5;Go|
00000010  6F 67 6C 65 20 43 68 72 6F 6D 65 3B 34 30 37 35  |ogle Chrome;4075|
00000020  31 33 37 43 2D 31 39 33 44 2D 34 44 39 42 2D 42  |137C-193D-4D9B-B|
00000030  41 32 46 2D 46 36 34 42 44 38 46 39 46 31 32 32  |A2F-F64BD8F9F122|
00000040
```


A file or directory on OS X may have additional metadata called extended attributes. These attributes were introduced in 10.4 as a way to incorporate more metadata functionality into OS X.

Extended attributes are stored as inline attributes in the HFS+ Attributes File.

Extended attributes can contain a variety of information. The attributes mainly found on files downloaded with the Chrome browser include:

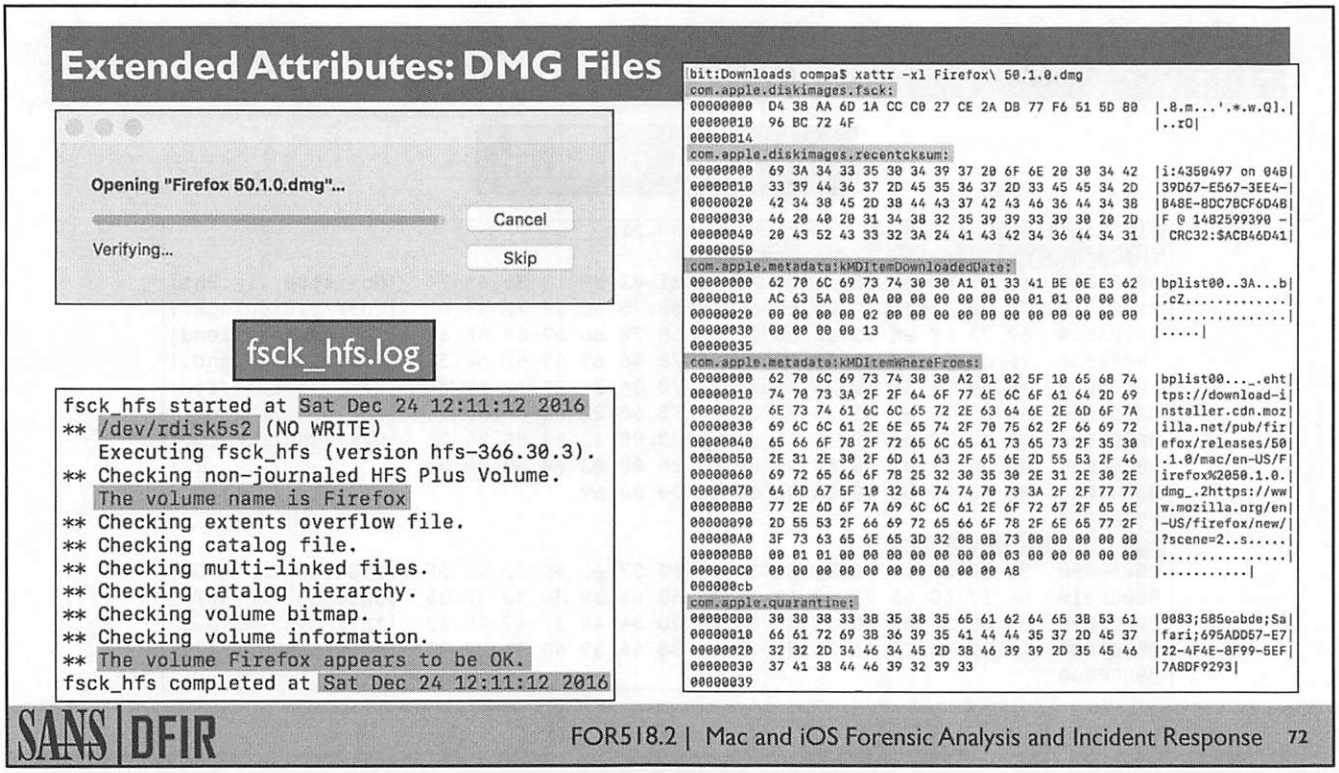
- `com.apple.metadata:kMDItemWhereFroms:` Where the file was downloaded from and the referring URL
- `com.apple.quarantine:` When the file was downloaded (hex Unix epoch value) and what application it was downloaded with

Extended attributes can be viewed by using the `xattr` command shown above.

The `xattr` command has the following output options:

- `-x:` View in hex
- `-l:` Outputs the attribute name and contents

Reference:
Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin—Chapter 16: HFS+ File System Concepts

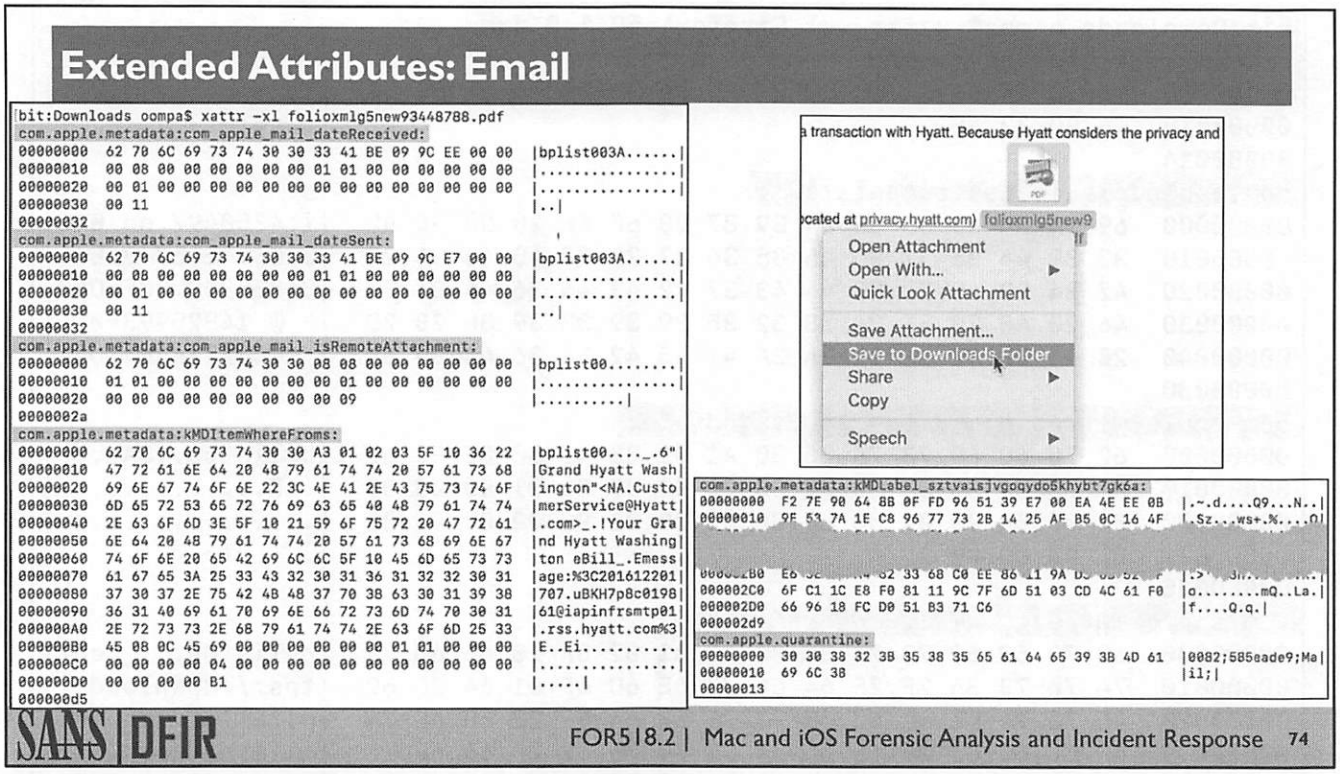


This screenshot shows another type of extended attribute specific to DMG files. Once a DMG file has been double-clicked and opened, two extended attributes get written.

- `com.apple.diskimages.fsck`: File System Check information
- `com.apple.diskimages.recentcksum`: Checksum information, including a Unix epoch timestamp of when the file was downloaded

These additional extended attributes show that a DMG was actually opened at least once and not just downloaded. The first open timestamp from this process can be found in the user's `~/Library/Logs/fsck_hfs.log` file.

```
[bit:Downloads oompa$ xattr -xl Firefox\ 50.1.0.dmg
com.apple.diskimages.fsck:
00000000  D4 38 AA 6D 1A CC C0 27 CE 2A DB 77 F6 51 5D 80  |.8.m...'.*.w.Q].|
00000010  96 BC 72 4F                                      |..rO|
00000014
com.apple.diskimages.recentcksum:
00000000  69 3A 34 33 35 30 34 39 37 20 6F 6E 20 30 34 42  |i:4350497 on 04B|
00000010  33 39 44 36 37 2D 45 35 36 37 2D 33 45 45 34 2D  |39D67-E567-3EE4-|
00000020  42 34 38 45 2D 38 44 43 37 42 43 46 36 44 34 38  |B48E-8DC7BCF6D48|
00000030  46 20 40 20 31 34 38 32 35 39 39 33 39 30 20 2D  |F @ 1482599390 -|
00000040  20 43 52 43 33 32 3A 24 41 43 42 34 36 44 34 31  | CRC32:$ACB46D41|
00000050
com.apple.metadata:kMDItemDownloadedDate:
00000000  62 70 6C 69 73 74 30 30 A1 01 33 41 BE 0E E3 62  |bplist00..3A...b|
00000010  AC 63 5A 08 0A 00 00 00 00 00 00 01 01 00 00 00  |.cZ.............|
00000020  00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 13                                   |.....|
00000035
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 5F 10 65 68 74  |bplist00..._.eht|
00000010  74 70 73 3A 2F 2F 64 6F 77 6E 6C 6F 61 64 2D 69  |tps://download-i|
00000020  6E 73 74 61 6C 6C 65 72 2E 63 64 6E 2E 6D 6F 7A  |nstaller.cdn.moz|
00000030  69 6C 6C 61 2E 6E 65 74 2F 70 75 62 2F 66 69 72  |illa.net/pub/fir|
00000040  65 66 6F 78 2F 72 65 6C 65 61 73 65 73 2F 35 30  |efox/releases/50|
00000050  2E 31 2E 30 2F 6D 61 63 2F 65 6E 2D 55 53 2F 46  |.1.0/mac/en-US/F|
00000060  69 72 65 66 6F 78 25 32 30 35 30 2E 31 2E 30 2E  |irefox%2050.1.0.|
00000070  64 6D 67 5F 10 32 68 74 74 70 73 3A 2F 2F 77 77  |dmg_.2https://ww|
00000080  77 2E 6D 6F 7A 69 6C 6C 61 2E 6F 72 67 2F 65 6E  |w.mozilla.org/en|
00000090  2D 55 53 2F 66 69 72 65 66 6F 78 2F 6E 65 77 2F  |-US/firefox/new/|
000000A0  3F 73 63 65 6E 65 3D 32 08 0B 73 00 00 00 00 00  |?scene=2..s.....|
000000B0  00 01 01 00 00 00 00 00 00 00 03 00 00 00 00 00  |................|
000000C0  00 00 00 00 00 00 00 00 00 A8                    |..........|
000000cb
com.apple.quarantine:
00000000  30 30 38 33 3B 35 38 35 65 61 62 64 65 3B 53 61  |0083;585eabde;Sa|
00000010  66 61 72 69 3B 36 39 35 41 44 44 35 37 2D 45 37  |fari;695ADD57-E7|
00000020  32 32 2D 34 46 34 45 2D 38 46 39 39 2D 35 45 46  |22-4F4E-8F99-5EF|
00000030  37 41 38 44 46 39 32 39 33                       |7A8DF9293|
00000039
```



On some newer systems, when an email attachment has been downloaded, a few extended attributes get attached to that file.

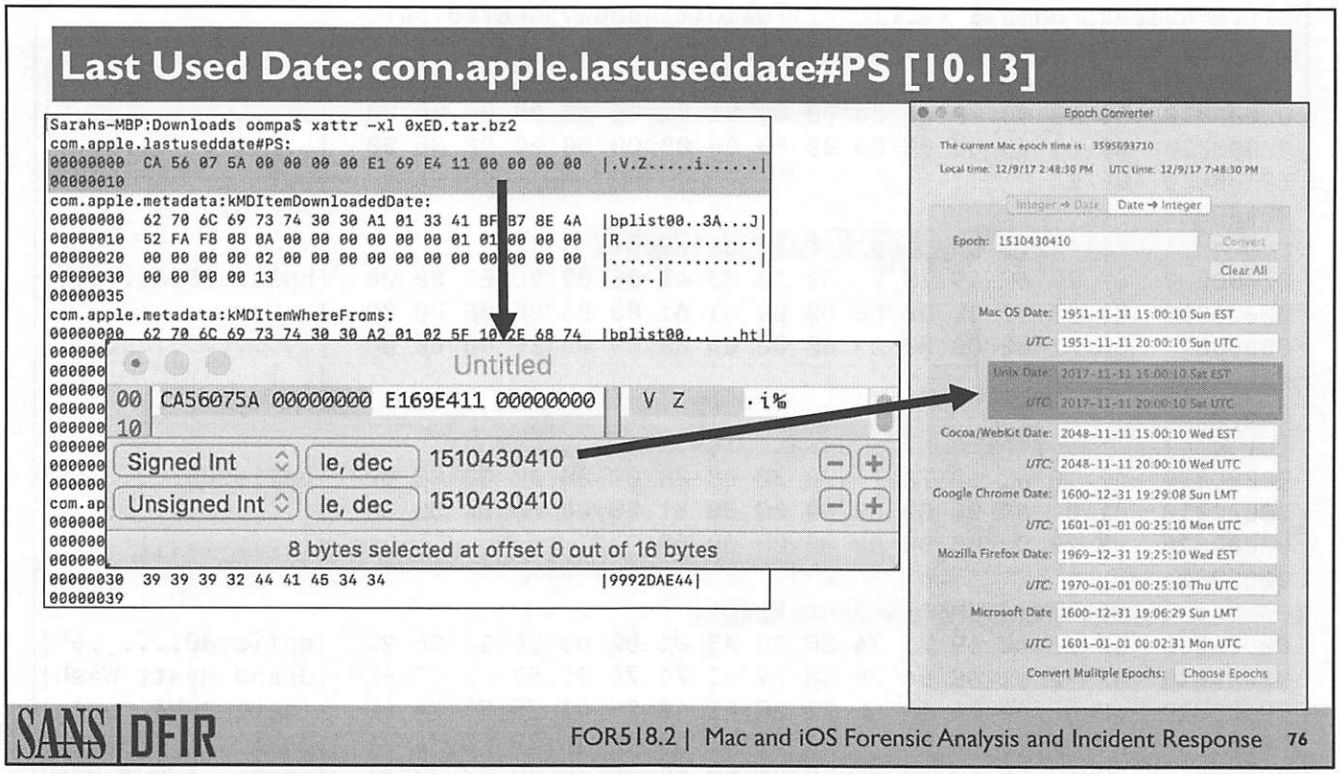
- `com.apple.metadata:com_apple_mail_dateReceived`: Binary plist containing the timestamp of when the email message was received
- `com.apple.metadata:com_apple_mail_dateSent`: Binary plist containing the timestamp of when the email message was sent
- `com.apple.metadata:com_apple_mail_isRemoteAttachment`: Binary plist containing a binary value if the attachment is local (0) or remote (1)

Another extended attribute, "`com.apple.metadata:kMDLabel_sztvaisjvgoqydo5khybt7gk6a`", contains unknown, possibly encrypted data; however, it does appear to get written on downloaded email attachments and other files sent/received. The string at the end of the attribute name appears to be unique to specific hardware.

Note the application found in the "`com.apple.quarantine`" attribute. It shows the file was downloaded using "Mail". The hex timestamp shows when the file was downloaded, as it does with other files.

```
bit:Downloads oompa$ xattr -xl folioxmlg5new93448788.pdf
com.apple.metadata:com_apple_mail_dateReceived:
00000000  62 70 6C 69 73 74 30 30 33 41 BE 09 9C EE 00 00  |bplist003A......|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                            |..|
00000032
com.apple.metadata:com_apple_mail_dateSent:
00000000  62 70 6C 69 73 74 30 30 33 41 BE 09 9C E7 00 00  |bplist003A......|
00000010  00 08 00 00 00 00 00 00 01 01 00 00 00 00 00 00  |................|
00000020  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000030  00 11                                            |..|
00000032
com.apple.metadata:com_apple_mail_isRemoteAttachment:
00000000  62 70 6C 69 73 74 30 30 08 08 00 00 00 00 00 00  |bplist00........|
00000010  01 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00 00 09                    |..........|
0000002a
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A3 01 02 03 5F 10 36 22  |bplist00...._.6"|
00000010  47 72 61 6E 64 20 48 79 61 74 74 20 57 61 73 68  |Grand Hyatt Wash|
00000020  69 6E 67 74 6F 6E 22 3C 4E 41 2E 43 75 73 74 6F  |ington"<NA.Custo|
00000030  6D 65 72 53 65 72 76 69 63 65 40 48 79 61 74 74  |merService@Hyatt|
00000040  2E 63 6F 6D 3E 5F 10 21 59 6F 75 72 20 47 72 61  |.com>_.!Your Gra|
00000050  6E 64 20 48 79 61 74 74 20 57 61 73 68 69 6E 67  |nd Hyatt Washing|
00000060  74 6F 6E 20 65 42 69 6C 6C 5F 10 45 6D 65 73 73  |ton eBill_.Emess|
00000070  61 67 65 3A 25 33 43 32 30 31 36 31 32 32 30 31  |age:%3C201612201|
00000080  37 30 37 2E 75 42 4B 48 37 70 38 63 30 31 39 38  |707.uBKH7p8c0198|
00000090  36 31 40 69 61 70 69 6E 66 72 73 6D 74 70 30 31  |61@iapinfrsmtp01|
000000A0  2E 72 73 73 2E 68 79 61 74 74 2E 63 6F 6D 25 33  |.rss.hyatt.com%3|
000000B0  45 08 0C 45 69 00 00 00 00 00 00 01 01 00 00 00  |E..Ei...........|
000000C0  00 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00  |................|
000000D0  00 00 00 00 B1                                   |.....|
000000d5
```

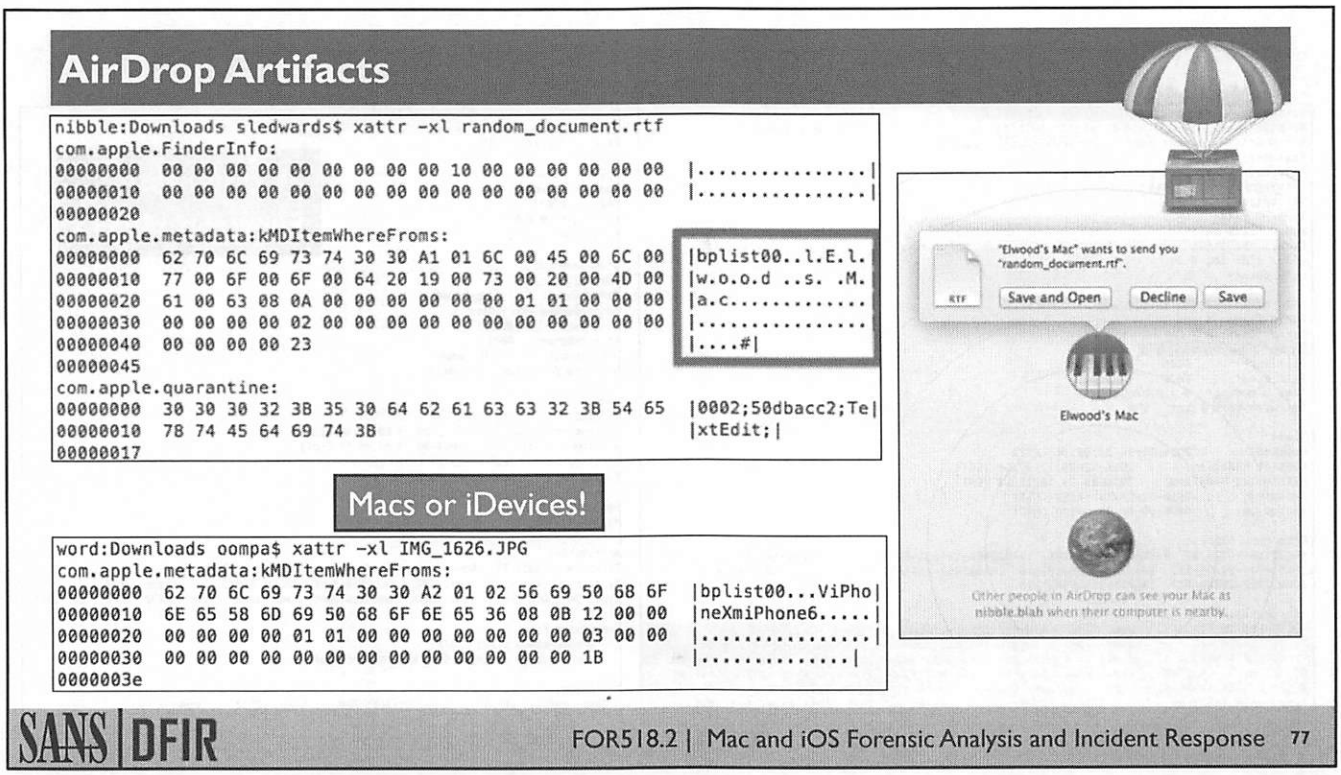
```
com.apple.metadata:kMDLabel_sztvaisjvgoqydo5khybt7gk6a:
00000000  F2 7E 90 64 8B 0F FD 96 51 39 E7 00 EA 4E EE 0B  |.~.d....Q9...N..|
00000010  9E 53 7A 1E C8 96 77 73 2B 14 25 AE B5 0C 16 4F  |.Sz...ws+.%....O|
[redacted]
000002B0  E6 [redacted] 32 33 68 C0 EE 86 11 9A [redacted]  |.>[redacted]3h[redacted]|
000002C0  6F C1 1C E8 F0 81 11 9C 7F 6D 51 03 CD 4C 61 F0  |o........mQ..La.|
000002D0  66 96 18 FC D0 51 B3 71 C6                       |f....Q.q.|
000002d9
com.apple.quarantine:
00000000  30 30 38 32 3B 35 38 35 65 61 64 65 39 3B 4D 61  |0082;585eade9;Ma|
00000010  69 6C 3B                                         |il;|
00000013
```



New extended attributes show up all the time. This one is particularly useful to forensic analysts. This one will keep a timestamp of when a file was last used, as it pertains to the file system.

This will generally be seen when the files have been used in the Finder windows; however, if a file is opened using the "open" command in the terminal, the timestamp will also be updated.

Not all file types will have this attribute, and research continues to determine which ones do and do not.



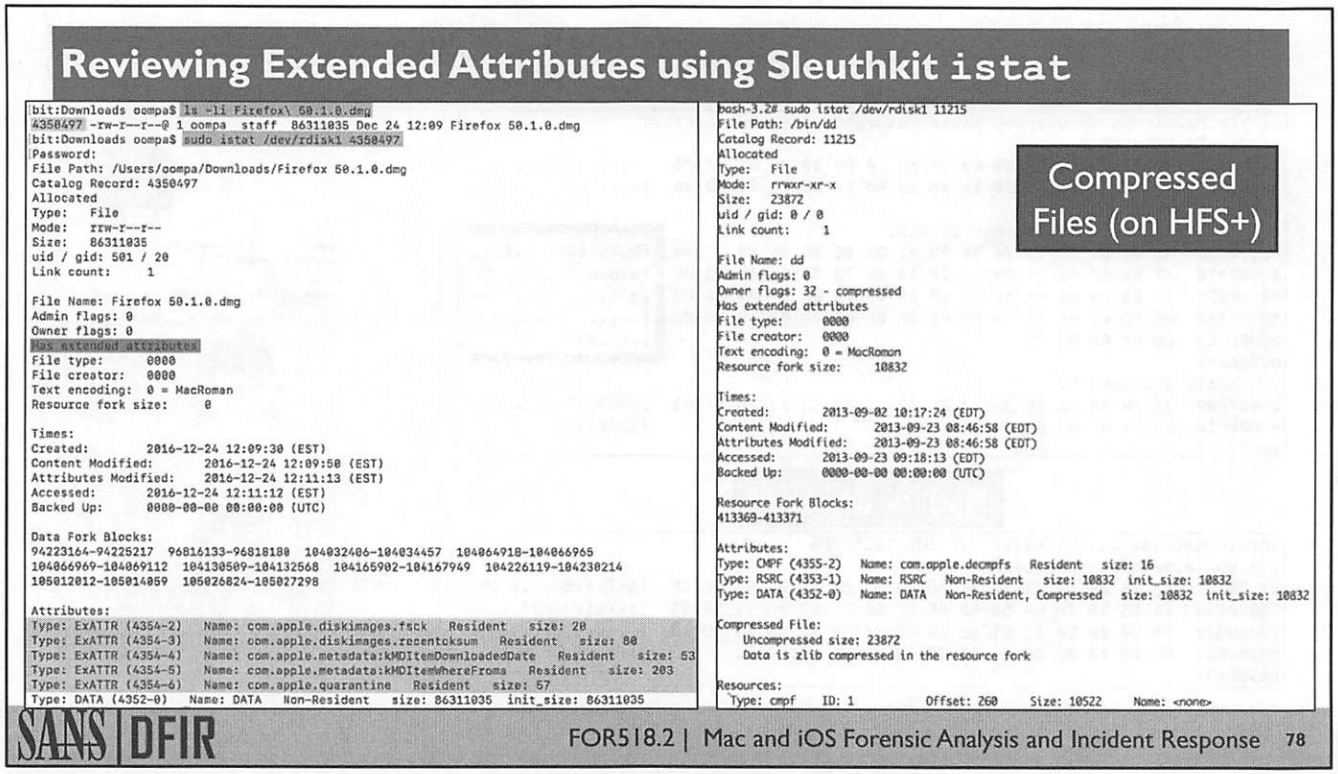
AirDrop is available on newer 10.7+ systems. This application allows users to "drop" files to other users in their vicinity. Note: This does not necessarily mean the user has to be on the same network.

In the example on the left, Elwood's Mac wants to send a document. If this file is saved, it will by default be saved to the `~/Downloads` directory.

Extended attributes for the file on the recipient's system show the file was from Elwood's Mac, shown in the top screenshot.

Be on the lookout for extended attributes with iPhone or iPad designators. With macOS 10.10, users are now able to AirDrop with their iDevices.

```
word:Downloads oompa$ xattr -xl IMG_1626.JPG
com.apple.metadata:kMDItemWhereFroms:
00000000  62 70 6C 69 73 74 30 30 A2 01 02 56 69 50 68 6F  |bplist00...ViPho|
00000010  6E 65 58 6D 69 50 68 6F 6E 65 36 08 0B 12 00 00  |neXmiPhone6.....|
00000020  00 00 00 00 01 01 00 00 00 00 00 00 00 03 00 00  |................|
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 1B        |..............|
0000003e
```



The first command shown in the left screenshot, "`ls -li`", shows the file's inode (CNID) number that we can use with the next command.

The Sleuthkit's (TSK) `istat` command prints the names and sizes of extended attributes in its "`Attributes`" section, shown in the screenshot above for the file with CNID (inode) 4350497.

Each attribute contains a TSK-assigned attribute number. For extended attributes, this number is `4354-#`, where # is filled in for each individual attribute. This can be used to extract the contents of the attribute using Sleuthkit's `icat` utility, shown later.

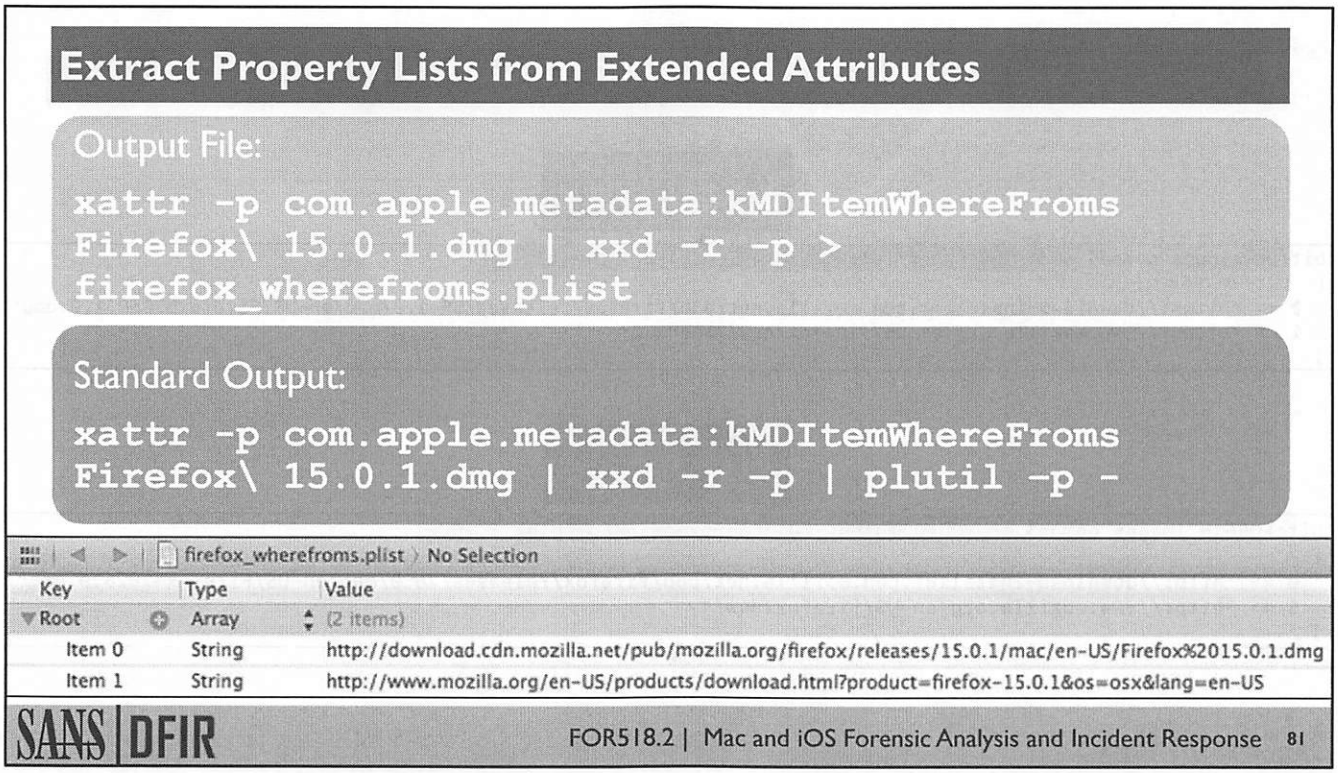
The right screenshot shows an example of the `/bin/dd` (CNID 11215) utility where the HFS+ file compression is used.

Note the attribute types:

- CMPF: Compressed File Data (com.apple.decmpfs stored in the $Attributes file)
- RSRC: Resource Fork
- DATA: Data Fork

```
bit:Downloads oompa$ ls -li Firefox\ 50.1.0.dmg
4350497 -rw-r--r--@ 1 oompa  staff  86311035 Dec 24 12:09 Firefox 50.1.0.dmg
bit:Downloads oompa$ sudo istat /dev/rdisk1 4350497
Password:
File Path: /Users/oompa/Downloads/Firefox 50.1.0.dmg
Catalog Record: 4350497
Allocated
Type:    File
Mode:    rrw-r--r--
Size:    86311035
uid / gid: 501 / 20
Link count:     1

File Name: Firefox 50.1.0.dmg
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:        2016-12-24 12:09:30 (EST)
Content Modified:       2016-12-24 12:09:50 (EST)
Attributes Modified:    2016-12-24 12:11:13 (EST)
Accessed:       2016-12-24 12:11:12 (EST)
Backed Up:      0000-00-00 00:00:00 (UTC)

Data Fork Blocks:
94223164-94225217  96816133-96818180  104032406-104034457  104064918-104066965
104066969-104069112  104130509-104132568  104165902-104167949  104226119-104230214
105012012-105014059  105026824-105027298

Attributes:
Type: ExATTR (4354-2)   Name: com.apple.diskimages.fsck   Resident   size: 20
Type: ExATTR (4354-3)   Name: com.apple.diskimages.recentcksum   Resident   size: 80
Type: ExATTR (4354-4)   Name: com.apple.metadata:kMDItemDownloadedDate   Resident   size: 53
Type: ExATTR (4354-5)   Name: com.apple.metadata:kMDItemWhereFroms   Resident   size: 203
Type: ExATTR (4354-6)   Name: com.apple.quarantine   Resident   size: 57
Type: DATA (4352-0)   Name: DATA   Non-Resident   size: 86311035  init_size: 86311035
```

```
bash-3.2# sudo istat /dev/rdisk1 11215
File Path: /bin/dd
Catalog Record: 11215
Allocated
Type:    File
Mode:    rrwxr-xr-x
Size:    23872
uid / gid: 0 / 0
Link count:      1

File Name: dd
Admin flags: 0
Owner flags: 32 - compressed
Has extended attributes
File type:       0000
File creator:    0000
Text encoding:   0 = MacRoman
Resource fork size:      10832

Times:
Created:         2013-09-02 10:17:24 (EDT)
Content Modified:        2013-09-23 08:46:58 (EDT)
Attributes Modified:     2013-09-23 08:46:58 (EDT)
Accessed:        2013-09-23 09:18:13 (EDT)
Backed Up:       0000-00-00 00:00:00 (UTC)

Resource Fork Blocks:
413369-413371

Attributes:
Type: CMPF (4355-2)   Name: com.apple.decmpfs   Resident   size: 16
Type: RSRC (4353-1)   Name: RSRC   Non-Resident   size: 10832  init_size: 10832
Type: DATA (4352-0)   Name: DATA   Non-Resident, Compressed   size: 10832  init_size: 10832

Compressed File:
    Uncompressed size: 23872
    Data is zlib compressed in the resource fork

Resources:
  Type: cmpf    ID: 1          Offset: 260     Size: 10522     Name: <none>
```



Some extended attributes contain binary property lists. To extract these for more detailed analysis, we can use the `xattr` command with the `-p` option. This option "prints" the attribute data.

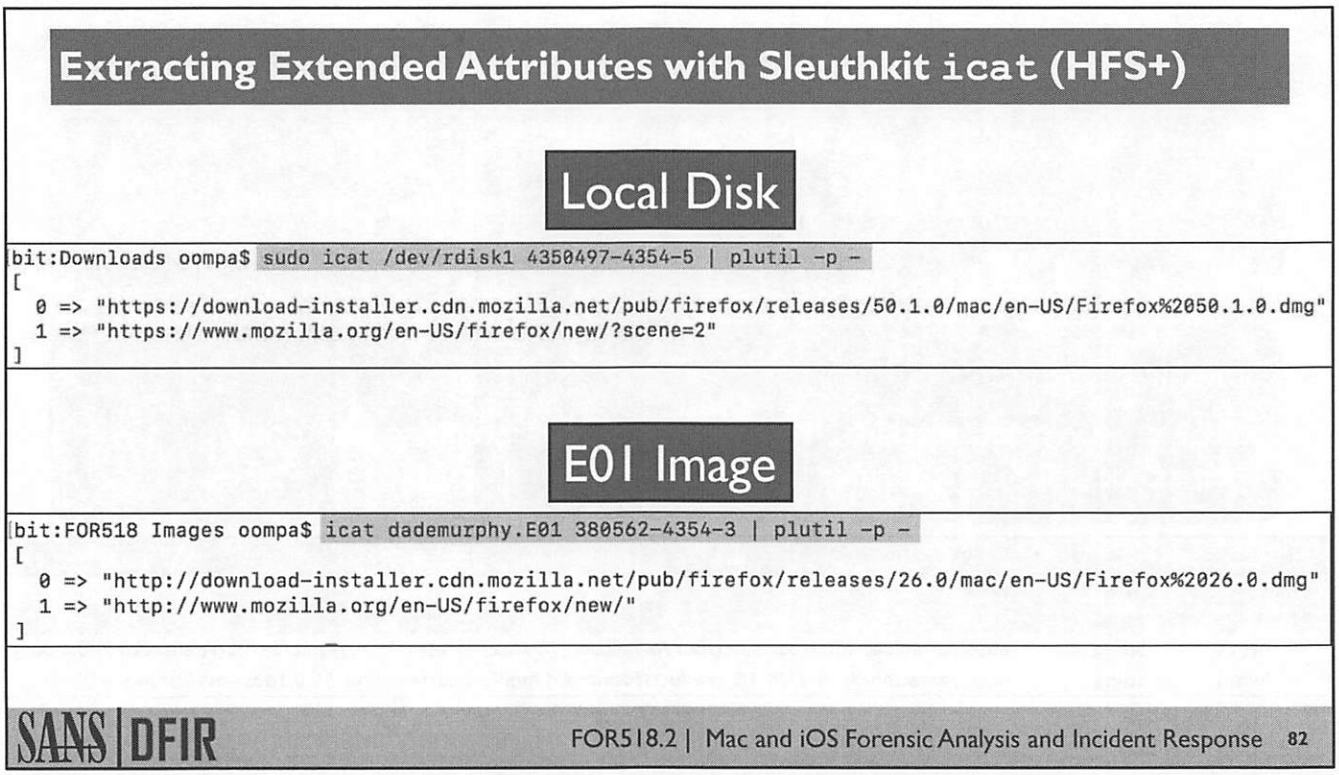
Using only the `-p` option, it will print the hex version of the data. To create a file, we must use the `xxd` command to "revert to binary" (`-r`) and print to "plaintext" (`-p`). We can use the ">" symbol to redirect the output to a file.

Together, the commands will look like this. This command prints the output of the `com.apple.metadata:kMDItemWhereFroms` attribute to a file named `firefox_wherefroms.plist`.

```
xattr -p com.apple.metadata:kMDItemWhereFroms Firefox\ 15.0.1.dmg | xxd -r -p
> firefox_wherefroms.plist
```

To print a binary plist directly to standard out on the Terminal, use the following command format:

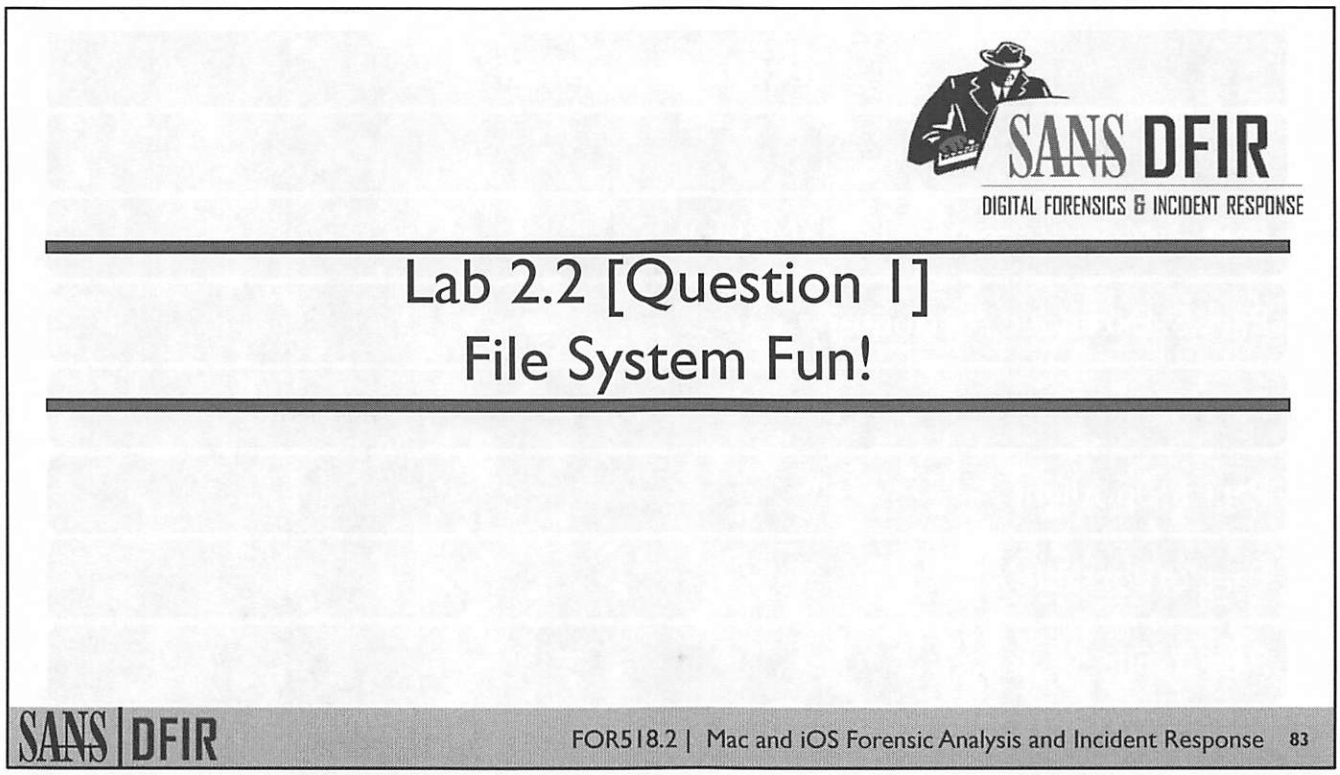
```
xattr -p <attribute name> <file> | xxd -r -p | plutil -p -
```



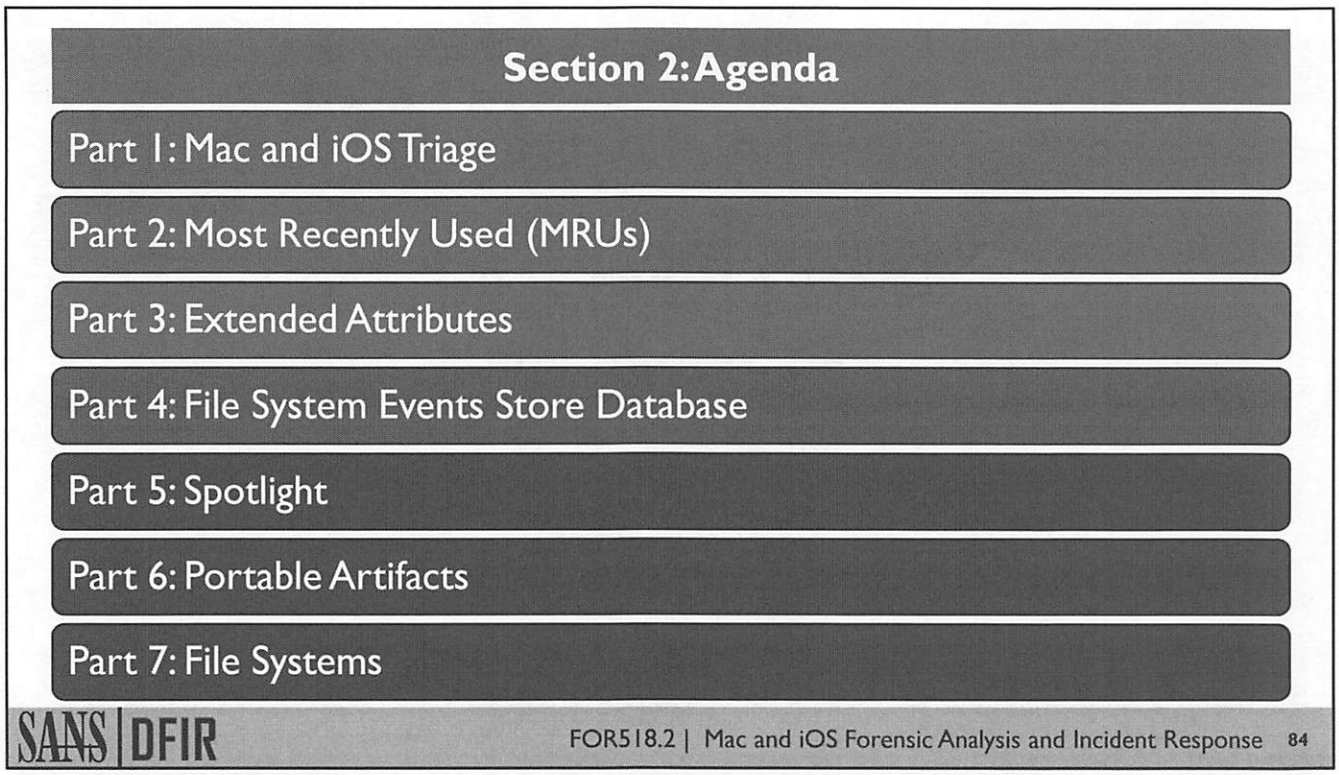
The Sleuthkit `icat` command can be used to print the data contained in a specific attribute. The TSK attribute number is appended to the CNID (inode) of the file to print the specific attribute.

The top screenshot shows an example of this using the local disk (`/dev/rdisk1`). The bottom screenshot is an example using an E01 image.

The screenshots show the `icat` command printing the extended attributes identified by `4354-5` for CNID `4350497` and `4354-3` for the CNID `380562`. This is the `com.apple.metadata:kMDItemWhereFroms` attribute for each file.



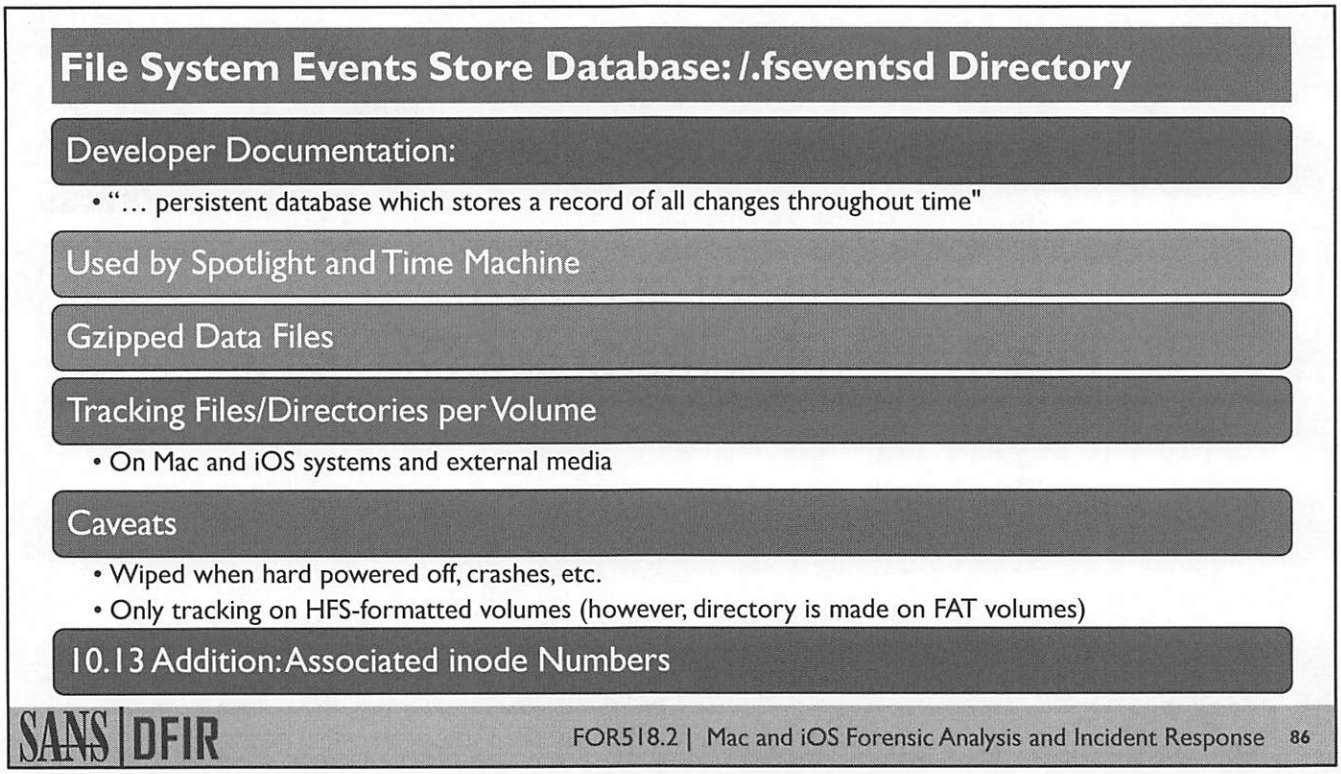
This page intentionally left blank.



This page intentionally left blank.

Section 2: Part 4
File System Events Store Database

This page intentionally left blank.



Each volume connected to a Mac system will have a `/.fseventsd` directory created. This directory is the File System Events Store Database that is responsible for storing file system changes on the volume.

Spotlight and Time Machine use this database to determine what files are new or have changed metadata properties. This directory contains gzipped files that require root privileges to unzip and view them. While this artifact can be incredibly useful, it does have some caveats. It can get wiped out when the system gets hard powered off or during a crash. However, if file carving is a possibility, these files can be found by searching for gzip files.

The `/.fseventsd` directory contains gzipped files, each named incrementally with no file extension. The `fseventsd-uuid` file contains the GUID of the FSEvents database. On an HDD volume, this should stay persistent, barring system malfunction. On an external USB drive, it is likely to change each time the drive is inserted into a system. FSEvent UUIDs will show changes in the system logs if you do a search for "`fsevents`". The /.fseventsd directory will be created on non-HFS+ volumes; however, it does not appear to create the database files.

The format of the unzipped files is noted to have changed over time in the Apple Developer Documentation and has not been thoroughly documented; however, filenames and file paths should be human readable. Therefore, we can run the strings utility on these files to find remnants of deleted directories or files. The results are shown on the next slide.

References:
Apple Developer Documentation: File System Events Programming Guide
`https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1`
"FSEvents and other interesting files" by Derrick Donnelly of BlackBag Technologies, Enfuse 2016
Apple Developer Documentation: FSEvents Reference
`https://developer.apple.com/library/content/documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html`

```
sh-3.2# pwd
/.fseventsd
sh-3.2# ls -lAr
total 74168
-rw-------  1 root  admin      36 Jun 30 17:09 fseventsd-uuid
-rw-------  1 root  admin   13488 Jul  6 10:33 0000000011ea5d4a
-rw-------  1 root  admin    3986 Jul  6 09:48 0000000011e942cb
-rw-------  1 root  admin   16438 Jul  6 09:39 0000000011e85439
-rw-------  1 root  admin   18365 Jul  6 08:38 00000000113fc77c
-rw-------  1 root  admin   17952 Jul  6 07:13 0000000011344ed6
-rw-------  1 root  admin    8247 Jul  6 04:07 000000001131f4eb
-rw-------  1 root  admin   12524 Jul  5 22:32 00000000112fcad6
-rw-------  1 root  admin   10181 Jul  5 20:50 0000000011267ce0
-rw-------  1 root  admin   20699 Jul  5 20:14 0000000011257dc1
-rw-------  1 root  admin   15236 Jul  5 20:13 0000000011242ded
```

```
sh-3.2# file *
00000000003fa974:    gzip compressed data, from Unix
0000000000417ee4:    gzip compressed data, from Unix
000000000042a918:    gzip compressed data, from Unix
0000000000450189:    gzip compressed data, from Unix
00000000004644c5:    gzip compressed data, from Unix
000000000046fa05:    gzip compressed data, from Unix
000000000047af8f:    gzip compressed data, from Unix
```

```
nibble:extracted sledwards$ xxd 00000000125482b4
0000000: 3153 4c44 0727 dc56 dc00 0000 0091 5554  1SLD.'.V......UT
0000010: 1200 0000 0000 0000 022e 4453 5f53 746f  ..........DS_Sto
0000020: 7265 00ca 8154 1200 0000 0055 0080 002e  re...T.....U....
0000030: 5472 6173 6865 7300 9354 5412 0000 0000  Trashes..TT.....
0000040: 4800 0001 476f 6f67 6c65 4368 726f 6d65  H...GoogleChrome
0000050: 5374 616e 6461 6c6f 6e65 456e 7465 7270  StandaloneEnterp
0000060: 7269 7365 2028 3129 2e6d 7369 009b 8254  rise (1).msi...T
0000070: 1200 0000 0055 0780 004b 656c 6968 6f73  .....U...Kelihos
0000080: 2d48 6c75 782d 3230 3133 2e7a 6970 00b3  -Hlux-2013.zip..
0000090: 8254 1200 0000 0055 0780 0062 6c61 6832  .T.....U...blah2
00000a0: 2e74 7874 00f4 5e54 1200 0000 0008 0080  .txt..^T........
00000b0: 0062 6c61 6833 2e74 7874 00d3 7254 1200  .blah3.txt..rT..
00000c0: 0000 0011 0180 0074 6573 742e 7478 7400  .......test.txt.
00000d0: 5c5e 5412 0000 0000 1500 8000            \^T.........
```

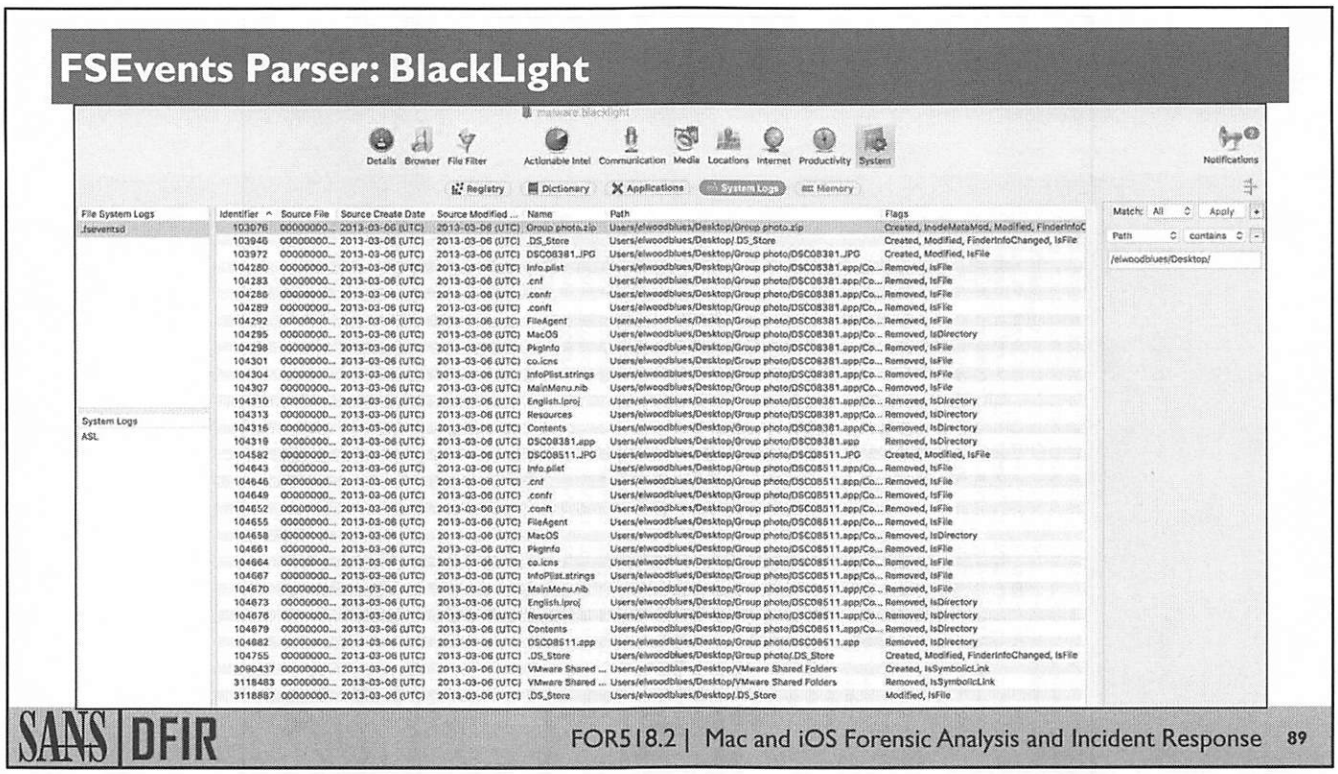
FSEvents Parser: Free Python Script
https://github.com/dlcowen/FSEventsParser

Python: SQLite Database or CSV Output

Table: fsevents

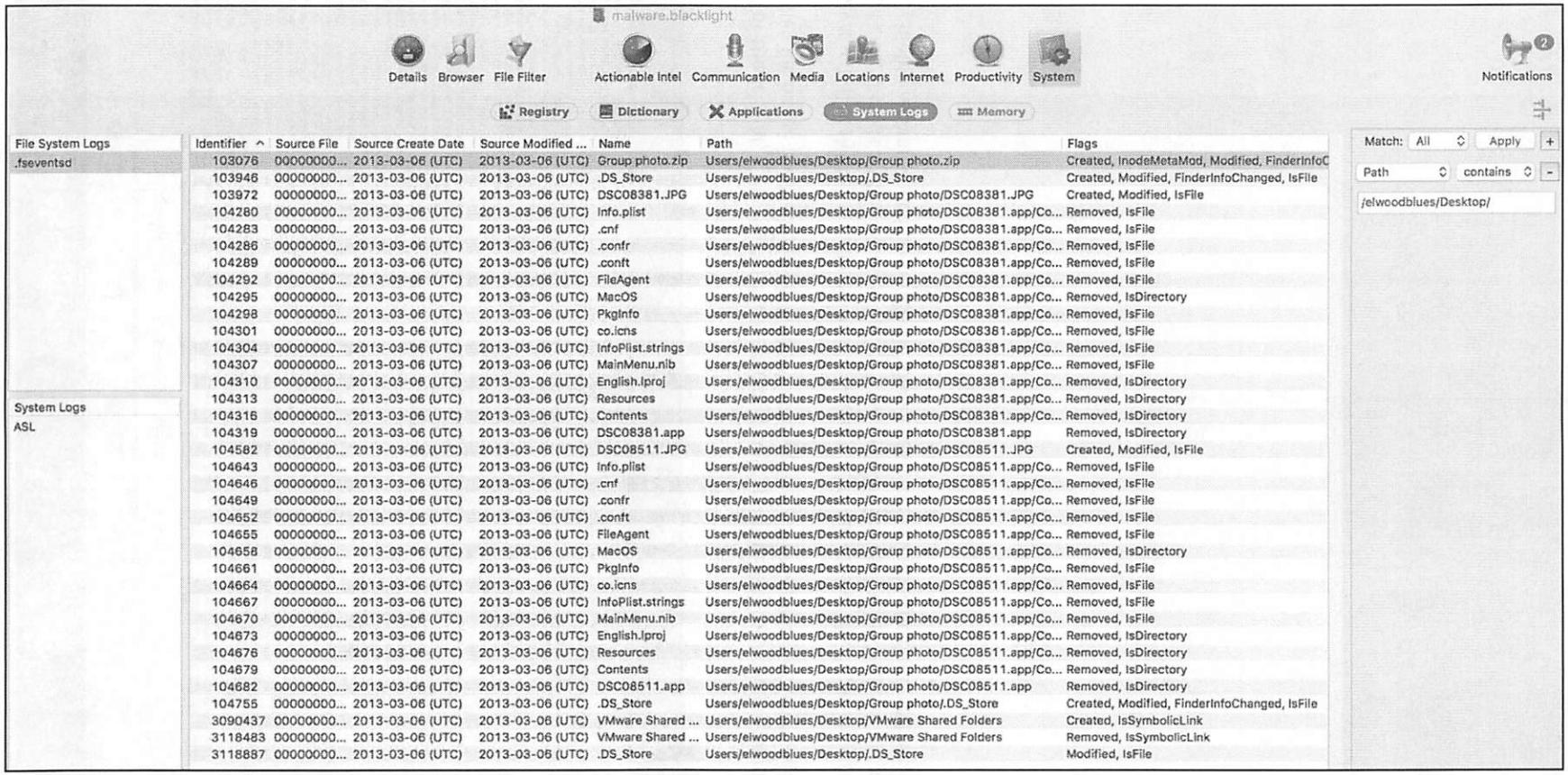
	record_filename	record_mask	record_mask_hex	name_date_strip	record_wd
	Filter	Filter	Filter	Filter	Filter
4	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/1DD5913F-789B-42DB-AFCB-6B84FF8320FA.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10459185
5	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/2515E71C-C9FC-43EF-AF59-D73A78E5F631.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10456565
6	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/2631C8E1-8EE9-48C4-86DB-878EDC929B02.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10474445
7	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/3C1D0A82-DE36-451A-A211-72C6D9368AF6.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10489409
8	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/480F88C6-675F-4896-A8F9-2D9F05503348.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10461231
9	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/626F9587-7F68-4357-B719-64C505E9535B.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10467913
10	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/66644644-A255-4DA4-B4BE-85C7E16529C0.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10476976
11	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/BBAB03AD-1018-4BAC-85A6-DE44331C0C3E.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10472117
12	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/CFEBB9A5-CF98-4CD5-97BB-12836E389734.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10466258
13	Users/oompa/Library/Calendars/4BCD81BB-AD04-4021-A206-18D749504F8E.calendar/Events/D2E48483-BDC8-4F99-AEF2-08D9F040E7C0.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10475257
14	Users/oompa/Library/Calendars/734F76E5-5012-4812-A372-B5CB17D15B61.calendar/Events/717A3166-CF95-41F1-9BE3-889BE0E35154.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10463051
15	Users/oompa/Library/Calendars/734F76E5-5012-4812-A372-B5CB17D15B61.calendar/Events/77EA15BF-2814-4BF4-969F-FE821B92D6F5.ics	ItemCreated;ItemIsDir;IgnoreSelf;ItemModified;	0x11028000	UNKNOWN	10489503

A Python script has been developed that allows forensic analysis of this data in a more human-readable format. This script takes in a directory of fsevent files and will output a SQLite database and CSV of parsed data. The data can then be analyzed in the analyst's preferred viewer.



In newer versions of BlackLight, a file system events parser is built in using the advanced processing capabilities. This can be run when the image is added or later in the "Evidence Status" processor on the left sidebar.

Once run, the analyst will find the items in the System | System Logs area of BlackLight, as shown in the example above. On the right side, there is a filter icon that can be used to filter the results for whatever the analyst might be looking for. This is highly recommended, as results for file system events tend to be quite verbose.



FSEvents Analysis: Example Scenario [1]

- Scenario: Newly created USB HFS+ volume
- Created directories and added files to mounted USB volume

	wd	mask_hex	filename	mask
1	18158642773887881568	0x00000002	NULL	Mount;
2	18158642773887883395	0x08000001	Directory_1	FolderEvent;Renamed;
3	18158642773887883520	0x08000001	Directory_2	FolderEvent;Renamed;
4	18158642773887883566	0x88000001	untitled folder	FolderEvent;Renamed;FolderCreated;
5	18158642773887883567	0x08000001	Directory_3	FolderEvent;Renamed;
6	18158642773887885468	0x55078000	Directory_1/DropboxInstaller.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
7	18158642773887885528	0x55038000	Directory_1/ExifTool-10.36.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
8	18158642773887885607	0x55078000	Directory_1/HexFiend.zip	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
9	18158642773887886915	0x55078000	Directory_2/IMG_2311.JPG.jpeg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
10	18158642773887886936	0x55078000	Directory_2/IMG_2326.JPG	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
11	18158642773887887932	0x55008000	.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
12	18158642773887887966	0x88000001	Directory_3/untitled folder	FolderEvent;Renamed;FolderCreated;
13	18158642773887887969	0x08000001	Directory_3/Directory_4	FolderEvent;Renamed;

Using the free Python script to parse the file system events, they can be viewed and queried in a SQLite database, as shown for the next few examples.

The scenario used for the following examples shows a new clean USB thumb drive that was wiped and reformatted with HFS+. The volume was created and mounted, and three directories were added to it: `Directory_1`, `Directory_2`, and `Directory_3`. The file system events show a "`Mount`" activity with three "`FolderEvent;Renamed`" activities for each of the three directories. This is because the first directory created was initially called "`untitled folder`", as shown above, which just got renamed.

The next section shows files being added to various directories: Installer DMG/ZIP files to `Directory_1`, images to `Directory_2`, and a new folder created in `Directory_3`, named `Directory_4`. These have the "`Created`" activity associated with each one.

Note the `.DS_Store` is shown. This means the user added these files using the Finder GUI window.

Reference:
`https://github.com/dlcowen/FSEventsParser`

	wd	mask_hex	filename	mask
1	18158642773887881568	0x00000002	NULL	Mount;
2	18158642773887883395	0x08000001	Directory_1	FolderEvent;Renamed;
3	18158642773887883520	0x08000001	Directory_2	FolderEvent;Renamed;
4	18158642773887883566	0x88000001	untitled folder	FolderEvent;Renamed;FolderCreated;
5	18158642773887883567	0x08000001	Directory_3	FolderEvent;Renamed;
6	18158642773887885468	0x55078000	Directory_1/DropboxInstaller.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
7	18158642773887885528	0x55038000	Directory_1/ExifTool-10.36.dmg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
8	18158642773887885607	0x55078000	Directory_1/HexFiend.zip	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
9	18158642773887886915	0x55078000	Directory_2/IMG_2311.JPG.jpeg	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
10	18158642773887886936	0x55078000	Directory_2/IMG_2326.JPG	Modified;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
11	18158642773887887932	0x55008000	.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
12	18158642773887887968	0x88000001	Directory_3/untitled folder	FolderEvent;Renamed;FolderCreated;
13	18158642773887887969	0x08000001	Directory_3/Directory_4	FolderEvent;Renamed;

FSEvents Analysis: Example Scenario [2]

- Created a new text document with TextEdit (Document Versions)

	filename	mask
	Filter	Filter
15	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoRBFc/Untitled 9.rtf	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
16	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoRBFc	FolderEvent;Removed;FolderCreated;
17	.DocumentRevisions-V100	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
18	.DocumentRevisions-V100/db-V1	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
19	.DocumentRevisions-V100/purgatory	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
20	.DocumentRevisions-V100/.cs	FolderEvent;FolderCreated;
21	.DocumentRevisions-V100/.cs/ChunkStoreDatabase	Created;FileEvent;
22	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
23	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-wal	InodeMetaMod;Created;FileEvent;
24	.DocumentRevisions-V100/db-V1/db.sqlite	Created;FileEvent;
25	.DocumentRevisions-V100/db-V1/db.sqlite-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
26	.DocumentRevisions-V100/db-V1/db.sqlite-wal	InodeMetaMod;Created;FileEvent;
27	.TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
28	.TemporaryItems/folders.0	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
29	.TemporaryItems/folders.0/TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
30	.DocumentRevisions-V100/staging	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
31	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/metadata	Modified;Renamed;Created;FileEvent;
32	.DocumentRevisions-V100/metadata	Renamed;PermissionChange;FileEvent;
33	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/LibraryStatus	Modified;Renamed;Created;FileEvent;
34	.DocumentRevisions-V100/LibraryStatus	Renamed;FileEvent;
35	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)	FolderEvent;Removed;FolderCreated;
36	Directory_3/Directory_4/Untitled 9.rtf	Renamed;ExtendedAttrModified;FinderInfoMod;FileEvent;
37	Directory_3/Directory_4/testdoc.rtf	Renamed;FileEvent;

Next, the user opened the TextEdit application and created an RTF document on the USB thumb drive. The initial name of the document was "`Untitled 9.rtf`". Once the user saved and renamed the document, it was saved as "`testdoc.rtf`". You can see temporary files being created before the document is actually saved as "`testdoc.rtf`".

Note all the "`.DocumentRevisions`" and "`.TemporaryItems`" file paths. This is how macOS Document Versions works; the details will be revealed in a later module.

Reference:
`https://github.com/dlcowen/FSEventsParser`

	filename	mask
	Filter	Filter
15	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoR8Fc/Untitled 9.rtf	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;FileEvent;
16	Directory_3/Directory_4/Untitled 9.rtf.sb-3b0a3d1b-CoR8Fc	FolderEvent;Removed;FolderCreated;
17	.DocumentRevisions-V100	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
18	.DocumentRevisions-V100/db-V1	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
19	.DocumentRevisions-V100/purgatory	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
20	.DocumentRevisions-V100/.cs	FolderEvent;FolderCreated;
21	.DocumentRevisions-V100/.cs/ChunkStoreDatabase	Created;FileEvent;
22	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
23	.DocumentRevisions-V100/.cs/ChunkStoreDatabase-wal	InodeMetaMod;Created;FileEvent;
24	.DocumentRevisions-V100/db-V1/db.sqlite	Created;FileEvent;
25	.DocumentRevisions-V100/db-V1/db.sqlite-journal	Modified;InodeMetaMod;Created;Removed;FileEvent;
26	.DocumentRevisions-V100/db-V1/db.sqlite-wal	InodeMetaMod;Created;FileEvent;
27	.TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
28	.TemporaryItems/folders.0	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
29	.TemporaryItems/folders.0/TemporaryItems	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
30	.DocumentRevisions-V100/staging	FolderEvent;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
31	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/metadata	Modified;Renamed;Created;FileEvent;
32	.DocumentRevisions-V100/metadata	Renamed;PermissionChange;FileEvent;
33	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)/LibraryStatus	Modified;Renamed;Created;FileEvent;
34	.DocumentRevisions-V100/LibraryStatus	Renamed;FileEvent;
35	.TemporaryItems/folders.0/TemporaryItems/(A Document Being Saved By revisiond)	FolderEvent;Removed;FolderCreated;
36	Directory_3/Directory_4/Untitled 9.rtf	Renamed;ExtendedAttrModified;FinderInfoMod;FileEvent;
37	Directory_3/Directory_4/textdoc.rtf	Renamed;FileEvent;

FSEvents Analysis: Example Scenario [3]

- Unzipped HexFiend.zip into Directory_1

	mask_hex	filename	mask
77	0xc0010001	.TemporaryItems/folders.501/Cleanup At Startup	FolderEvent;PermissionChange;FinderInfoMod;FolderCreated;
78	0x88000001	Directory_1/.BAH.boX1A	FolderEvent;Renamed;FolderCreated;
79	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
80	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
81	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;

- ... <cut for brevity>

	mask_hex	filename	mask
212	0x8c030001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app	FolderEvent;Renamed;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
213	0x0a020001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj	FolderEvent;Renamed;ExtendedAttrModified;Removed;
214	0x55008000	Directory_1/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
215	0x84010001	.Trashes	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
216	0x84010001	.Trashes/501	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
217	0x88020001	Directory_1/Hex Fiend.app	FolderEvent;Renamed;ExtendedAttrModified;FolderCreated;
218	0x00028000	Directory_1/Hex Fiend.app/Contents/Library/LaunchServices/com.ridiculousfish.H...	ExtendedAttrModified;FileEvent;
219	0x00020001	Directory_1/Hex Fiend.app/Contents/Resources	FolderEvent;ExtendedAttrModified;

The user then unzipped the installer file `HexFiend.zip` into `Directory_1`. The unzipping process creates many temporary files in ".TemporaryItems/". (For brevity, many are not shown.)

Once unzipped, you can see the "`Hex Fiend.app`" directory structure being written to disk with all the files that make up an application directory/bundle. The "`.Trashes`" folder is also created in this activity.

Reference:
`https://github.com/dlcowen/FSEventsParser`

	mask_hex	filename	mask
77	0xc0010001	.TemporaryItems/folders.501/Cleanup At Startup	FolderEvent;PermissionChange;FinderInfoMod;FolderCreated;
78	0x88000001	Directory_1/.BAH.boX1A	FolderEvent;Renamed;FolderCreated;
79	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
80	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;
81	0x1d018000	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app/Cont...	Modified;Renamed;InodeMetaMod;Created;PermissionChange;FileEvent;

	mask_hex	filename	mask
212	0x8c030001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj/Hex Fiend.app	FolderEvent;Renamed;InodeMetaMod;PermissionChange;ExtendedAttrModified;FolderCreated;
213	0x0a020001	.TemporaryItems/folders.501/Cleanup At Startup/.BAH.me1dj	FolderEvent;Renamed;ExtendedAttrModified;Removed;
214	0x55008000	Directory_1/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;FileEvent;
215	0x84010001	.Trashes	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
216	0x84010001	.Trashes/501	FolderEvent;InodeMetaMod;PermissionChange;FolderCreated;
217	0x88020001	Directory_1/Hex Fiend.app	FolderEvent;Renamed;ExtendedAttrModified;FolderCreated;
218	0x00028000	Directory_1/Hex Fiend.app/Contents/Library/LaunchServices/com.ridiculousfish.H...	ExtendedAttrModified;FileEvent;
219	0x00020001	Directory_1/Hex Fiend.app/Contents/Resources	FolderEvent;ExtendedAttrModified;

FSEvents Analysis: Example Scenario [4]

- ExifTool-10.36.dmg file moved directories
- File removal
  - HexFiend.zip sent to trash, then trash was emptied
    - "Removed" added to mask
  - IMG_2326.JPG sent to trash (not emptied)

	mask_hex	filename	mask
294	0x08008000	Directory_1/ExifTool-10.36.dmg	Renamed;FileEvent;
295	0x08008000	Directory_2/ExifTool-10.36.dmg	Renamed;FileEvent;
296	0x08008000	Directory_1/HexFiend.zip	Renamed;FileEvent;
297	0x14008000	Directory_1/.DS_Store	Modified;InodeMetaMod;FileEvent;
298	0x0a008000	.Trashes/501/HexFiend.zip	Renamed;Removed;FileEvent;
299	0x5d078000	IMG_2326.JPG	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
300	0x08008000	.Trashes/501/IMG_2326.JPG	Renamed;FileEvent;
301	0x57008000	.Trashes/501/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;Removed;FileEvent;
302	0x14008000	.DS_Store	Modified;InodeMetaMod;FileEvent;

Trashed, Trash Emptied

Just Trashed

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 97

Finally, the user wants to remove files from the USB thumb drive.

The first file, "`ExifTool-10.36.dmg`", was sent to the trash, and the trash was emptied explicitly by the user. Note the "`Removed`" event for this item.

Second, the "`IMG_2326.JPG`" file was sent to the trash, but the user did not empty it. The mask does not show the file was removed. It can still be recovered from the "`.Trashes/501/`" directory on the USB thumb drive.

Reference:
`https://github.com/dlcowen/FSEventsParser`

	mask_hex	filename	mask
294	0x08008000	Directory_1/ExifTool-10.36.dmg	Renamed;FileEvent;
295	0x08008000	Directory_2/ExifTool-10.36.dmg	Renamed;FileEvent;
296	0x08008000	Directory_1/HexFiend.zip	Renamed;FileEvent;
297	0x14008000	Directory_1/.DS_Store	Modified;InodeMetaMod;FileEvent;
298	0x0a008000	.Trashes/501/HexFiend.zip	Renamed;Removed;FileEvent;
299	0x5d078000	IMG_2326.JPG	Modified;Renamed;InodeMetaMod;Created;PermissionChange;ExtendedAttrModified;FinderInfoMod;ExtendedAttrRemoved;FileEvent;
300	0x08008000	.Trashes/501/IMG_2326.JPG	Renamed;FileEvent;
301	0x57008000	.Trashes/501/.DS_Store	Modified;InodeMetaMod;Created;FinderInfoMod;Removed;FileEvent;
302	0x14008000	.DS_Store	Modified;InodeMetaMod;FileEvent;

Trashed, Trash Emptied

Just Trashed

FSEvents Analysis: Volumes

```
1  select wd, mask_hex,filename, mask, source from fsevents where filename like '%Volumes/Google%'
```

	wd	mask_hex	filename	mask	source
1	3829041	0x00000002	/Volumes/Google Chrome	Mount;	./00000000003af26b
2	3828991	0x80010001	Volumes/Google Chrome	FolderEvent;PermissionChange;FolderCreated;	./00000000003af26b
3	4511193	0x00000004	/Volumes/Google Chrome	Unmount;	./000000000045b9cb
4	4511185	0x02000001	Volumes/Google Chrome	FolderEvent;Removed;	./000000000045b9cb
5	29046995	0x00000002	/Volumes/Google Earth	Mount;	./0000000001bc10a7
6	29046984	0x80010001	Volumes/Google Earth	FolderEvent;PermissionChange;FolderCreated;	./0000000001bc10a7
7	30498055	0x00000004	/Volumes/Google Earth	Unmount;	./0000000001d2510c
8	30498053	0x02000001	Volumes/Google Earth	FolderEvent;Removed;	./0000000001d2510c

Other events can show system usage, such as "`Mount`" and "`Unmount`" volume activities.

The example above shows the file system events on a macOS image. Whenever a volume gets mounted onto "`/Volumes`", it is recorded. This example shows two volumes being mounted/unmounted: One for Google Chrome and another for Google Earth. It will show a mount point being created (`FolderCreated`), as well as removed after it is unmounted (`FolderEvent;Removed;`).

Reference:
`https://github.com/dlcowen/FSEventsParser`

FSEvents Analysis: Temporal Context [1]

- Logs!

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/var/log/asl%'
```

	wd	filename	mask	source_created_time	source_modified_time	other_dates
20	23920394	private/var/log/asl/BB.2017.12.31.G80.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	UNKNOWN
21	24926258	private/var/log/asl/2016.12.07.G80.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
22	24924808	private/var/log/asl/2016.12.07.U501.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
23	24926270	private/var/log/asl/2016.12.08.G80.asl	Created;PermissionChange;ExtendedAttrMod...	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
24	24948823	private/var/log/asl/2016.12.08.U501.asl	Created;PermissionChange;ExtendedAttrMod...	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%tracev3%'
```

	wd	filename	mask
605	18158642773886498059	private/var/db/diagnostics/logdata.Persistent.20161225T175907.tracev3	Modified;Created;Removed;FileEvent;
606	18158642773886498082	private/var/db/diagnostics/logdata.Persistent.20161225T175909.tracev3	Modified;Created;Removed;FileEvent;
607	18158642773886498091	private/var/db/diagnostics/logdata.Persistent.20161225T175911.tracev3	Modified;Created;Removed;FileEvent;

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 100

All these file system events are being recorded; however, no timestamps are saved for each individual event. No problem—we'll just have to get creative!

The FSEvents Python script will show that the source created and modified timestamps: These are the file system timestamps of those fsevents gzip files. If the files are exported out of the image, they may show the wrong timestamps, as shown in the top screenshot; however, you can still go to the forensic image to get the correct timestamps. These timestamps can help filter down a scope of time the file activity occurred within.

The script also tries to guess timestamps by using timestamps within the filenames of nearby files, as shown in "`other_dates`". These timestamps are pulled from the ASL filenames shown in the top screenshot.

We can do the same with other files. In the bottom screenshot, there are plenty of other logs that contain timestamps within their filenames, such as the new unified logging files that end in "`.tracev3`". They follow the format `YYYYMMDD HHMMSS` and show the general time the messages were put into them.

Reference:
`https://github.com/dlcowen/FSEventsParser`

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/var/log/asl%'
```

	wd	filename	mask	source_created_time	source_modified_time	other_dates
20	23920394	private/var/log/asl/BB.2017.12.31.G80.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	UNKNOWN
21	24926258	private/var/log/asl/2016.12.07.G80.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
22	24924808	private/var/log/asl/2016.12.07.U501.asl	Modified;FileEvent;	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
23	24926270	private/var/log/asl/2016.12.08.G80.asl	Created;PermissionChange;ExtendedAttrMod...	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08
24	24948823	private/var/log/asl/2016.12.08.U501.asl	Created;PermissionChange;ExtendedAttrMod...	2016-12-26 15:07:14	2016-12-26 15:07:00	2016.12.07,2016.12.08

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%tracev3%'
```

	wd	filename	mask
605	18158642773886498059	private/var/db/diagnostics/logdata.Persistent.20161225T175907.tracev3	Modified;Created;Removed;FileEvent;
606	18158642773886498082	private/var/db/diagnostics/logdata.Persistent.20161225T175909.tracev3	Modified;Created;Removed;FileEvent;
607	18158642773886498091	private/var/db/diagnostics/logdata.Persistent.20161225T175911.tracev3	Modified;Created;Removed;FileEvent;

FSEvents Analysis: Temporal Context [2]

- Photo Library and Messages

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/Photos Library.photoslibrary/Masters/2016/05/18%'
```

	wd	filename	mask
1	18158642773871313107	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18	FolderEvent;ExtendedAttrModified;FolderCre...
2	18158642773871314490	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119	FolderEvent;ExtendedAttrModified;FolderCre...
3	18158642773871314585	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119/IMG_0315.PNG	Modified;Created;HardLink;FileEvent;
4	18158642773871313113	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823	FolderEvent;ExtendedAttrModified;FolderCre...
5	18158642773871313788	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823/IMG_0316.PNG	Modified;Created;HardLink;FileEvent;
6	18158642773871313408	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757	FolderEvent;ExtendedAttrModified;FolderCre...
7	18158642773871313797	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0317.PNG	Modified;Created;HardLink;FileEvent;
8	18158642773871313800	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0318.PNG	Modified;Created;HardLink;FileEvent;

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/Messages/Archive/%'
```

	wd	filename
40	39699407	Users/oompa/Library/Messages/Archive/2016-12-15/Heather Mahalik on 2016-12-15 at 17.47.23.ichat
41	39701691	Users/oompa/Library/Messages/Archive/2016-12-15/Chat with Heather Mahalik et al on 2016-12-15 at 20.10.47.ichat
42	39701654	Users/oompa/Library/Messages/Archive/2016-12-15/Heather Mahalik on 2016-12-15 at 17.47.23.ichat

102

We can also use some file paths from other applications, assuming the user is using these applications.

The top screenshot shows an example of timestamps from the Photos library. The photos are inherently organized by time frame.

The bottom screenshot shows chat messages being archived, which get recorded with a general timestamp of when the conversation occurred.

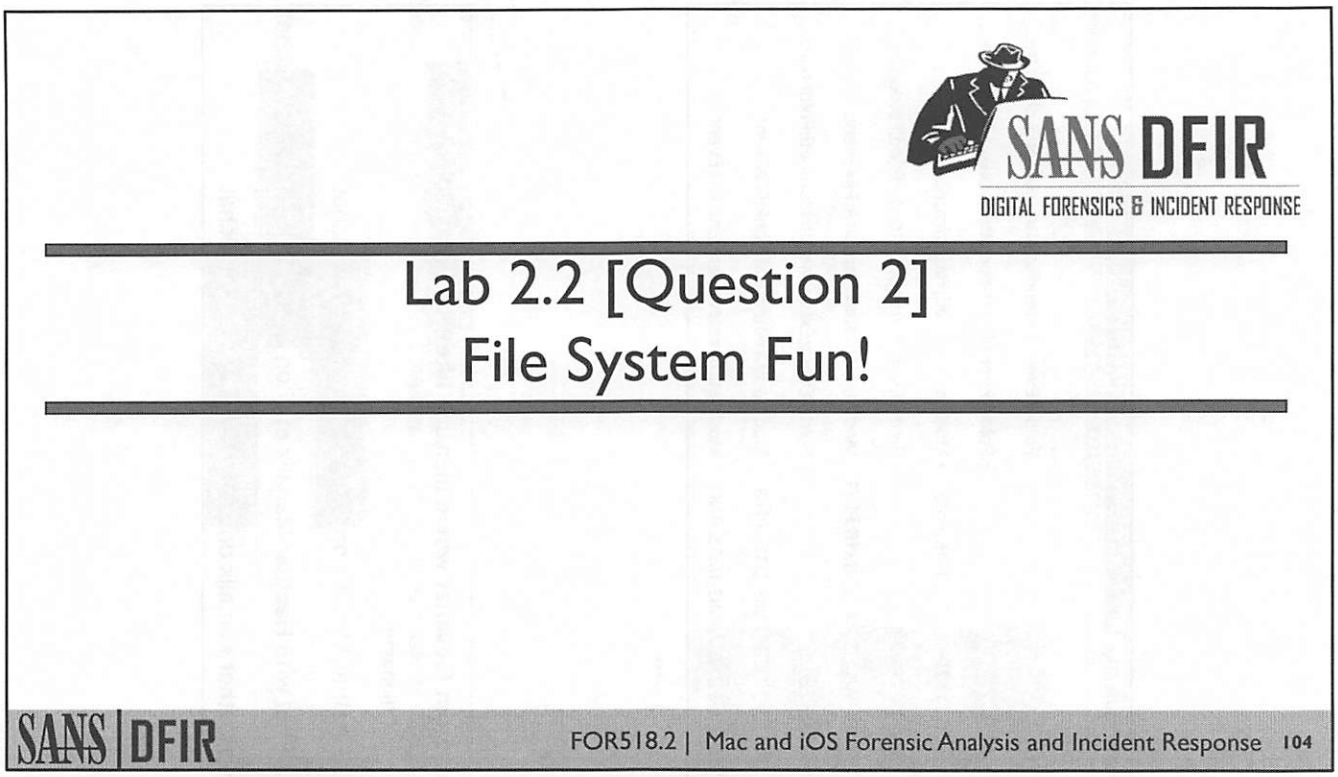
Reference:
`https://github.com/dlcowen/FSEventsParser`

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/Photos Library.photoslibrary/Masters/2016/05/18%'
```

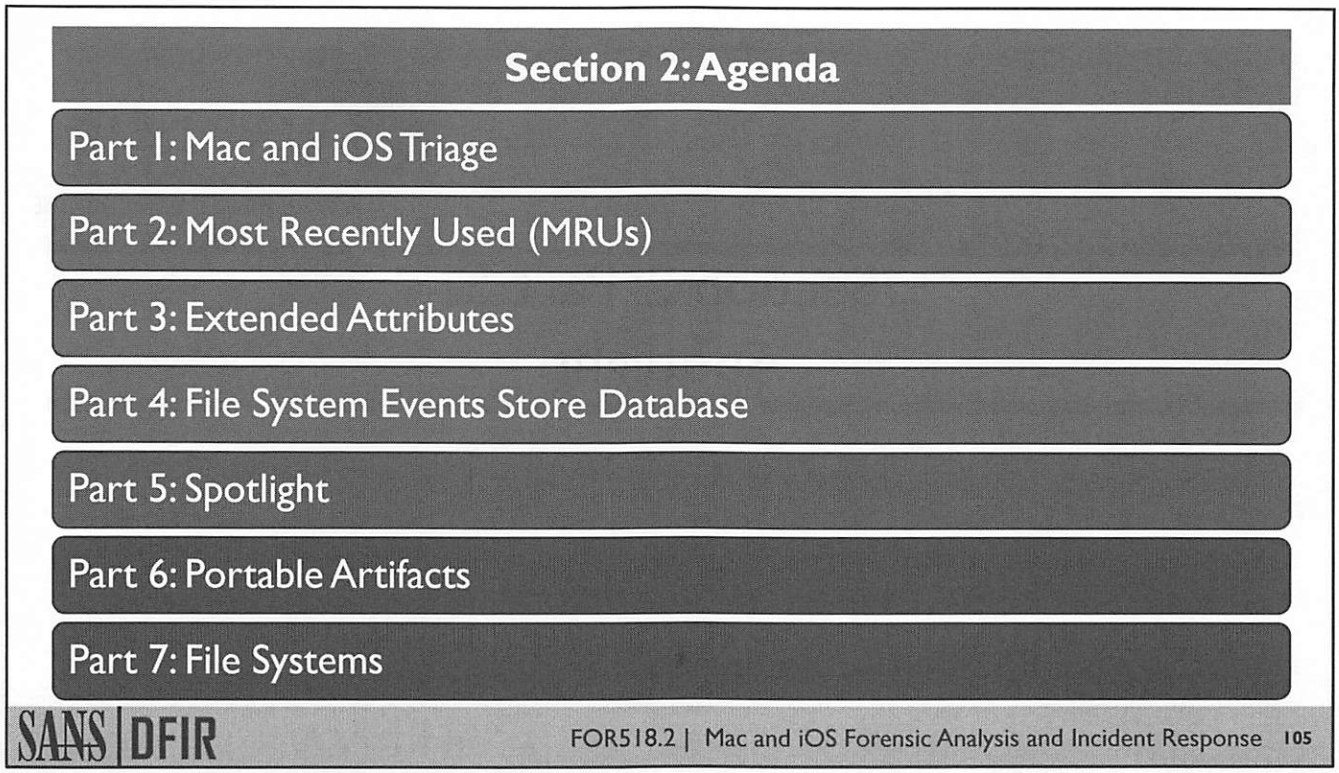
	wd	filename	mask
1	18158642773871313107	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18	FolderEvent;ExtendedAttrModified;FolderCre...
2	18158642773871314490	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119	FolderEvent;ExtendedAttrModified;FolderCre...
3	18158642773871314585	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-212119/IMG_0315.PNG	Modified;Created;HardLink;FileEvent;
4	18158642773871313113	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823	FolderEvent;ExtendedAttrModified;FolderCre...
5	18158642773871313788	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-220823/IMG_0316.PNG	Modified;Created;HardLink;FileEvent;
6	18158642773871313408	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757	FolderEvent;ExtendedAttrModified;FolderCre...
7	18158642773871313797	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0317.PNG	Modified;Created;HardLink;FileEvent;
8	18158642773871313800	Users/oompa/Pictures/Photos Library.photoslibrary/Masters/2016/05/18/20160518-221757/IMG_0318.PNG	Modified;Created;HardLink;FileEvent;

```
1  select wd,filename,mask, source_created_time,source_modified_time,other_dates from fsevents where filename like '%/Messages/Archive/%'
```

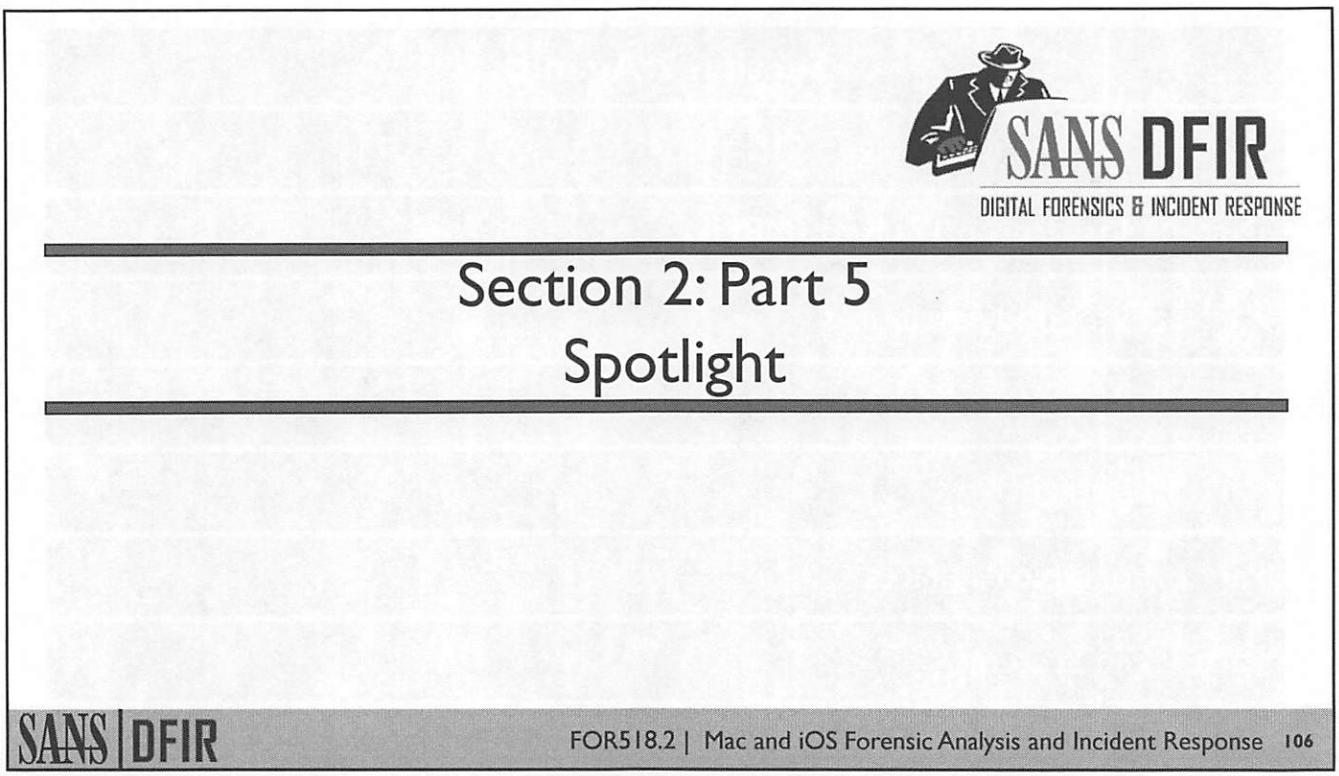
	wd	filename
40	39699407	Users/oompa/Library/Messages/Archive/2016-12-15/Heather Mahalik on 2016-12-15 at 17.47.23.ichat
41	39701691	Users/oompa/Library/Messages/Archive/2016-12-15/Chat with Heather Mahalik et al on 2016-12-15 at 20.10.47.ichat
42	39701654	Users/oompa/Library/Messages/Archive/2016-12-15/Heather Mahalik on 2016-12-15 at 17.47.23.ichat



This page intentionally left blank.



This page intentionally left blank.



This page intentionally left blank.

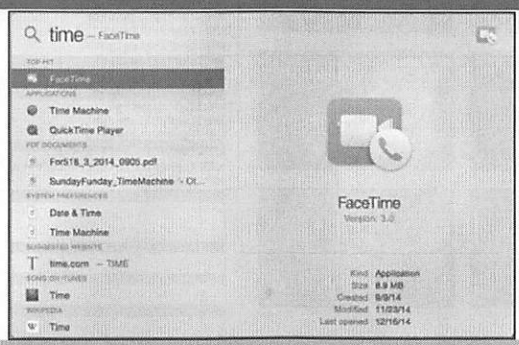
Spotlight

Desktop search system, introduced in 10.4

Indexed file metadata

Similar data found in extended attributes

Writable internal/external media indexed on per-volume basis

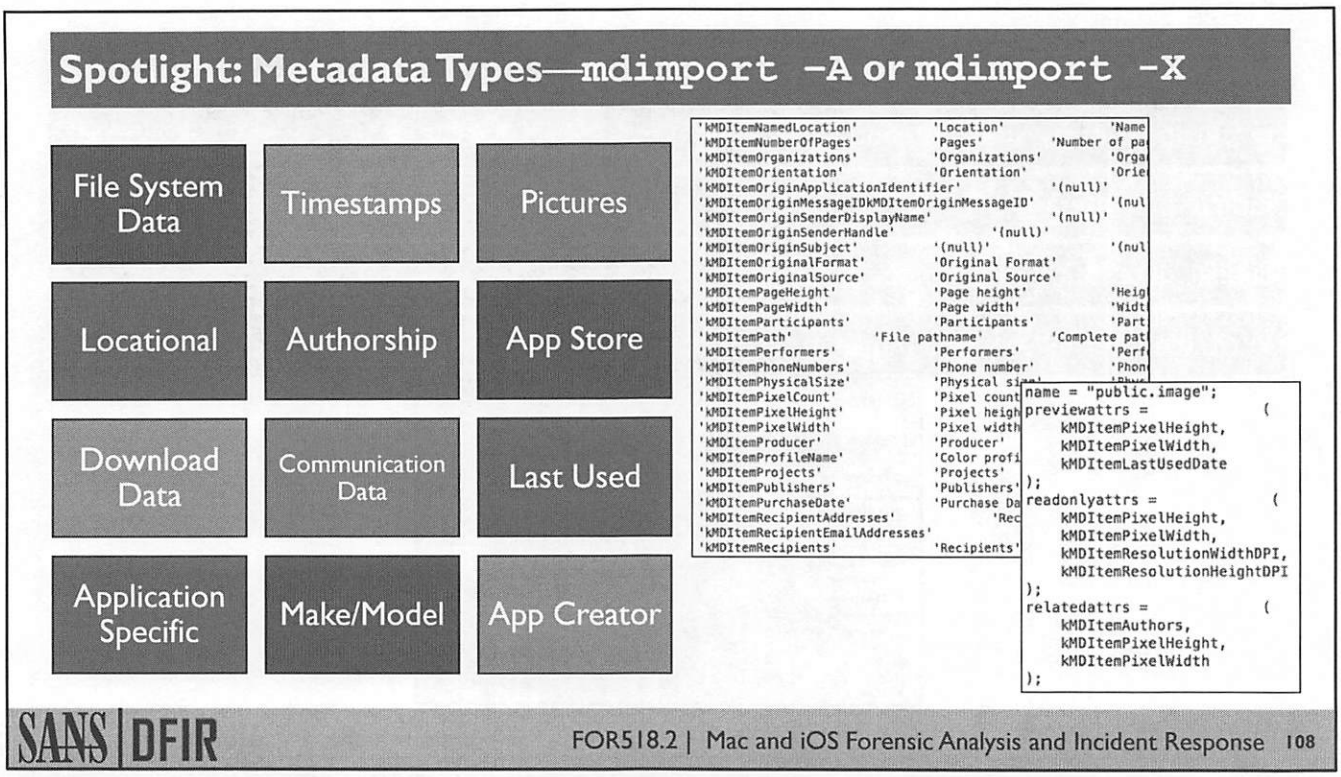


OS X uses Spotlight to index the system to allow the user to search for applications, documents, and other files quickly. This feature was introduced in OS X 10.4 and on iPhones with iOS 3. File metadata, including the extended attributes, are all indexed. On a live system, the Spotlight icon in the top right of the screen can be used to search the Spotlight index. In the example above, a search for "time" was performed. Live search results will appear below in various categories:

- Top Hit: Most likely hit
- Applications
- System Preferences: Items in the preference panels in the system preferences
- Documents
- Folders
- Messages: Contains hits from email
- Events: Contains hits from calendar entries
- Images
- PDF Documents
- Webpages
- Music
- Presentations
- Developer (if installed)
- Look Up: Look up the search term in Dictionary application
- Web Searches: Perform a web search for the search term

While system hard drives and external drives are indexed, other locations may not be indexed by default. These locations include DMG files, CDs and DVDs, hidden files, and system directories. A volume can explicitly be told not to be indexed by placing an empty, hidden file with the filename `.metadata_never_index` at the root of the volume. A volume can be set not to index by using the `mdutil` command.

References:
Mac OS X and iOS Internals: To the Apple's Core by Jonathan Levin—Chapter 2: The User Experience Layer
`mdutil` Man Page



What type of data gets indexed? Anything and everything!

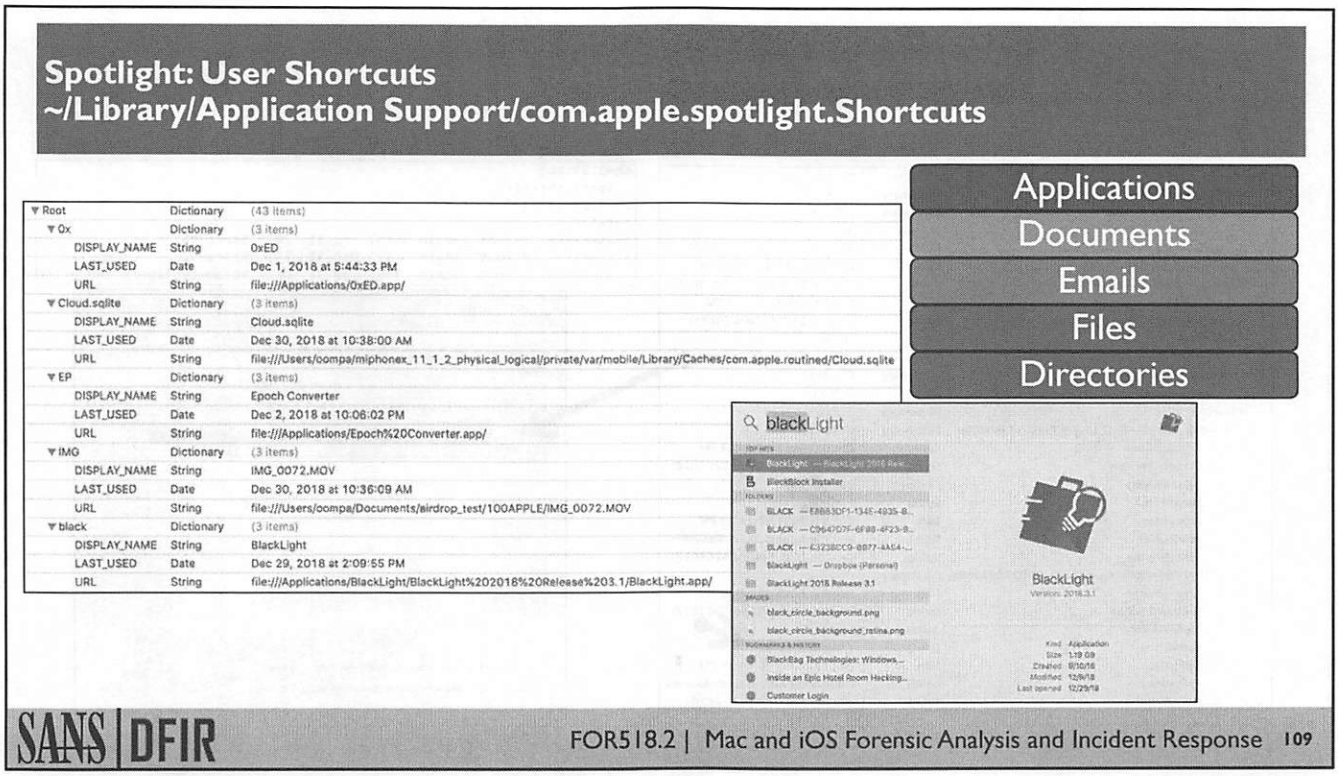
A small example of what might be indexed:

- Another copy of file system metadata, including filenames, logical and physical file sizes, UID/GID, and file system timestamps.
- Timestamps: Along with the file system timestamps, you may also find download dates, last used dates, and date added dates.
- Photos data may include width/length, make and model of hardware that took the photo, aperture, ISO speeds, pixel count, and resolution.
- Authorship information may include who originated the file and what type of application created the document.
- If an application was downloaded or purchased through the Apple App Store, certain receipt data will be indexed.
- Communication information such as what email address sent an attachment or what the hostname was of the phone that AirDropped a photo to the system.
- Depending on what software is installed on the system, certain applications may have their own metadata attributes.

Two native command line utilities can show us what type of data may be indexed. These commands will only show the attributes that are associated with the host system, not the mounted system. Many of the `kMD*` attributes will be the same across systems, but keep an eye out for those application-specific metadata attributes. Newer systems will generally contain more attributes than older systems.

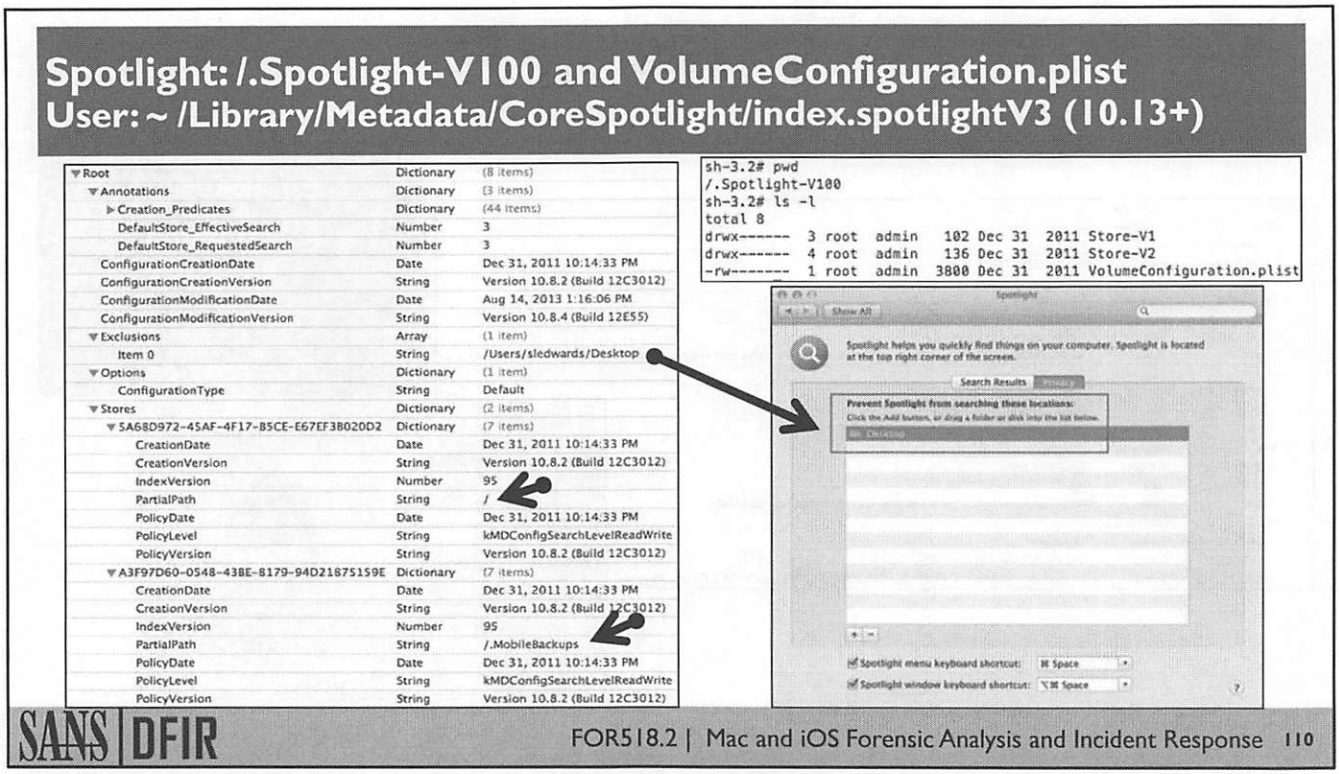
The "`mdimport -A`" command (shown on the left) will print out the attributes' names and descriptions. The "`mdimport -X`" command (shown on the right) prints out metadata attributes associated with a specific file type; for example, attributes associated with photos.

Reference:
Man Page for `mdimport`



Spotlight is an application that is used to index and create a searchable database of the OS X system. Users can search for items using the Spotlight magnifying glass icon in the top right of the GUI, shown above in the screenshot.

These are the first few letters the user typed before selecting an item in the dropdown. If the user types the same few letters again and selects something, the timestamp will be updated; otherwise, it appears that this is a historical listing of Spotlight searches.



The hidden `.Spotlight-V100` directory located in the root of the volume contains the Spotlight Store directories.

`Store-V1` is used with older OS X versions (10.6 and below), while the second version of the store, `Store-V2`, was introduced in 10.7.

You will see both directories on newer versions of OS X, as shown in the screenshot above.

The `.Spotlight-V100` directory contains the property list `VolumeConfiguration.plist`. This property list contains indexing exclusions (among other Spotlight configuration data). In the example above, the user's desktop was excluded from indexing.

The `ConfigurationModificationDate` key contains the timestamp when the Spotlight configuration was last modified.

The two GUIDs in the example show that two volumes are currently being indexed:

- `/` - The root volume, and
- `/.MobileBackups`: a Time Machine-related volume.

The GUIDs reference the Spotlight Stores where the volume indexing database is stored.

Starting in 10.13, each user has their own Spotlight database located in their Library directory. [`http://www.swiftforensics.com/2018/10/the-user-spotlight-database.html`]

```
sh-3.2# pwd
/.Spotlight-V100
sh-3.2# ls -l
total 8
drwx------  3 root  admin   102 Dec 31  2011 Store-V1
drwx------  4 root  admin   136 Dec 31  2011 Store-V2
-rw-------  1 root  admin  3800 Dec 31  2011 VolumeConfiguration.plist
```

▽Root	Dictionary	(8 items)
▽Annotations	Dictionary	(3 items)
▷Creation_Predicates	Dictionary	(44 items)
DefaultStore_EffectiveSearch	Number	3
DefaultStore_RequestedSearch	Number	3
ConfigurationCreationDate	Date	Dec 31, 2011 10:14:33 PM
ConfigurationCreationVersion	String	Version 10.8.2 (Build 12C3012)
ConfigurationModificationDate	Date	Aug 14, 2013 1:16:06 PM
ConfigurationModificationVersion	String	Version 10.8.4 (Build 12E55)
▽Exclusions	Array	(1 item)
Item 0	String	/Users/sledwards/Desktop
▽Options	Dictionary	(1 item)
ConfigurationType	String	Default
▽Stores	Dictionary	(2 items)
▽5A68D972-45AF-4F17-B5CE-E67EF3B020D2	Dictionary	(7 items)
CreationDate	Date	Dec 31, 2011 10:14:33 PM
CreationVersion	String	Version 10.8.2 (Build 12C3012)
IndexVersion	Number	95
PartialPath	String	/
PolicyDate	Date	Dec 31, 2011 10:14:33 PM
PolicyLevel	String	kMDConfigSearchLevelReadWrite
PolicyVersion	String	Version 10.8.2 (Build 12C3012)
▽A3F97D60-0548-43BE-8179-94D21875159E	Dictionary	(7 items)
CreationDate	Date	Dec 31, 2011 10:14:33 PM
CreationVersion	String	Version 10.8.2 (Build 12C3012)
IndexVersion	Number	95
PartialPath	String	/.MobileBackups
PolicyDate	Date	Dec 31, 2011 10:14:33 PM
PolicyLevel	String	kMDConfigSearchLevelReadWrite
PolicyVersion	String	Version 10.8.2 (Build 12C3012)

Spotlight: File Structure: /.Spotlight-V100/Store-V2

```
sh-3.2# pwd
/.Spotlight-V100/Store-V2
sh-3.2# ls -la
total 0
drwx------    4 root  admin   136 Dec 31  2011 .
drwx------    5 root  admin   170 Dec 31  2011 ..
drwx------  121 root  admin  4114 Aug 14 13:15 5A68D972-45AF-4F17-B5CE-E67EF3B020D2
drwx------   70 root  admin  2380 Aug 10 11:36 A3F97D60-0548-438E-8179-94D21875159E
sh-3.2# cd 5A68D972-45AF-4F17-B5CE-E67EF3B020D2/
sh-3.2# ls
.store.db                       live.0.indexIds                 live.2.shadowIndexHead          live.5.indexCompactDirectory
0.directoryStoreFile            live.0.indexPositions           live.3.directoryStoreFile       live.5.indexDirectory
0.directoryStoreFile.shadow     live.0.indexPostings            live.3.directoryStoreFile.shadow live.5.indexGroups
0.indexArrays                   live.0.indexUpdates             live.3.indexArrays              live.5.indexHead
0.indexCompactDirectory         live.0.shadowIndexGroups        live.3.indexCompactDirectory    live.5.indexIds
0.indexDirectory                live.0.shadowIndexHead          live.3.indexDirectory           live.5.indexPositionTable
0.indexGroups                   live.1.directoryStoreFile       live.3.indexGroups              live.5.indexPositions
0.indexHead                     live.1.directoryStoreFile.shadow live.3.indexHead               live.5.indexPostings
0.indexIds                      live.1.indexArrays              live.3.indexIds                 live.5.indexTermIds
0.indexPositions                live.1.indexCompactDirectory    live.3.indexPositions           live.5.indexUpdates
0.indexPostings                 live.1.indexDirectory           live.3.indexPostings            live.5.shadowIndexArrays
0.indexUpdates                  live.1.indexGroups              live.3.indexUpdates             live.5.shadowIndexCompactDirectory
0.shadowIndexGroups             live.1.indexHead                live.3.shadowIndexGroups        live.5.shadowIndexDirectory
0.shadowIndexHead               live.1.indexIds                 live.3.shadowIndexHead          live.5.shadowIndexGroups
Cache                           live.1.indexPositions           live.4.directoryStoreFile       live.5.shadowIndexHead
Lion.created                    live.1.indexPostings            live.4.directoryStoreFile.shadow live.5.shadowIndexPositionTable
Lion.modified                   live.1.indexUpdates             live.4.indexArrays              live.5.shadowIndexTermIds
indexState                      live.1.shadowIndexGroups        live.4.indexCompactDirectory    permStore
journalAttr.1995                live.1.shadowIndexHead          live.4.indexDirectory           reverseDirectoryStore
journalExclusion                live.2.directoryStoreFile       live.4.indexGroups              reverseDirectoryStore.shadow
journals.live                   live.2.directoryStoreFile.shadow live.4.indexHead               reverseStore.updates
journals.repair                 live.2.indexArrays              live.4.indexIds                 shutdown_time
journals.scan                   live.2.indexCompactDirectory    live.4.indexPositions           store.db
live.0.directoryStoreFile       live.2.indexDirectory           live.4.indexPostings            store.updates
live.0.directoryStoreFile.shadow live.2.indexGroups             live.4.indexUpdates             store_generation
live.0.indexArrays              live.2.indexHead                live.4.shadowIndexGroups        tmp.Lion
live.0.indexCompactDirectory    live.2.indexIds                 live.4.shadowIndexHead          tmp.SnowLeopard
live.0.indexDirectory           live.2.indexPositions           live.5.directoryStoreFile       tmp.spotlight.loc
live.0.indexGroups              live.2.indexPostings            live.5.directoryStoreFile.shadow tmp.spotlight.state
live.0.indexHead                live.2.shadowIndexGroups        live.5.indexArrays
```


Each Spotlight Store directory contains many files.

The example above is from a Mountain Lion OS X system. These files are the indexing databases used by Spotlight. The format for most of these files is unknown at this time.

The two places where data is readily available for investigators are the `Cache` directory and the `store.db` index database.

```
sh-3.2# pwd
/.Spotlight-V100/Store-V2
sh-3.2# ls -la
total 0
drwx------    4 root  admin   136 Dec 31  2011 .
drwx------    5 root  admin   170 Dec 31  2011 ..
drwx------  121 root  admin  4114 Aug 14 13:15 5A68D972-45AF-4F17-B5CE-E67EF3B020D2
drwx------   70 root  admin  2380 Aug 10 11:36 A3F97D60-0548-43BE-8179-94D21875159E
sh-3.2# cd 5A68D972-45AF-4F17-B5CE-E67EF3B020D2/
sh-3.2# ls
.store.db                              live.0.indexIds                        live.2.shadowIndexHead                 live.5.indexCompactDirectory
0.directoryStoreFile                   live.0.indexPositions                  live.3.directoryStoreFile              live.5.indexDirectory
0.directoryStoreFile.shadow            live.0.indexPostings                   live.3.directoryStoreFile.shadow       live.5.indexGroups
0.indexArrays                          live.0.indexUpdates                    live.3.indexArrays                     live.5.indexHead
0.indexCompactDirectory                live.0.shadowIndexGroups               live.3.indexCompactDirectory           live.5.indexIds
0.indexDirectory                       live.0.shadowIndexHead                 live.3.indexDirectory                  live.5.indexPositionTable
0.indexGroups                          live.1.directoryStoreFile              live.3.indexGroups                     live.5.indexPositions
0.indexHead                            live.1.directoryStoreFile.shadow       live.3.indexHead                       live.5.indexPostings
0.indexIds                             live.1.indexArrays                     live.3.indexIds                        live.5.indexTermIds
0.indexPositions                       live.1.indexCompactDirectory           live.3.indexPositions                  live.5.indexUpdates
0.indexPostings                        live.1.indexDirectory                  live.3.indexPostings                   live.5.shadowIndexArrays
0.indexUpdates                         live.1.indexGroups                     live.3.indexUpdates                    live.5.shadowIndexCompactDirectory
0.shadowIndexGroups                    live.1.indexHead                       live.3.shadowIndexGroups               live.5.shadowIndexDirectory
0.shadowIndexHead                      live.1.indexIds                        live.3.shadowIndexHead                 live.5.shadowIndexGroups
Cache                                  live.1.indexPositions                  live.4.directoryStoreFile              live.5.shadowIndexHead
Lion.created                           live.1.indexPostings                   live.4.directoryStoreFile.shadow       live.5.shadowIndexPositionTable
Lion.modified                          live.1.indexUpdates                    live.4.indexArrays                     live.5.shadowIndexTermIds
indexState                             live.1.shadowIndexGroups               live.4.indexCompactDirectory           permStore
journalAttr.1995                       live.1.shadowIndexHead                 live.4.indexDirectory                  reverseDirectoryStore
journalExclusion                       live.2.directoryStoreFile              live.4.indexGroups                     reverseDirectoryStore.shadow
journals.live                          live.2.directoryStoreFile.shadow       live.4.indexHead                       reverseStore.updates
journals.repair                        live.2.indexArrays                     live.4.indexIds                        shutdown_time
journals.scan                          live.2.indexCompactDirectory           live.4.indexPositions                  store.db
live.0.directoryStoreFile              live.2.indexDirectory                  live.4.indexPostings                   store.updates
live.0.directoryStoreFile.shadow       live.2.indexGroups                     live.4.indexUpdates                    store_generation
live.0.indexArrays                     live.2.indexHead                       live.4.shadowIndexGroups               tmp.Lion
live.0.indexCompactDirectory           live.2.indexIds                        live.4.shadowIndexHead                 tmp.SnowLeopard
live.0.indexDirectory                  live.2.indexPositions                  live.5.directoryStoreFile              tmp.spotlight.loc
live.0.indexGroups                     live.2.indexPostings                   live.5.directoryStoreFile.shadow       tmp.spotlight.state
live.0.indexHead                       live.2.shadowIndexGroups               live.5.indexArrays
```

Spotlight: File Structure—Spotlight Cache Directory

/.Spotlight-V100/Store-V2/<GUID>/Cache/

- Nested directory structure
- Numerous text files
- Filename = CNID (inode) number
- 10.13: Does not appear to be used similarly

```
sh-3.2# tree -L 4 . | more
.
└── 0000
    └── 0000
        ├── 0000
        │   ├── 2304.txt
        │   ├── 2898.txt
        │   ├── 2899.txt
        │   └── 2904.txt
        ├── 0001
        │   ├── 102024.txt
        │   ├── 102447.txt
        │   ├── 102448.txt
        │   ├── 102449.txt
        │   ├── 102450.txt
        │   ├── 102451.txt
        │   ├── 102452.txt
        │   ├── 102453.txt
        │   ├── 102454.txt
        │   ├── 102455.txt
        │   ├── 102456.txt
        │   └── 102457.txt
```

```
word:0005 oompa$ pwd
/Volumes/dade_mounted/.Spotlight-V100/Store-V2/0A00840B-5B8C-44DC-9337-B6B58F5D5607/
Cache/0000/0000/0005
word:0005 oompa$ cat 375242.txt
Dade Murphy <z3r0cool95@gmail.com> Kate Libby <katelibby11@gmail.com>  Heading west
In the airport getting some breakfast, I'll see you in a few days!
word:0005 oompa$ sudo find /Volumes/dade_mounted/ -inum 375242
/Volumes/dade_mounted//Users/zerocool/Library/Mail/V2/IMAP-z3r0cool95@imap.gmail.com
/[Gmail].mbox/Sent Mail.mbox/21C7CD98-9DCB-4CE5-B19E-931FBE825451/Data/Messages/198.
emlx
```


One directory that can be of use to forensic analysts is the Spotlight Cache directory. This directory contains a nesting of other subdirectories containing many text files. These text files hold text-based versions of their original documents, including emails, documents, and chats.

The number preceding each `.txt` extension is the CNID/inode number for the file it represents. The lower screenshot shows an example of this. The filename `375242.txt` contains the email-related data.

We can find the associated file by searching the system for the CNID, `375242`. We can use the `find` command on the mounted volume. Use the `-inum` argument to find the file by CNID/inode number.

```
sudo find /Volumes/dade_mounted/ -inum 375242
```

Experimentation shows these text files will no longer be available when a file is deleted or simply moved to the trash.

Spotlight: Find by Metadata Key—mdfind

Use on native disk or on a mounted image

```
mdfind "kMDItemLongitude == * "
```

```
word:/ oompa$ mdfind "kMDItemLongitude == * " -onlyin /Volumes/dade_mounted/
/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2013/12/15/20131215-202049/wYL6+GwO
QPqKDYPXqhayMA/IMG_0004.jpg
/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2013/12/15/20131215-202034/Royh3GNQ
QIOOan5qlKnC%A/IMG_0004.jpg
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/sub/0139451307dd27b8865a4
ba7924387874893ca2223/IMG_0004.JPG
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/sub-shared/2CC4BCB7-B178-
4551-8A7E-D6BFF2C420D6/IMG_0002.JPG
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/sub-shared/1FB94DBF-808F-
4642-887C-2CB0B48FD712/IMG_0004.JPG
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/sub-shared/F602AFF0-A250-
465B-818B-0463F6935E2B/IMG_0001.JPG
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/sub-shared/C7C4AD09-059A-
4EDE-8A2E-4E99E2EDC25B/IMG_0003.JPG
/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Masters/2013/12/15/20131215-202034/IMG_0004.
JPG
```


The "mdfind" command will find files based on certain metadata criteria.

In the example, the attribute "kMDItemLongitude" was searched for. This will print out the filenames and paths for all files on the system (or mounted image) where this attribute was indexed. This example will show us files that contain locational data.

Reference:
Man Page for mdfind

Spotlight: Useful Metadata Searches—Location

- `mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemLatitude == *"`

```
word:dade_mounted oompa$ mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemLatitude == *"
/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2013/12/15
/20131215-202049/wYL6+GwOQPqKDYPXqhayMA/IMG_0004.jpg
/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2013/12/15
/20131215-202034/Royh3GNQQIOOan5qlKnC%A/IMG_0004.jpg
/Volumes/dade_mounted/Users/zerocool/Library/Application Support/iLifeAssetManagement/assets/
sub/0139451307dd27b8865a4ba7924387874893ca2223/IMG_0004.JPG
```

```
kMDItemGPSDateStamp          = "2013:12:15"
kMDItemHasAlphaChannel       = 0
kMDItemImageDirection        = 313.0193548387097
kMDItemIsApplicationManaged  = 1
kMDItemISOSpeed              = 800
kMDItemKind                  = "JPEG image"
kMDItemLatitude              = 38.94850833333334
kMDItemLogicalSize           = 1754408
kMDItemLongitude             = -77.33985
kMDItemOrientation           = 0
kMDItemPhysicalSize          = 1757184
kMDItemPixelCount            = 7990272
```


The `mdfind` command will find files based on certain metadata criteria.

In the example, the attribute "`kMDItemLatitude`" was searched for. This will print out the filenames and paths for all files on the system (or mounted image) where this attribute was indexed. This example will show us files that contain locational data.

This example uses the `mdfind` command with the `-onlyin` argument to limit the search in the mounted image directory to search for files that have the `kMDItemLatitude` metadata attribute.

This will print out all files that contain locational attributes on the mounted image.

The bottom screenshot shows the output `mdls (covered soon)` on of one of the files with the coordinates that were indexed from this photo.

References:
Man Page for `mdfind`
Man Page for `mdls`

Spotlight: List Metadata—mdls

```
word:/ oompa$ mdls "/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2
013/12/15/20131215-202049/wYL6+GwOQPqKDYPXqhayMA/IMG_0004.jpg"
kMDItemAcquisitionMake          = "Apple"
kMDItemAcquisitionModel         = "iPhone 4S"
kMDItemAltitude                 = 74
kMDItemAperture                 = 2.52606882168926
kMDItemBitsPerSample            = 32
kMDItemColorSpace               = "RGB"
kMDItemContentCreationDate      = 2013-12-16 01:20:47 +0000
kMDItemContentModificationDate = 2013-12-16 01:20:47 +0000
kMDItemContentType              = "public.jpeg"
kMDItemContentTypeTree          = (
    "public.jpeg",
    "public.image",
    "public.data",
    "public.item",
    "public.content"
)
kMDItemCreator                  = "QuickTime 7.7.1"
kMDItemDateAdded                = 2013-12-16 01:20:49 +0000
kMDItemDisplayName              = "IMG_0004.jpg"
kMDItemEXIFVersion              = "2.2"
kMDItemExposureMode             = 0
kMDItemExposureProgram          = 2
kMDItemExposureTimeSeconds      = 0.06666666666666667
```


The `mdls` command can be used to list the metadata associated with a particular file from the Spotlight databases. In the screenshot above, we can see the following attributes of the file:

- Timestamps
- Content Types
- Download Date
- Download Location
- File Sizes
- File Ownership
- File Properties

This screenshot is only a partial output of the metadata associated with this file. The complete output of this command is shown on the next page.

Reference:
Man Page for `mdls`

```
word:/ oompa$ mdls "/Volumes/dade_mounted/Users/zerocool/Pictures/iPhoto Library.photolibrary/Previews/2
013/12/15/20131215-202049/wYL6+GwOQPqKDYPXqhayMA/IMG_0004.jpg"
kMDItemAcquisitionMake          = "Apple"
kMDItemAcquisitionModel         = "iPhone 4S"
kMDItemAltitude                 = 74
kMDItemAperture                 = 2.52606882168926
kMDItemBitsPerSample            = 32
kMDItemColorSpace               = "RGB"
kMDItemContentCreationDate      = 2013-12-16 01:20:47 +0000
kMDItemContentModificationDate = 2013-12-16 01:20:47 +0000
kMDItemContentType              = "public.jpeg"
kMDItemContentTypeTree          = (
    "public.jpeg",
    "public.image",
    "public.data",
    "public.item",
    "public.content"
)
kMDItemCreator                  = "QuickTime 7.7.1"
kMDItemDateAdded                = 2013-12-16 01:20:49 +0000
kMDItemDisplayName              = "IMG_0004.jpg"
kMDItemEXIFVersion              = "2.2"
kMDItemExposureMode             = 0
kMDItemExposureProgram          = 2
kMDItemExposureTimeSeconds      = 0.06666666666666667
kMDItemFlashOnOff               = 0
kMDItemFNumber                  = 2.4
kMDItemFocalLength              = 4.28
kMDItemFSContentChangeDate      = (null)
kMDItemFSCreationDate           = (null)
kMDItemFSCreatorCode            = ""
kMDItemFSFinderFlags            = (null)
kMDItemFSHasCustomIcon          = (null)
kMDItemFSInvisible              = 0
kMDItemFSIsExtensionHidden      = (null)
kMDItemFSIsStationery           = (null)
kMDItemFSLabel                  = 0
kMDItemFSName                   = (null)
kMDItemFSNodeCount              = (null)
kMDItemFSOwnerGroupID           = (null)
kMDItemFSOwnerUserID            = (null)
kMDItemFSSize                   = 1754408
kMDItemFSTypeCode               = ""
kMDItemGPSDateStamp             = "2013:12:15"
kMDItemHasAlphaChannel          = 0
kMDItemImageDirection           = 313.0193548387097
kMDItemIsApplicationManaged     = 1
kMDItemISOSpeed                 = 800
kMDItemKind                     = "JPEG image"
kMDItemLatitude                 = 38.94850833333334
kMDItemLogicalSize              = 1754408
kMDItemLongitude                = -77.33985
kMDItemOrientation              = 0
kMDItemPhysicalSize             = 1757184
kMDItemPixelCount               = 7990272
kMDItemPixelHeight              = 2448
kMDItemPixelWidth               = 3264
kMDItemProfileName              = "sRGB IEC61966-2.1"
kMDItemRedEyeOnOff              = 0
kMDItemResolutionHeightDPI      = 72
kMDItemResolutionWidthDPI       = 72
kMDItemSupportFileType          = (
    MDSystemFile
)
kMDItemTimestamp                = "00:59:06"
kMDItemWhiteBalance             = 0
```

Spotlight: Useful Metadata Searches—Application Usage

- `find /Applications/ -iname '*.app' -exec echo {} \; -exec mdls -name` **`kMDItemUseCount`** `-name` **`kMDItemLastUsedDate`** `-name` **`kMDItemUsedDates`** `{} \;`

Can also use on a mounted image (i.e., /Volumes/galaga_mounted)

```
/Applications//DB Browser for SQLite.app
kMDItemLastUsedDate = 2016-12-24 15:59:23 +0000
kMDItemUseCount     = 9
kMDItemUsedDates    = (
    "2016-12-07 05:00:00 +0000",
    "2016-12-14 05:00:00 +0000",
    "2016-12-15 05:00:00 +0000",
    "2016-12-16 05:00:00 +0000",
    "2016-12-24 05:00:00 +0000"
)
/Applications//Dictionary.app
kMDItemLastUsedDate = 2016-12-23 14:42:02 +0000
kMDItemUseCount     = 2
kMDItemUsedDates    = (
    "2016-12-11 05:00:00 +0000",
    "2016-12-23 05:00:00 +0000"
)
/Applications//Dropbox.app
kMDItemLastUsedDate = 2016-12-26 04:15:18 +0000
kMDItemUseCount     = 2
kMDItemUsedDates    = (
    "2016-12-18 05:00:00 +0000",
    "2016-12-25 05:00:00 +0000"
)
```


This command prints the name of an application along with the "Last Used Date", "Use Count", and "Item Used Dates" associated with it. This can give a good idea of when an application was used. This is very similar to that of Windows Prefetch data.

One caveat to the `kMDItemUsedDates` is that these are only created once in a given day. Another attribute, `kMDItemUseCount`, will keep track of the number of times an application (or file) was used; however, I have found in testing that just a simple file open may show up once, twice, or three times in the `kMDItemUseCount` tally! This is due to multiple processes that open the files. I would advise that this number may be unreliable and should only be used to get an idea of whether a file was opened and generally how often.

The `kMDItemUsedDates` timestamps show the day as well as the time zone of the application usage, while the `kMDItemLastUsedDate` shows the specific time that a user last opened that file.

For example, DB Browser for SQLite.app was used nine times in five days (all in UTC -5:00 time zone), with the last used time being on 12/24/2016.

The command uses the `find` command in the `/Applications/` directory to search for files with the .app extension (`-iname`) and executes the following command (`-exec`). You can also use this on a mounted image (`/Volumes/dade_mounted`) to search that system's Spotlight database instead of your own.

```
echo {} \; -exec mdls -name kMDItemUsedDates {}
```

The first part of the command (`echo`) "prints" the file path and name of the file found with the `find` command. (The curly brackets `{}` are used as a variable to store the filename/path.) The second part (`-exec`) prints the metadata output for only the `kMDItemUsedDates`.

Spotlight: Useful Metadata Searches—Downloaded Files

- `mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemDownloadedDate == *"`
- `mdls -name kMDItemDisplayName -name kMDItemDownloadedDate Firefox\ 26.0.dmg`

```
word:Downloads oompa$ mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemDownloadedDate == *"
/Volumes/dade_mounted/Users/zerocool/Downloads/pages.pdf
/Volumes/dade_mounted/Users/zerocool/Downloads/Firefox 26.0.dmg
/Volumes/dade_mounted/Users/zerocool/Pictures/url.html
/Volumes/dade_mounted/Users/zerocool/Downloads/url.html
/Volumes/dade_mounted/Users/zerocool/Downloads/bitcoin-0.8.6-macosx.dmg
/Volumes/dade_mounted/Users/zerocool/Downloads/AdobeFlashPlayerInstaller_11_ltrosxd_aaa_aih.dmg
/Volumes/dade_mounted/Users/zerocool/Downloads/python-3.3.2-macosx10.6.dmg
/Volumes/dade_mounted/Users/zerocool/Downloads/python-2.7.6.msi
word:Downloads oompa$ mdls -name kMDItemDisplayName -name kMDItemDownloadedDate Firefox\ 26.0.dmg
kMDItemDisplayName     = "Firefox 26.0.dmg"
kMDItemDownloadedDate = (
    "2013-12-15 03:01:01 +0000"
)
```

Used in Safari, Messages, and AirDrop

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 120

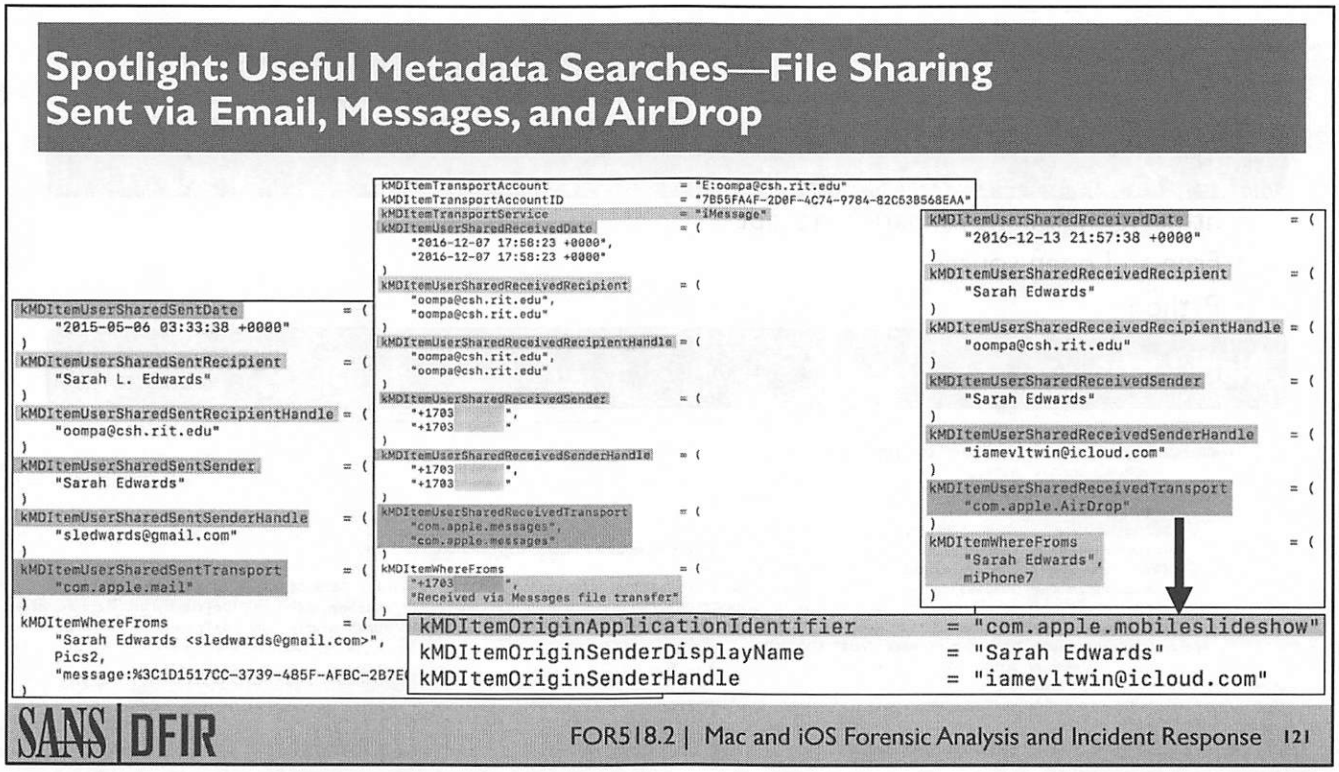
We can search for downloaded items by using the `kMDItemDownloadedDate` metadata attribute with the `mdfind` utility.

```
mdfind -onlyin /Volumes/dade_mounted/ -name "kMDItemDownloadedDate == *"
```

Using the `mdls` command on one of these files will show the filename and when the item was downloaded.

```
mdls -name kMDItemDisplayName -name kMDItemDownloadedDate Firefox\ 26.0.dmg
```

It is worth noting that this search does not show everything that was ever downloaded. It will only show items from applications that use this metadata attribute, such as Safari, Messages, and AirDrop.



On newer operating systems, there are additional sharing metadata attributes that can be forensically very interesting and useful. Using `mdfind` to query files that contain the `kMDItemUserSharedSentDate` attribute, we can find files that have been shared out. The screenshot on the left shows an example of a file sent via Apple Mail (com.apple.mail) on 05/06/2015. This email had the subject "Pics2" and was sent from sledwards@gmail.com to oompa@csh.rit.edu.

On the flip side, we can also use `mdfind` to query for files that have been received by the system using "`kMDItemUserSharedReceivedDate`". The example in the middle shows a file received via the Messages application (com.apple.messages) from a phone number "+1703 …" to oompa@csh.rit.edu on 12/07/2016. This file was received using the iMessage protocol in the Messages application.

Files received via the AirDrop service can be hard to discern without digging a bit further. These newer sharing metadata attributes can make this easier to determine. The screenshot on the right shows a file was received on 12/13/2016 from iamevltwin@icloud.com to oompa@csh.rit.edu using AirDrop (com.apple.AirDrop). The `kMDItemWhereFroms` attribute gives us the username and device name that were used to AirDrop from. In this case, it was Sarah's "miPhone7". The bottom right screenshot shows that it was AirDropped from the com.apple.mobileslideshow application on Sarah's miPhone7. This is the Photos application on the iPhone.

Offline Spotlight Parsing

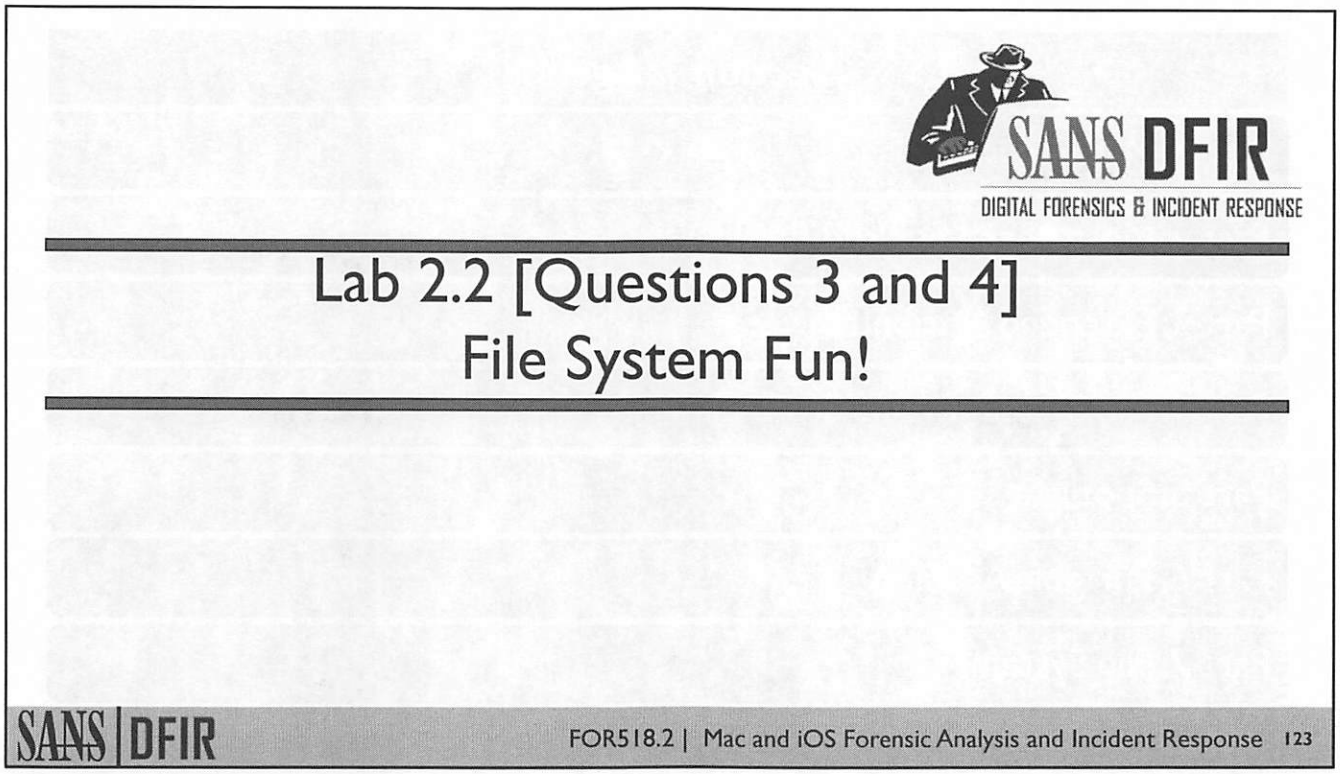
mac_apt

- https://github.com/ydkhatri/mac_apt
- Free and open source
- Python

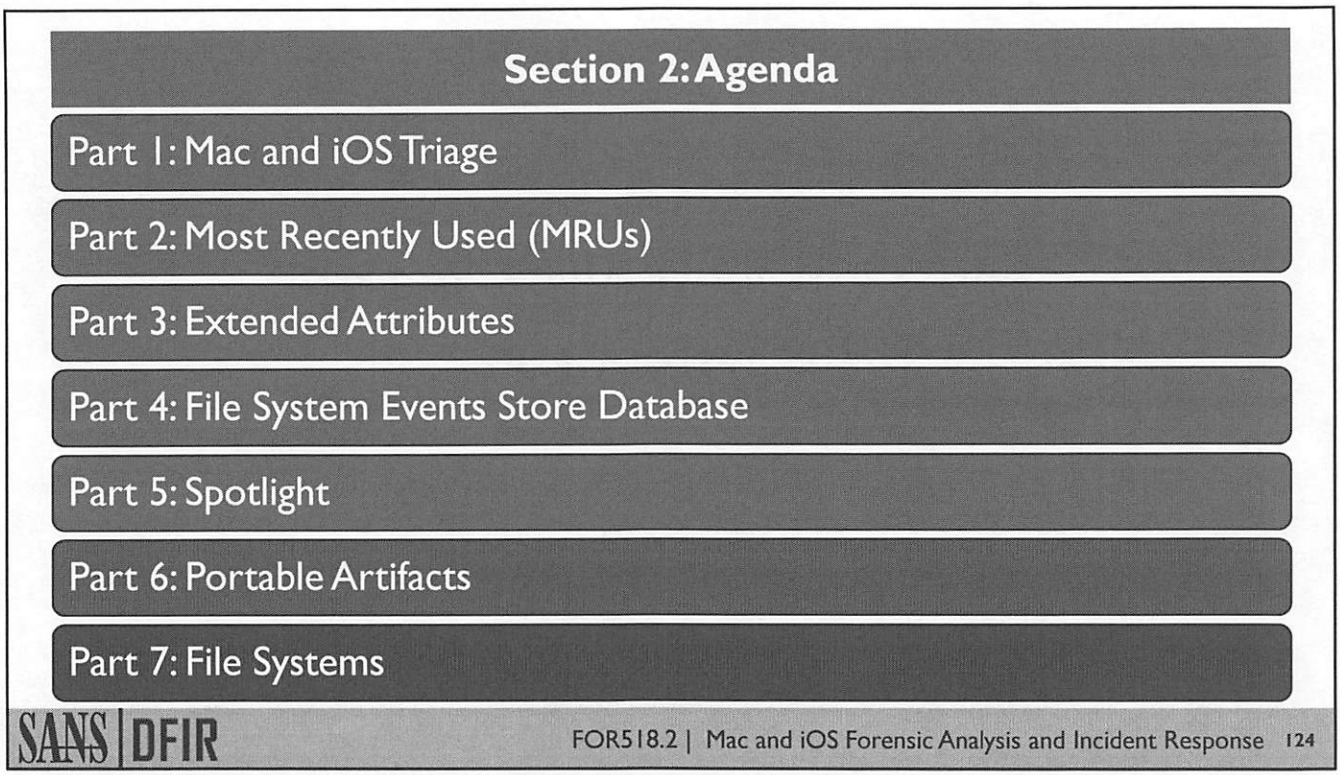
BlackLight

```
                kMDItemLastUsedDate: 2018-02-25 19:57:06Z (???) (???) (???)
          kMDItemLastUsedDate_Ranking: 2018-02-25 00:00:00Z (???) (???) (???)
               kMDItemLogicalSize: 55358955
              kMDItemPhysicalSize: 55361536
                  kMDItemUseCount: 1
              kMDItemUsedDates[0]: 2018-02-25 05:00:00Z (???) (???) (???)
             kMDItemWhereFroms[0]: https://download-installer.cdn.mozilla.net/pub/firefox/releases/58.0.2/mac/en-US/Firefox%2058.0.2.dmg
             kMDItemWhereFroms[1]: https://www.mozilla.org/en-US/firefox/new/?scene=2
                com.apple.quarantine: 303038333b35613933313531323b5361666172693b43334534463435432d453235312d343544332d393346422d4244304336324430393433332
com.apple.metadata:kMDItemWhereFroms: <?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"><plist vers
com.apple.metadata:kMDItemDownloadedDate: <?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"><plist vers
com.apple.lastuseddate#PS: 1215935a0000000027cb472500000000
                    DateAdded: 2018-02-25 19:57:06
                    BSD Flags: 0
```


There are a few tools that have the capability to parse the store.db files offline (versus mounting and live queries). The first is the open-source mac_apt utility that is a multiplatform Python script, while the second is BlackLight (using the Advanced Parsing).



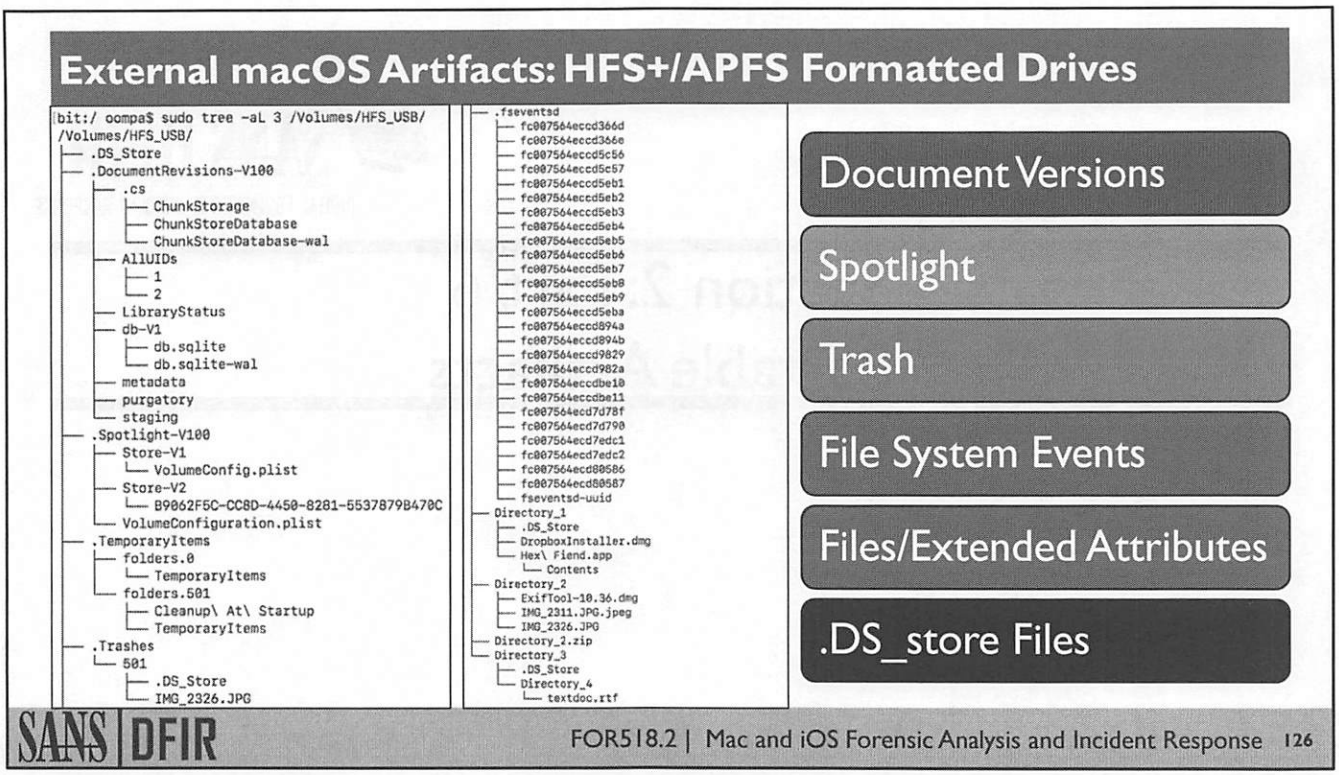
This page intentionally left blank.



This page intentionally left blank.

Section 2: Part 6
Portable Artifacts

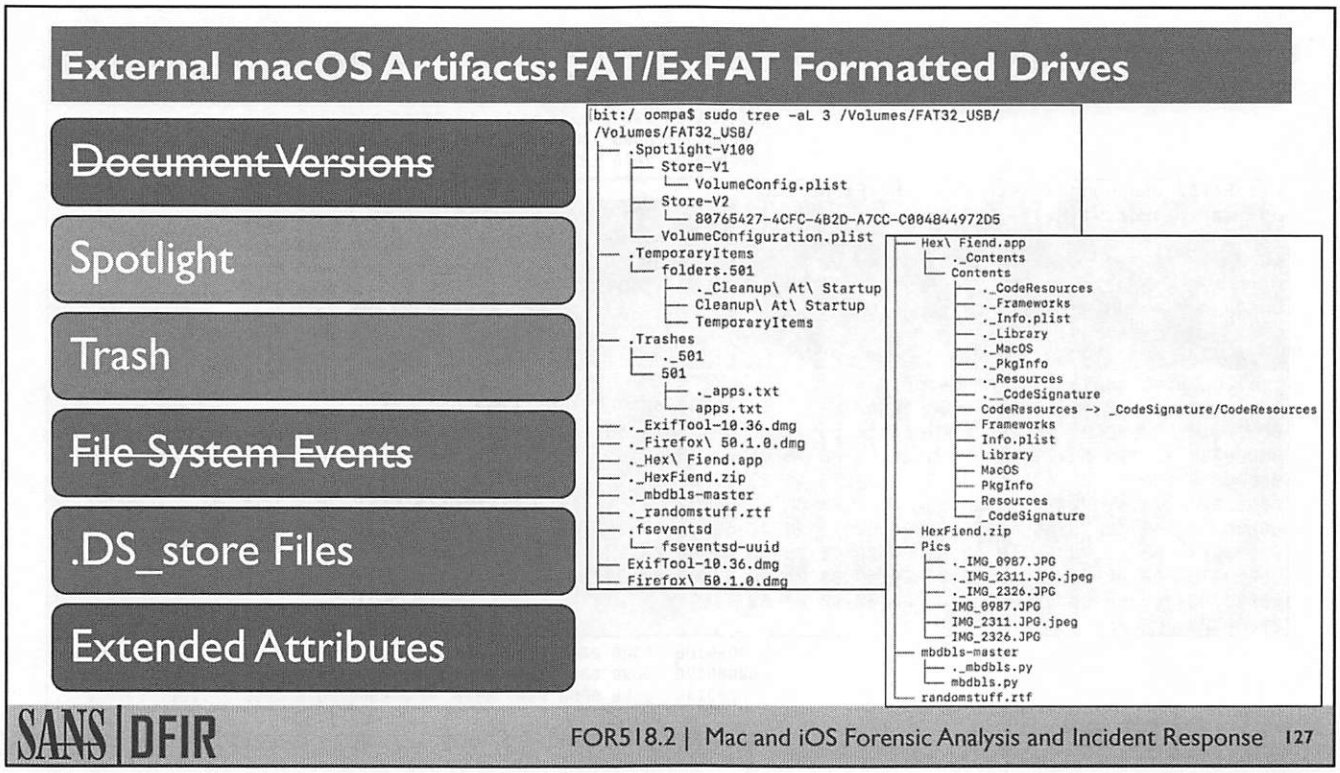
This page intentionally left blank.



macOS can leave behind some interesting artifacts that may be shown on other non-Mac systems. These artifacts can help us determine if a file was originally found on a Mac system. These artifacts may include various directories, extended attributes, and those sneaky `.DS_store` files that seem to get everywhere!

Files copied on an HFS+ formatted drive will show signs of Mac origination, including the following:

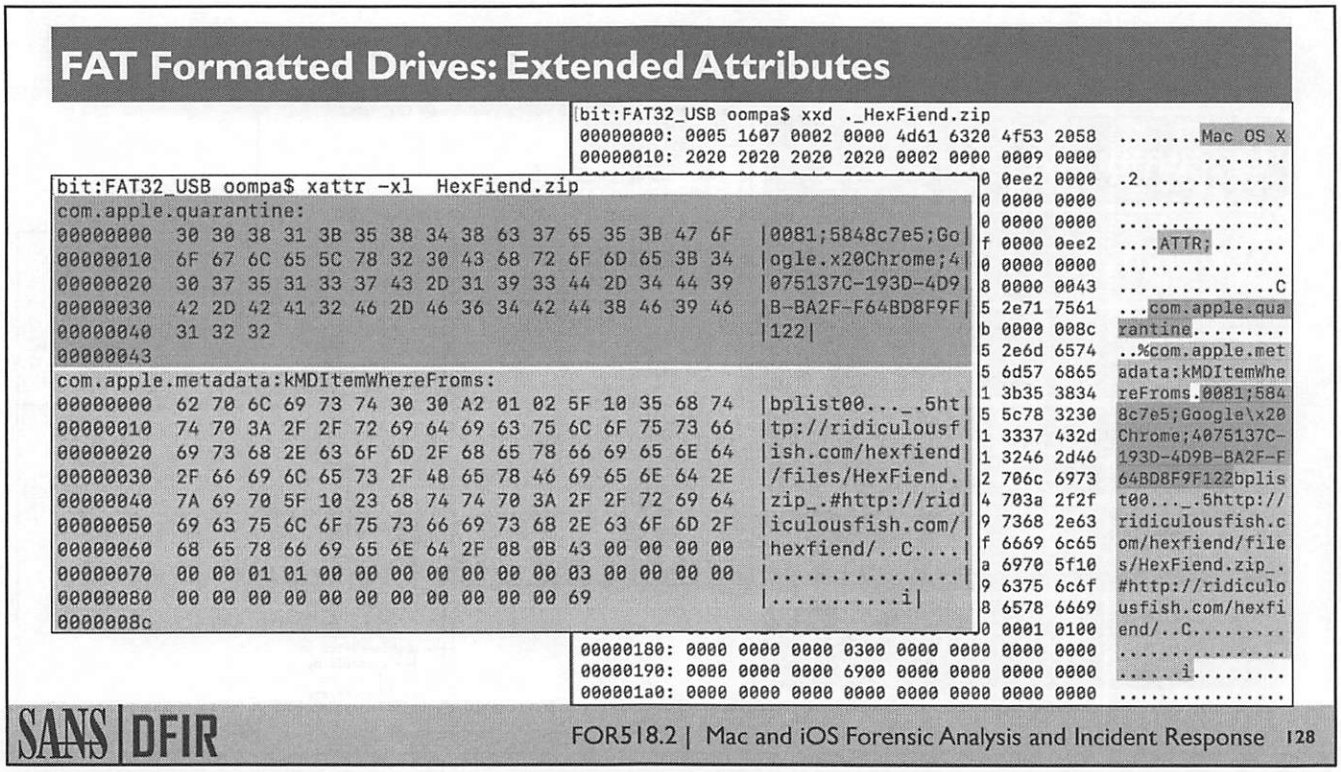
- Extended Attributes
- Spotlight Index
- Document Versions
- Trashes Directory
- `.fseventsd` Directory
- `.DS_store` Files



Files copied to FAT or ExFAT formatted drives will show signs of OS X originations, including a Spotlight directory (with `store.db` Spotlight index!) and a `.Trashes` directory if files were "removed".

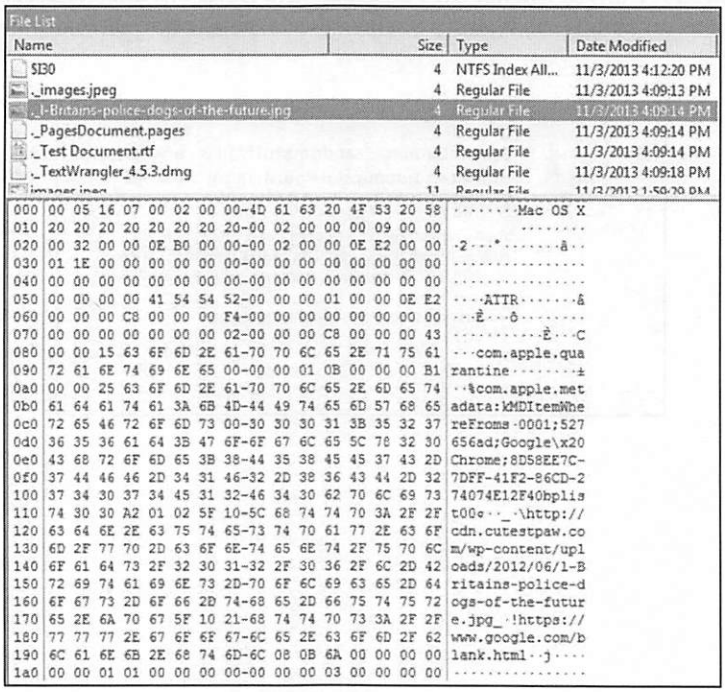
Extended attribute information is also relocated along with each file; however, FAT-based file systems have no support for them, and therefore the data needs to be stored differently. Each file that has extended attributes will have an additional file beginning with a "`._`" that matches the name of the file. This contains the extended attribute information, as shown in the next slide.

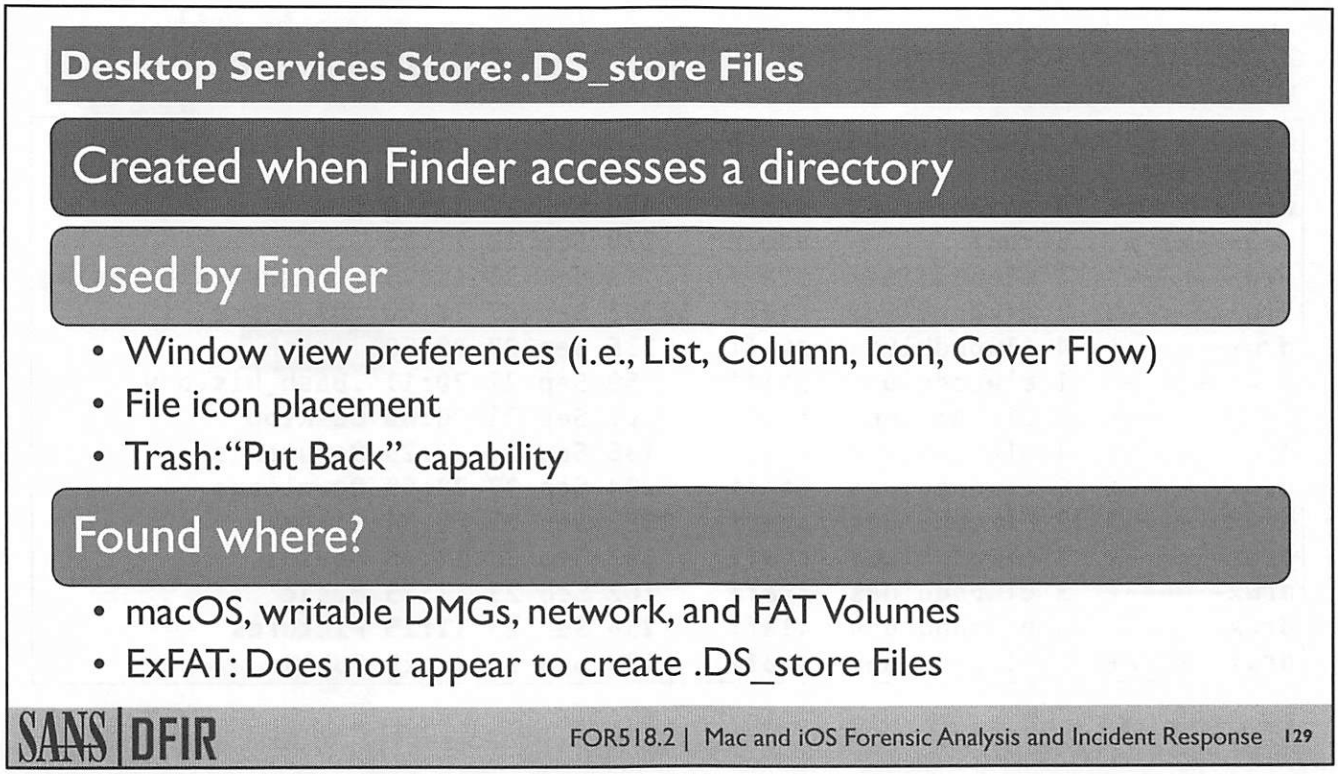
Unfortunately, FAT and ExFAT drives will not have the Document Revisions directories, as these volumes do not support Document Versions capability. Along the same line, the `.fseventsd` directory will exist, but no files appear to be written into it.



Extended attributes that are normally stored in the Attributes File on HFS+ file systems are stored in separate files on FAT32 and ExFAT file systems. These files share the same filename as the associated file with a "dot underscore" (`._`) appended to the beginning. These files are 4,096 bytes in size.

Shown in the larger screenshot are the file signatures "`Mac OS X`" and "`ATTR`", followed by the contents of the extended attribute. At the end of the file is the text "`This resource fork intentionally left blank`" (not shown in screenshot). The format of these "`._`" files contains the attribute names first, followed by the data. The inset screenshot shows what the data will look like if printed with the "`xattr`" command.





Everyone has seen those pervasive, almost annoying `.DS_store` files—they seem to be everywhere!

These files are created when the Finder application is used to view a directory. While all over OS X drives, these files are copied along with files to external hard drives, thumb drives, or network drives. These files implement a B-tree format and are used to save the Finder viewing settings. These are settings such as the column used to sort by, icon placement, or perhaps of most interest, the Trash "Put Back" capability.

These files can be written on many types of volumes; however, they do not appear to get written to ExFAT volumes.

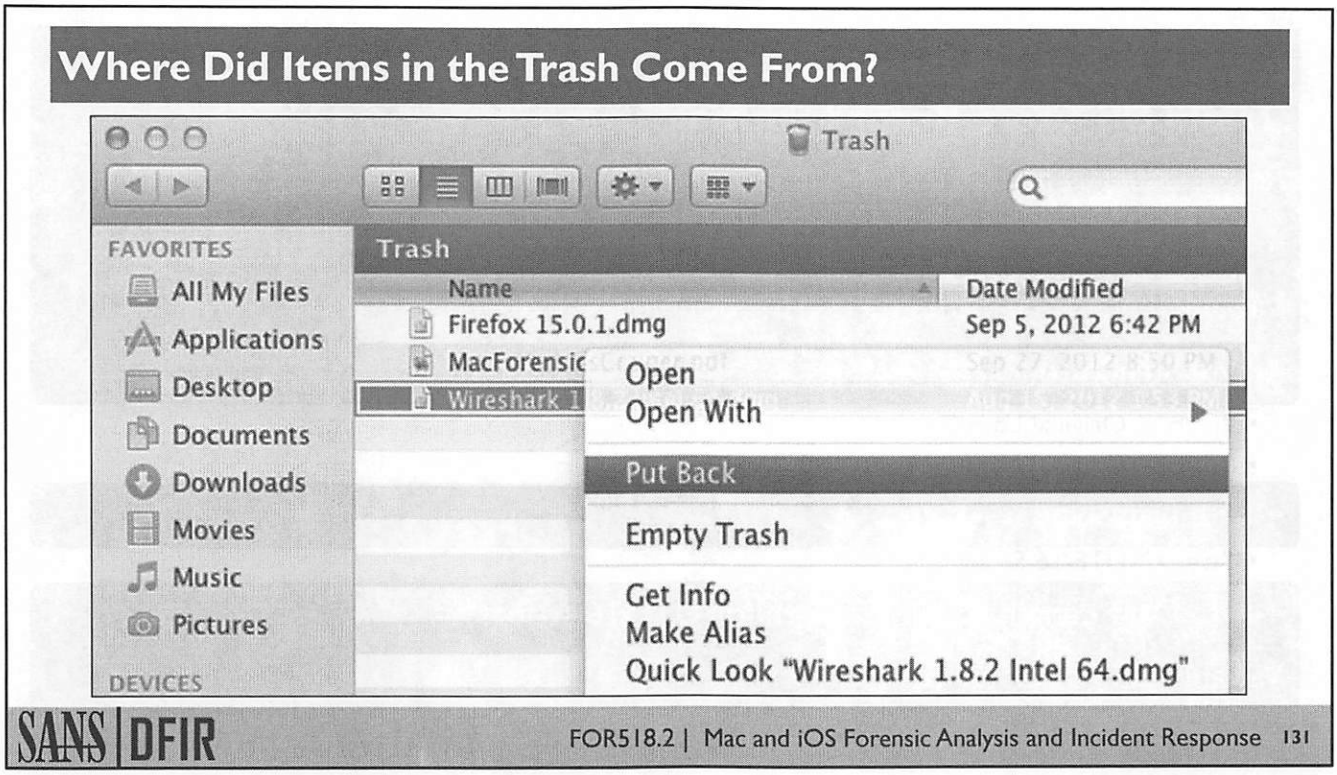
Just the existence of these files can indicate if a directory was browsed using the Finder application.

References:
`https://wiki.mozilla.org/DS_Store_File_Format`
`https://github.com/dscho/dsstore`

Trash: ~/.Trash

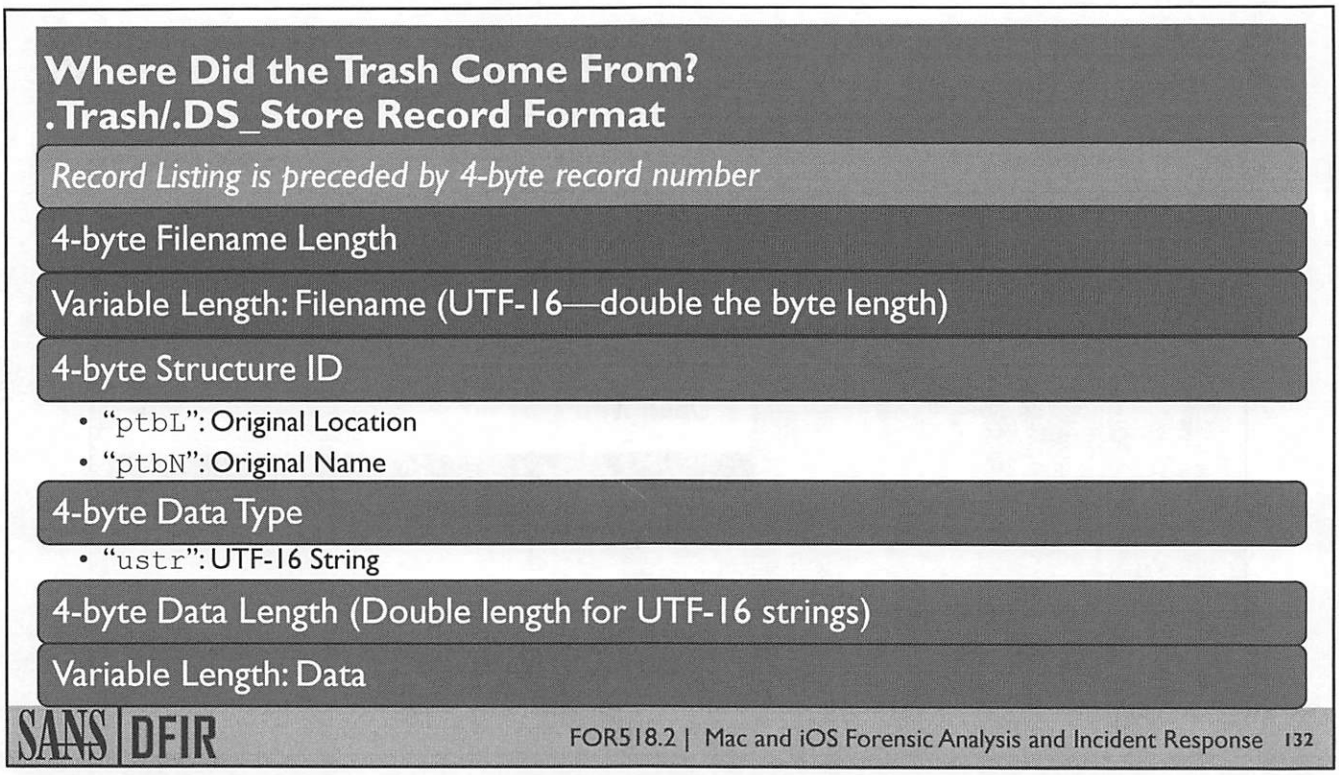
```
Elwoods-Mac:~ elwoodblues$ ls -la
total 40
drwxr-xr-x+ 14 elwoodblues  staff    476 Sep 27 20:50 .
drwxr-xr-x   5 root         admin    170 Sep 23 11:25 ..
-rw-------   1 elwoodblues  staff      3 Sep 23 11:25 .CFUserTextEncoding
-rw-r--r--@  1 elwoodblues  staff  12292 Sep 27 20:50 .DS_Store
drwx------   4 elwoodblues  staff    136 Sep 27 20:50 .Trash
-rw-------   1 elwoodblues  staff     59 Sep 27 20:11 .bash_history
drwx------+  3 elwoodblues  staff    102 Sep 23 20:09 Desktop
drwx------+  4 elwoodblues  staff    136 Sep 23 11:25 Documents
drwx------+  6 elwoodblues  staff    204 Sep 27 20:50 Downloads
drwx------@ 32 elwoodblues  staff   1088 Sep 27 20:34 Library
drwx------+  3 elwoodblues  staff    102 Sep 23 11:25 Movies
drwx------+  3 elwoodblues  staff    102 Sep 23 11:25 Music
drwx------+  4 elwoodblues  staff    136 Sep 23 11:25 Pictures
drwxr-xr-x+  5 elwoodblues  staff    170 Sep 23 11:25 Public
```


Each user has their own hidden `.Trash` directory. This is located in the root of their home directory.



The `.Trash` contains "deleted" items. Each trashed file can be restored using the "Put Back" option on the right-click context menu.

This "Put Back" data can be found in the hidden `.DS_Store` file in the `.Trash` directory.



Each record in the `.DS_store` file can be identified by a UTF-16 filename. Just prior to this filename is a 4-byte filename length. These are followed by a 4-byte Structure ID and 4-byte Data Type. Some of these Structure IDs and Data Types have been documented in the reference pages below. The final part of the record is the data size and data—this can be a variable size.

Each deleted file will have:

- Filename
- Original File Path

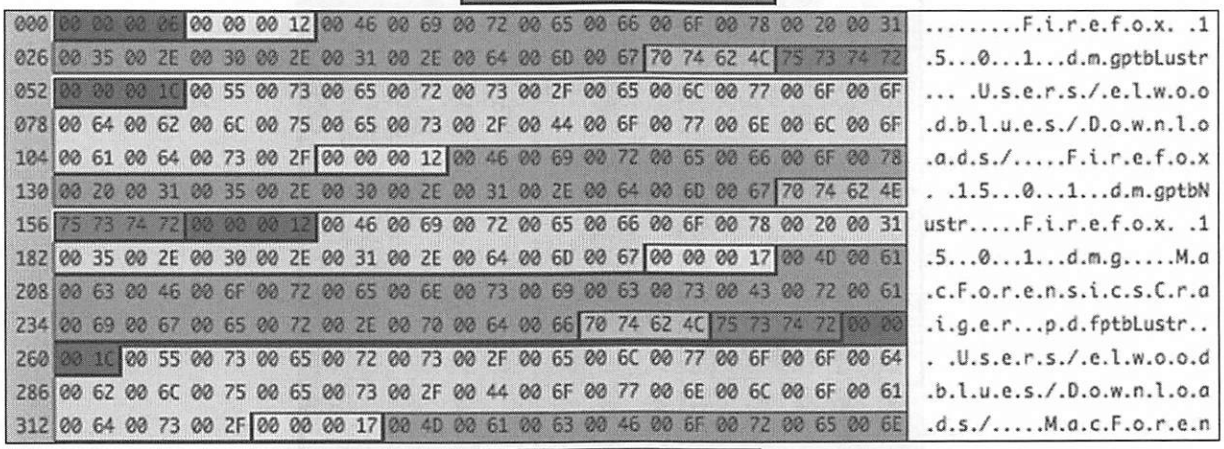
These records are preceded by a 4-byte record number. This will determine how many records are in this file.

There are many more Structure IDs and Data Types that can be seen in .DS_store files, and some have been documented here: `https://metacpan.org/pod/distribution/Mac-Finder-DSStore/DSStoreFormat.pod.`

References:
`https://wiki.mozilla.org/DS_Store_File_Format`
`https://github.com/dscho/dsstore`

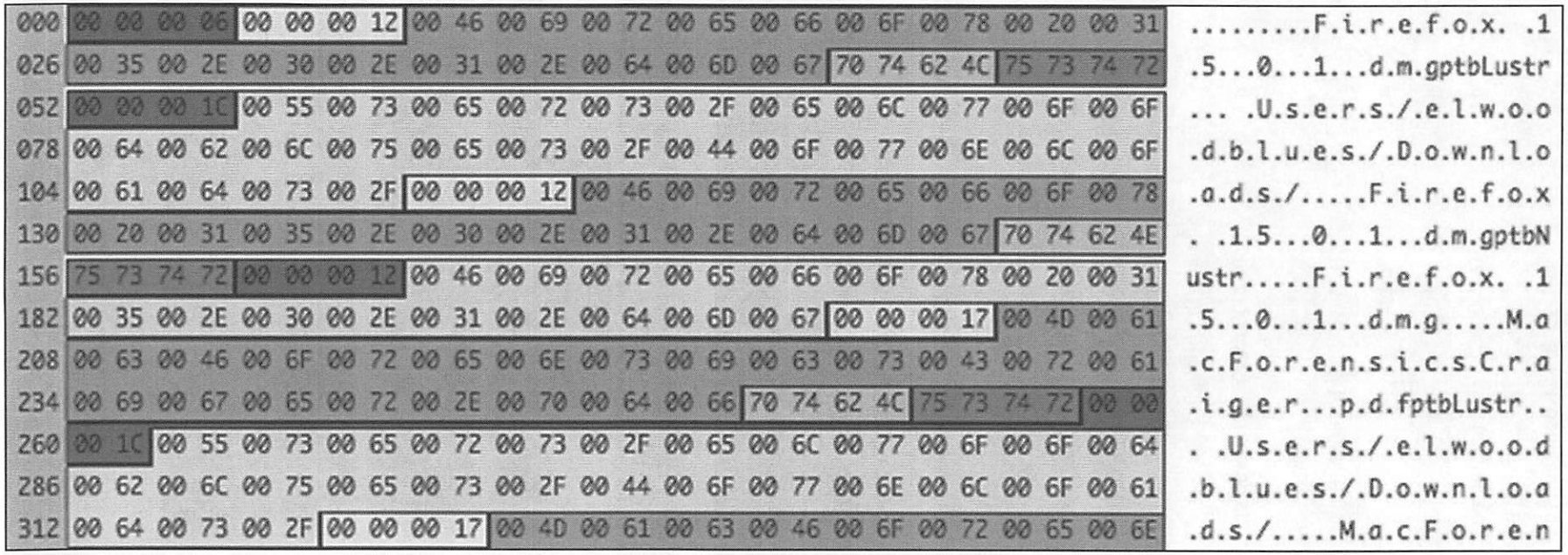
Trash .DS_Store Example

Data Before ...



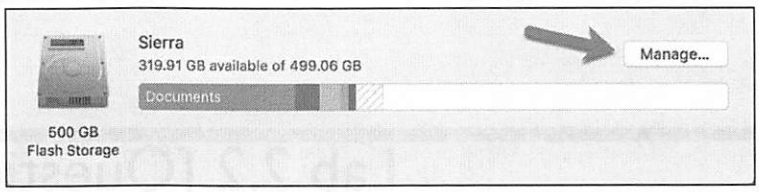
... Data After

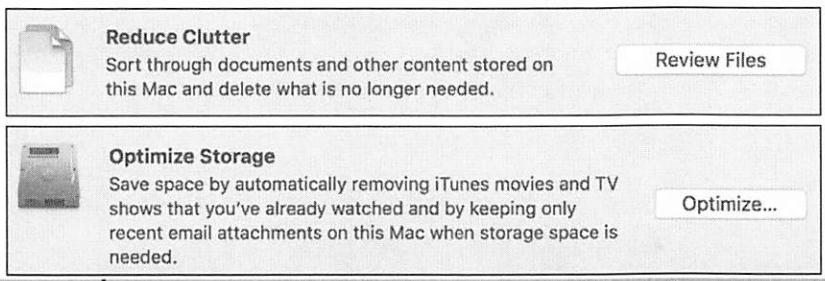
Field	Size (bytes)	Data
Number of Records	4	0x00000006 = 6 Records
Record 1: Filename Size	4	0x00000012 = 18 (characters)
Record 1: Filename	Variable	"Firefox 15.0.1.dmg" (UTF-16: length doubled)
Record 1: Structure ID	4	"ptbL"
Record 1: Data Type	4	"ustr"
Record 1: Data Size	4	0x0000001C = 28 (characters)
Record 1: Data	Variable	"Users/elwoodblues/Downloads/" (UTF-16: length doubled)
Record 2: Filename Size	4	0x00000012 = 18 (characters)
Record 2: Filename	Variable	"Firefox 15.0.1.dmg" (UTF-16: length doubled)
Record 2: Structure ID	4	"ptbN"
Record 2: Data Type	4	"ustr"
Record 2: Data Size	4	0x00000012 = 18 (characters)
Record 2: Data	Variable	"Firefox 15.0.1.dmg" (UTF-16: length doubled)

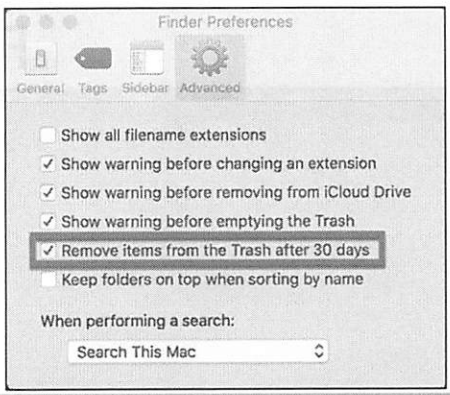


New Trash Features in 10.12

- New Default Option: "Remove items from the Trash after 30 days"
  - Not default in 10.13
- "Optimized Storage" and "Reduce Clutter"
- Downloaded Archive Files via Safari
  - Automatically unarchived and archive moved to Trash





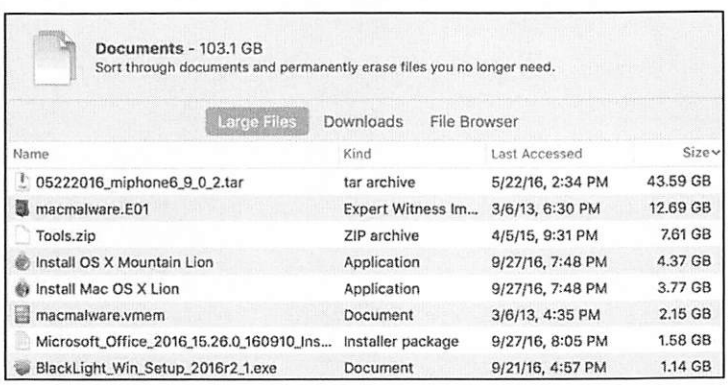


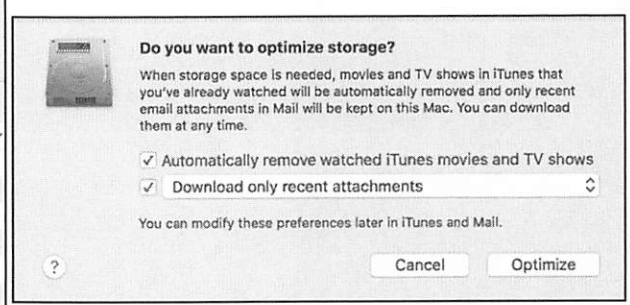
Introduced in Sierra (10.12), macOS now wants to help the user get the most out of the storage space on their systems. The user is reminded of these features when they look at the "`About This Mac | Storage`" menus, as shown in the upper-right screenshot.

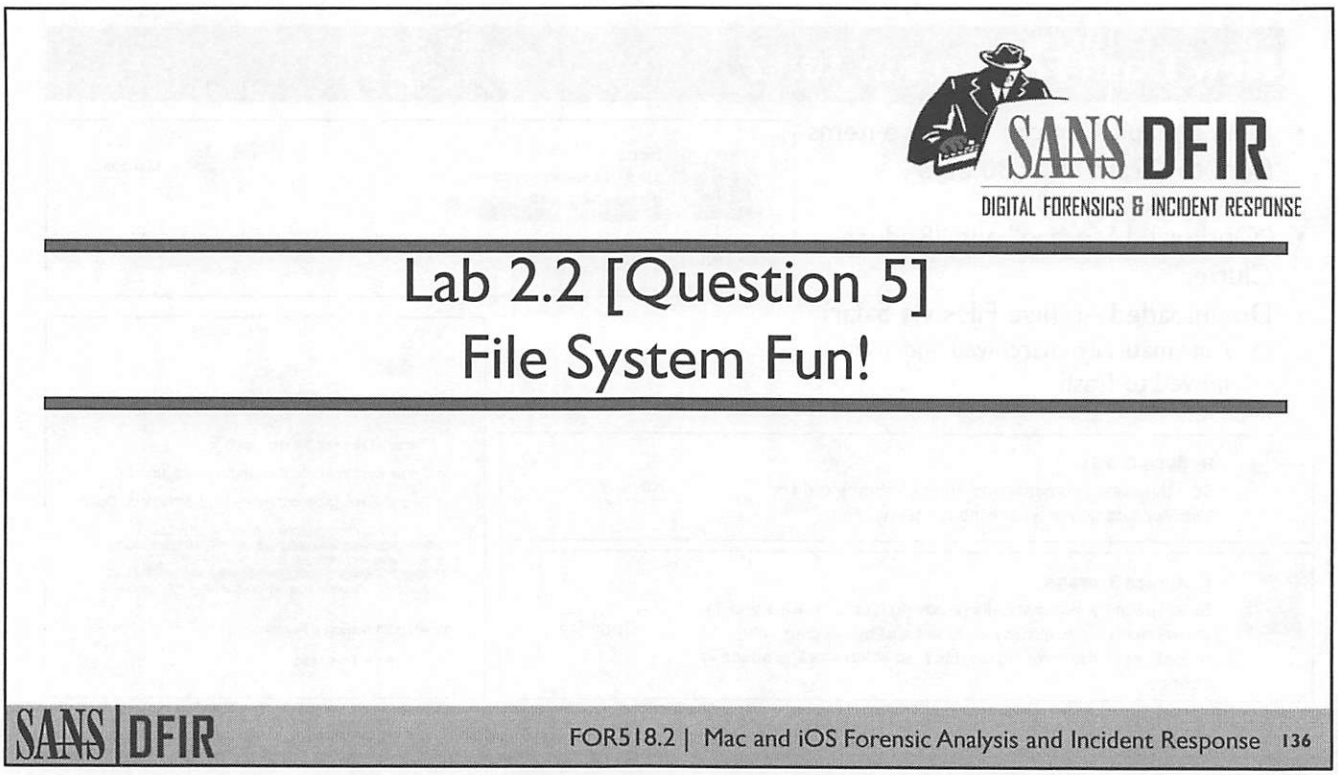
A new default option for the Trash is to remove items after 30 days, whereas before it was stored in the Trash until the user emptied it. This functionality can be turned off in Finder Preferences, as shown above. It is on by default on new clean installs. This does not appear to be on by default in 10.13.

Other functions to be aware of are the "Reduce Clutter" and "Optimize Storage" features. "Reduce Clutter" aims to help the user remove files that have not been used in a while or are very large. The "Review Files" button brings up the window below on the right, which the user can use to delete these items. The "Optimize Storage" feature allows the user to control whether movies are removed after watching and if Mail attachments are downloaded, as shown below on the left.

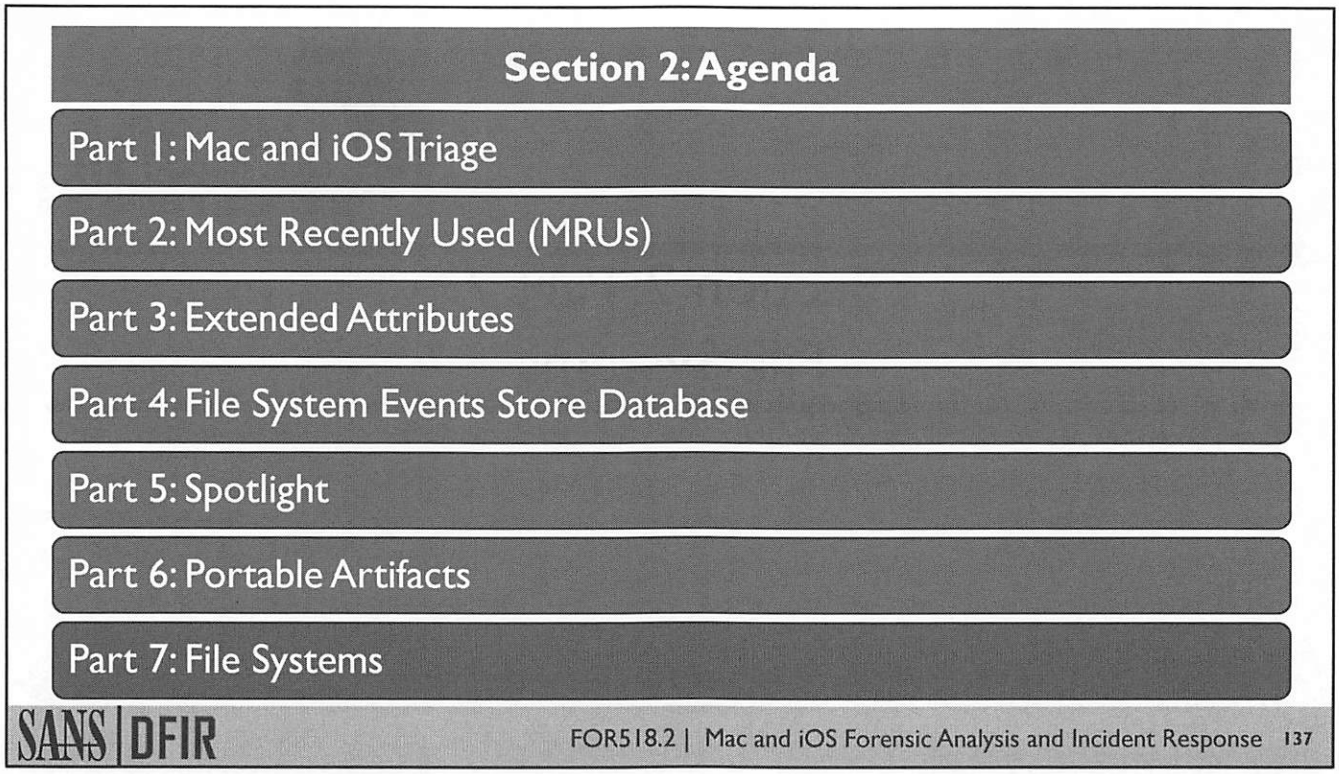
Another "feature" is when an archive file is downloaded through Safari, it is automatically unarchived in `~/Downloads` and the original archive file is moved to `~/.Trash`.



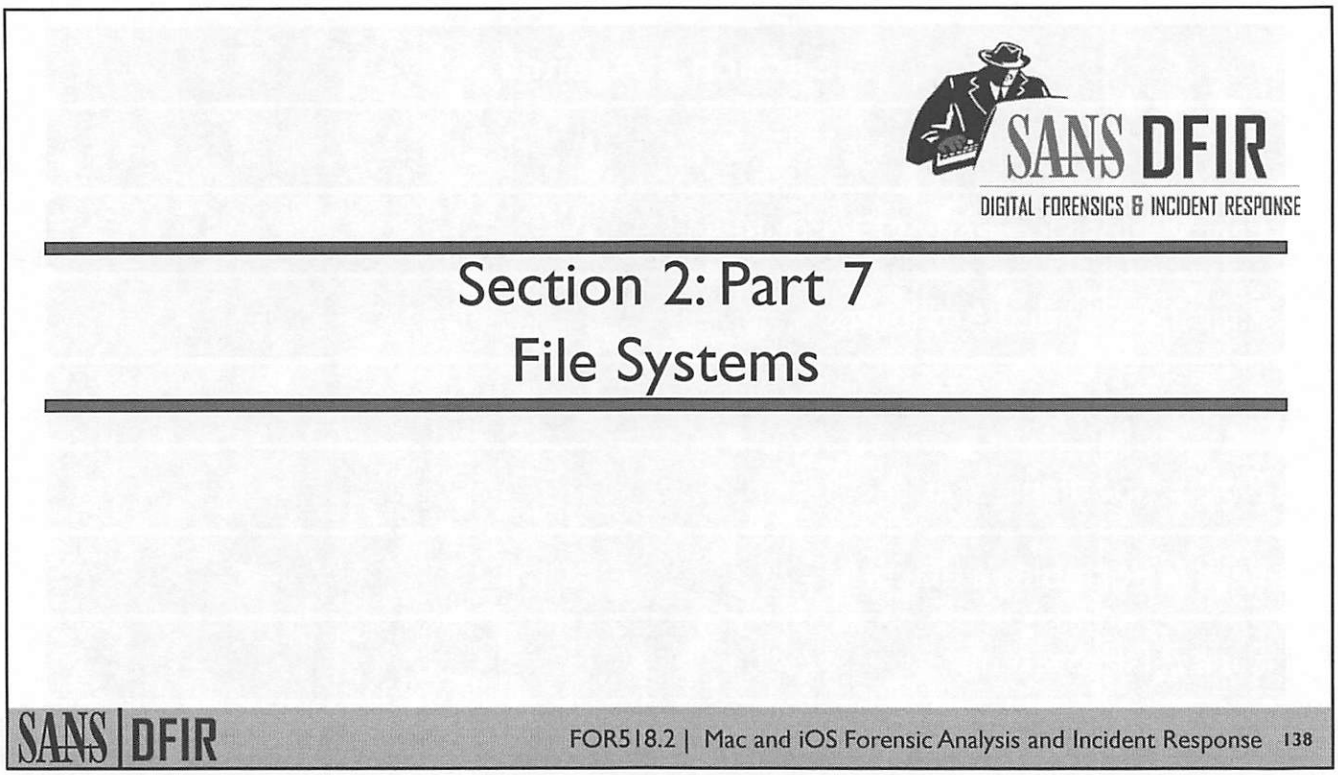




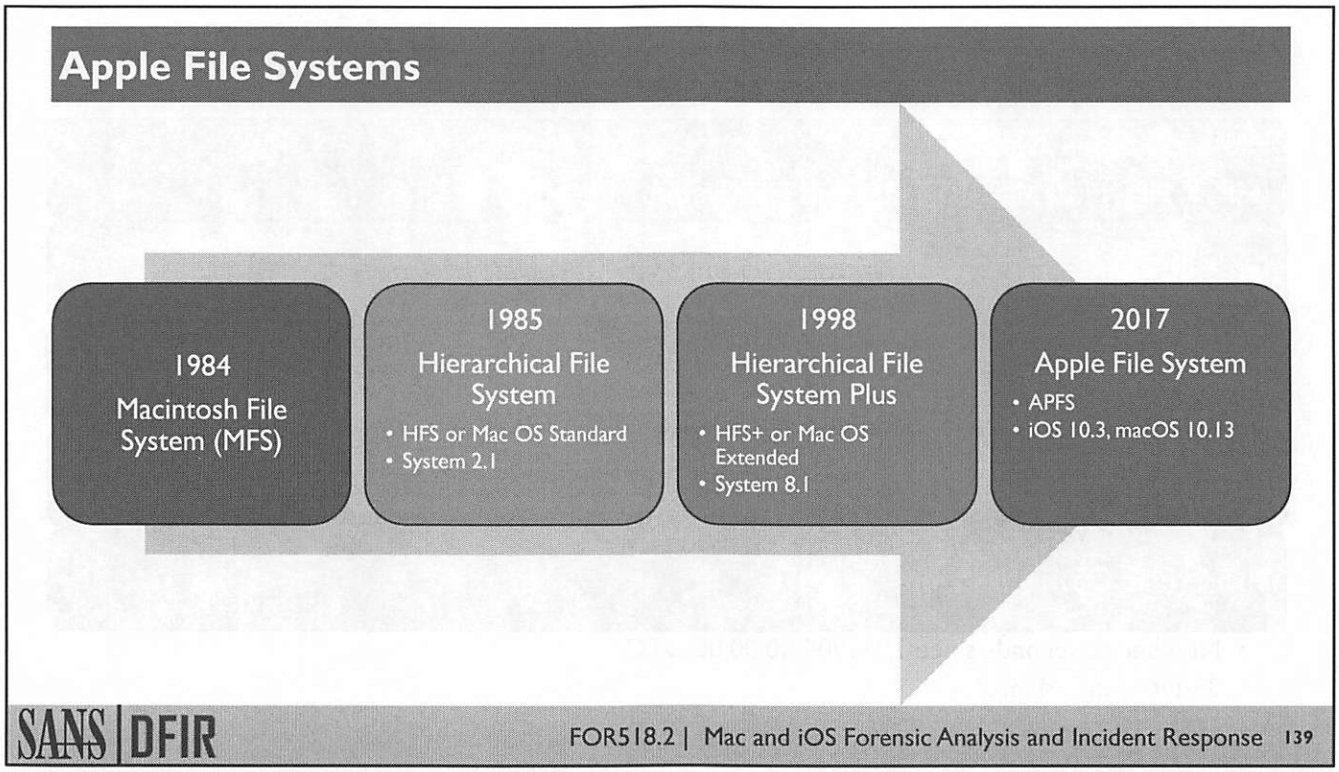
This page intentionally left blank.



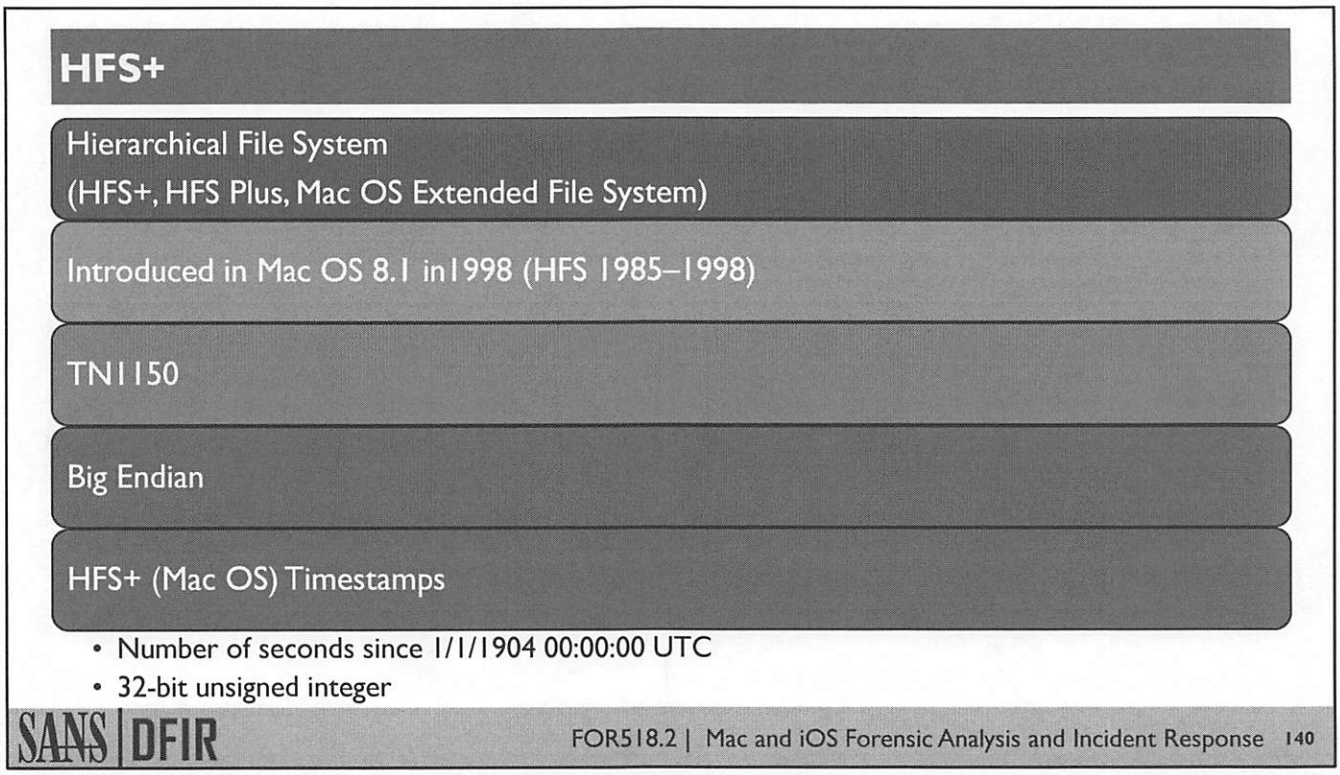
This page intentionally left blank.



This page intentionally left blank.



A major change in recent versions of the operating systems is the change of file system. This will be sure to give us some big analysis challenges!



The Mac OS Extended File System is more widely known as Hierarchical File System or HFS Plus (HFS+). 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 and have longer filenames, larger file sizes, and more file metadata.

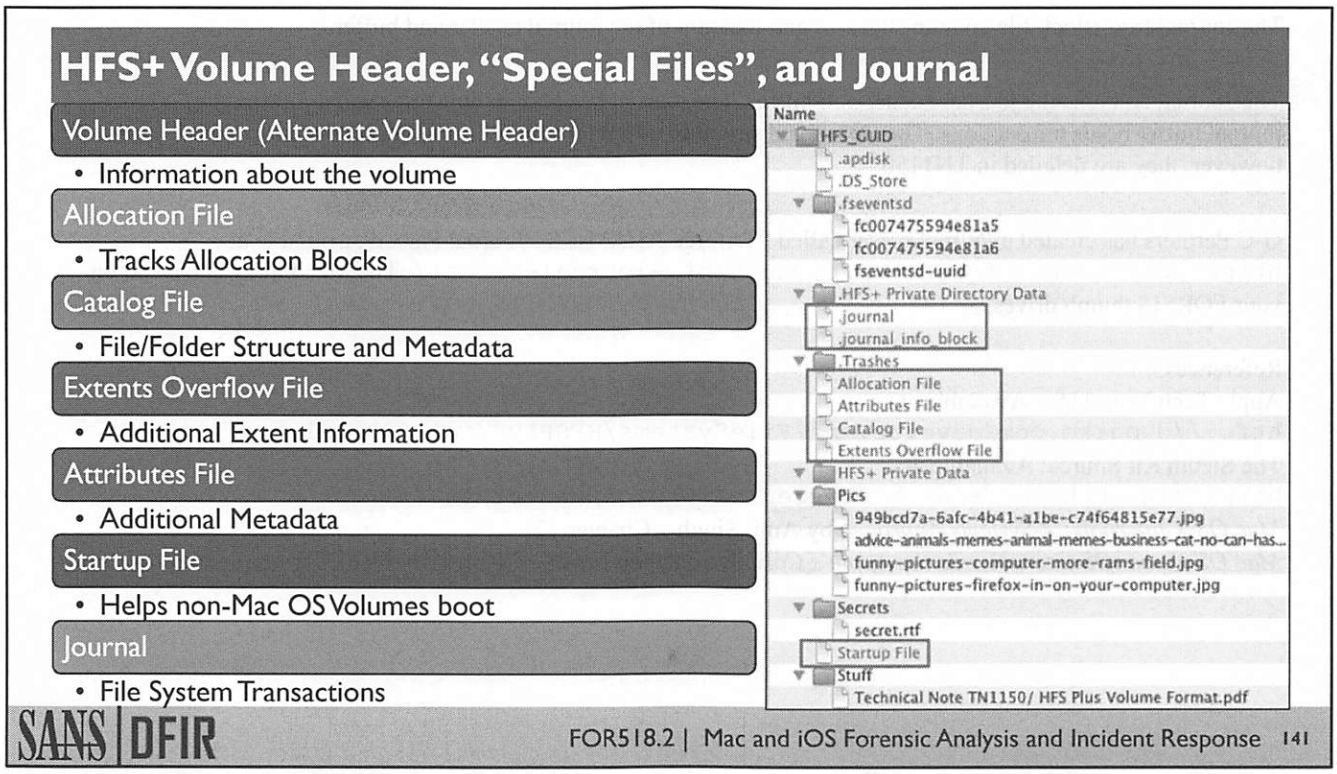
This file system was introduced in Mac OS 8.1 long before Mac OS X.

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.html` 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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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+ 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

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: (`.journal` and `.journal_info_block`).

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

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

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.

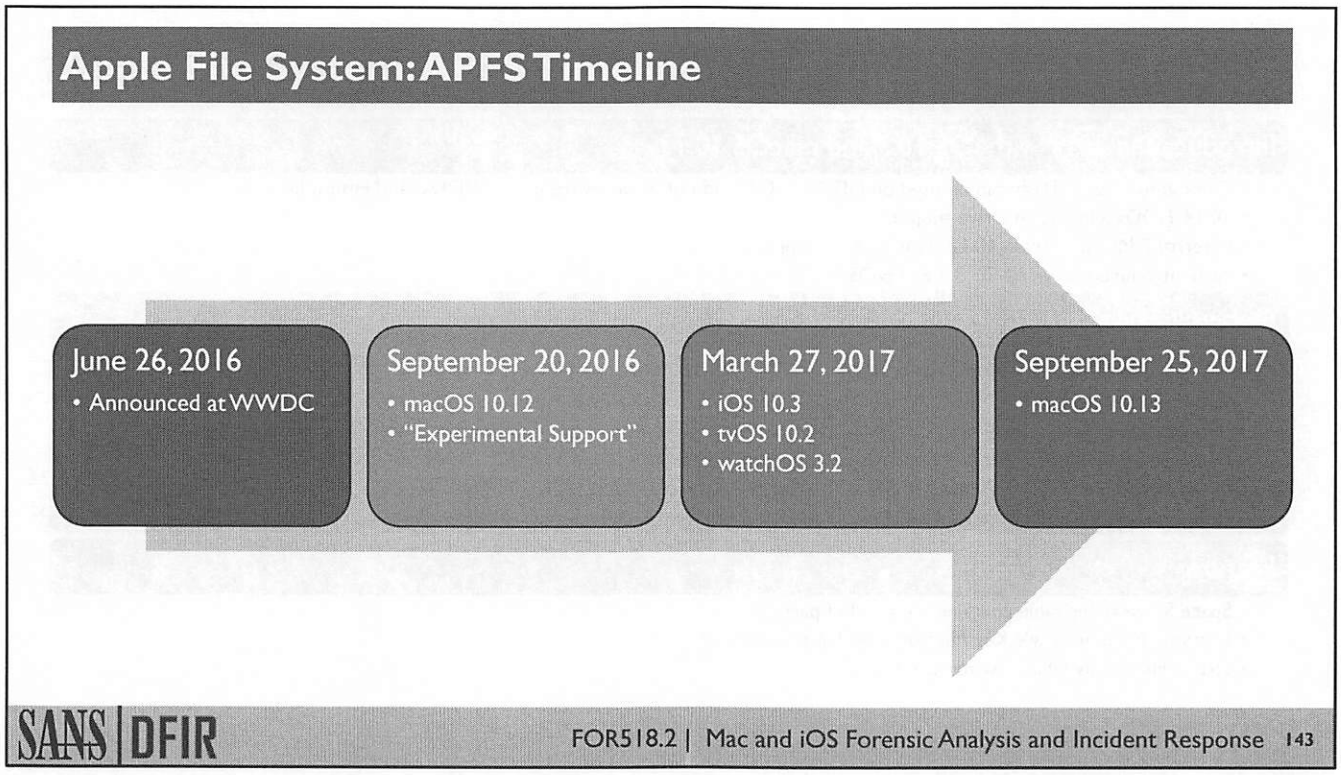
- `.journal_info_block`
- `.journal`

The Journal Info Block file contains the size and location of the journal header and buffer.

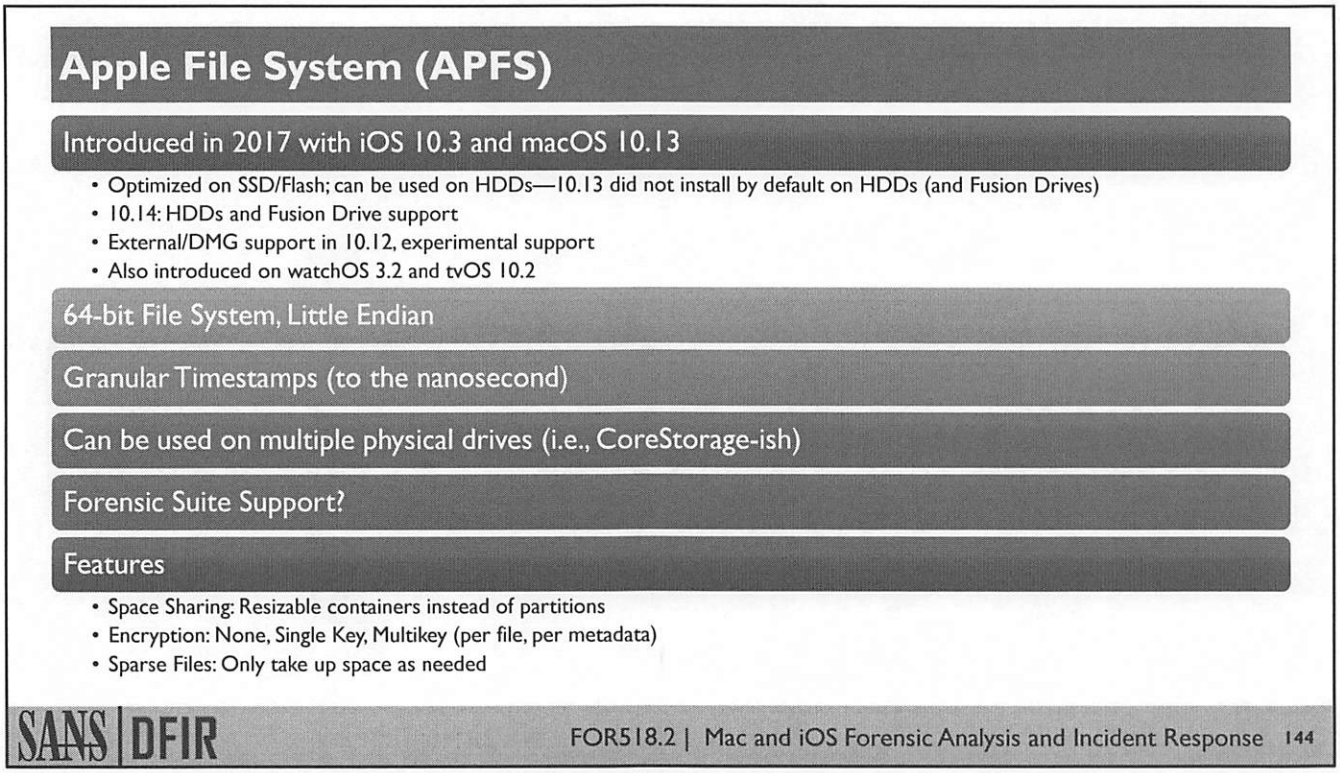
The Journal contains the journal header and buffer. The journal header contains transaction information. The journal buffer holds transactions. The layout and structure of the journal files will not be covered in this class; however, they are detailed in TN1150.

G-C Partners has created a journal parser called "Triforce AHJP HFS+ Journal Parser", available at `https://www.gettriforce.com/product/hfs-journal-parser/`. The Mac binary is located on your FOR518 thumb drives.

References:
Apple Tech Note 1150: Available at
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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

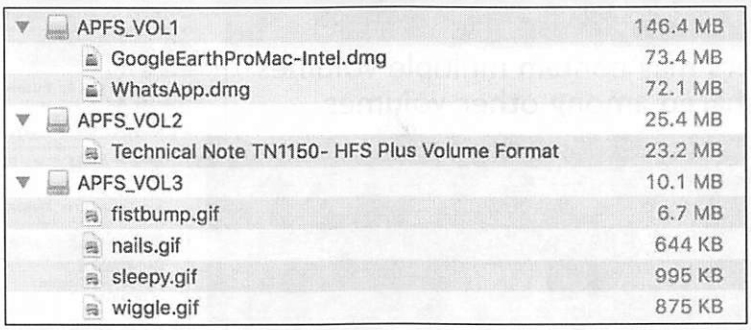


Since APFS was announced at WWDC, it has been introduced to mobile devices first (iOS), experimental support in 10.12, and then finally in 10.13 desktop (macOS) support.



References:
`https://developer.apple.com/library/content/documentation/FileManagement/Conceptual/APFS_Guide/`
`https://developer.apple.com/videos/play/wwdc2016/701/` (View in Safari.)

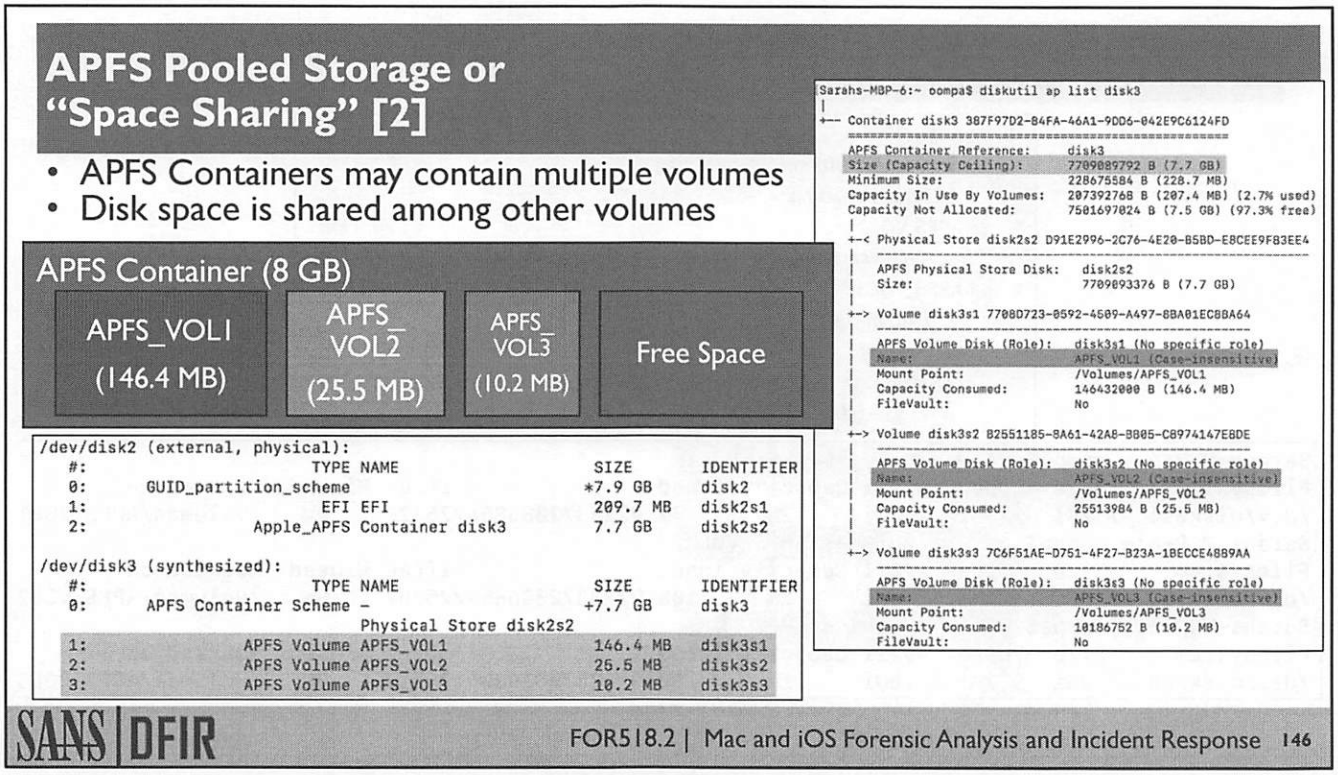
APFS Pooled Storage or "Space Sharing" [1]




```
[Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL1
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk5s1  7.2Gi  140Mi  7.0Gi     2%      79 9223372036854775728    0%   /Volumes/APFS_VOL1
[Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL2
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk5s2  7.2Gi   24Mi  7.0Gi     1%     100 9223372036854775707    0%   /Volumes/APFS_VOL2
[Sarahs-MBP-6:~ oompa$ df -h /Volumes/APFS_VOL3
Filesystem     Size   Used  Avail Capacity iused               ifree %iused  Mounted on
/dev/disk5s3  7.2Gi  9.7Mi  7.0Gi     1%      88 9223372036854775719    0%   /Volumes/APFS_VOL3
```

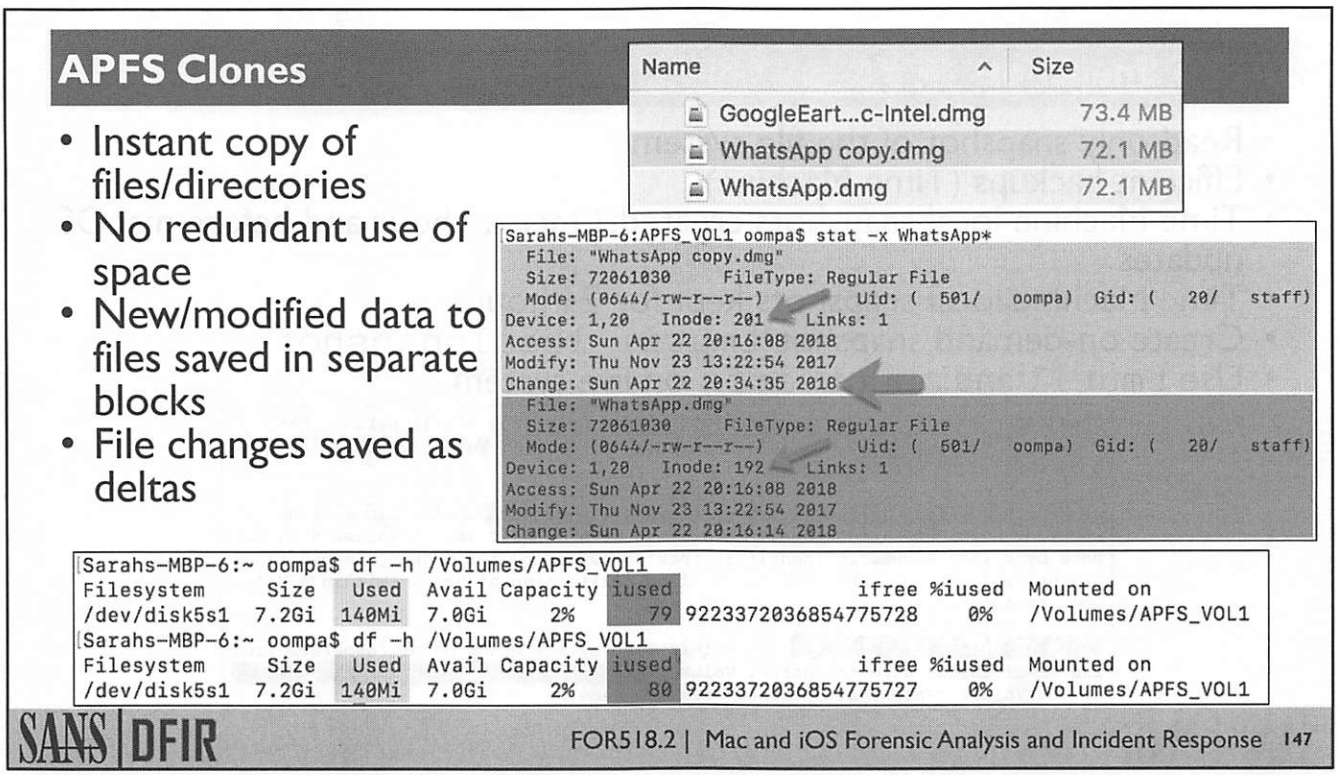

APFS uses "Space Sharing", or pooled storage, meaning that an APFS container may be broken into many APFS volumes that share space. These volumes can grow and shrink as needed using the unallocated space available from what is available on the physical disk(s).

The example above shows a single APFS formatted thumb drive with three APFS volumes: APFS_VOL1, APFS_VOL2, and APFS_VOL3. The files in each of these volumes will take up their needed space as shown in the 'df' command in the screenshot above. The total thumb drive space stays the same, as it is available to all volumes in the APFS container.



The 'diskutil ap list' output on the right shows each APFS volume as a different "consumed" size. The total for this APFS Container is 8 GB (really, 7.7 GB) but that space is shared among each volume. If the user wanted to add an additional volume, as long as there is space available, it would work.

Each volume can be encrypted separately if the user chooses to do so. If this were the case, the "FileVault:" would show a "Yes" instead of the "No" shown above.



APFS uses clones to create instant copies of files without the need to use redundant blocks on the file system for storage. If the cloned file changes, the changes will be saved in deltas on the file system.

The example above shows the cloned file "WhatsApp copy.dmg" that was created from "WhatsApp.dmg". The 'stat' command output shows that the inode number changed, as did the file system metadata (timestamps). The file size listed is also the same, as the file did not change—it was just a copy/clone.

If the 'df' command is used on this volume after the cloned file was created, it will show one more inode used, but the space on the volume stays the same—no additional file system blocks were used.

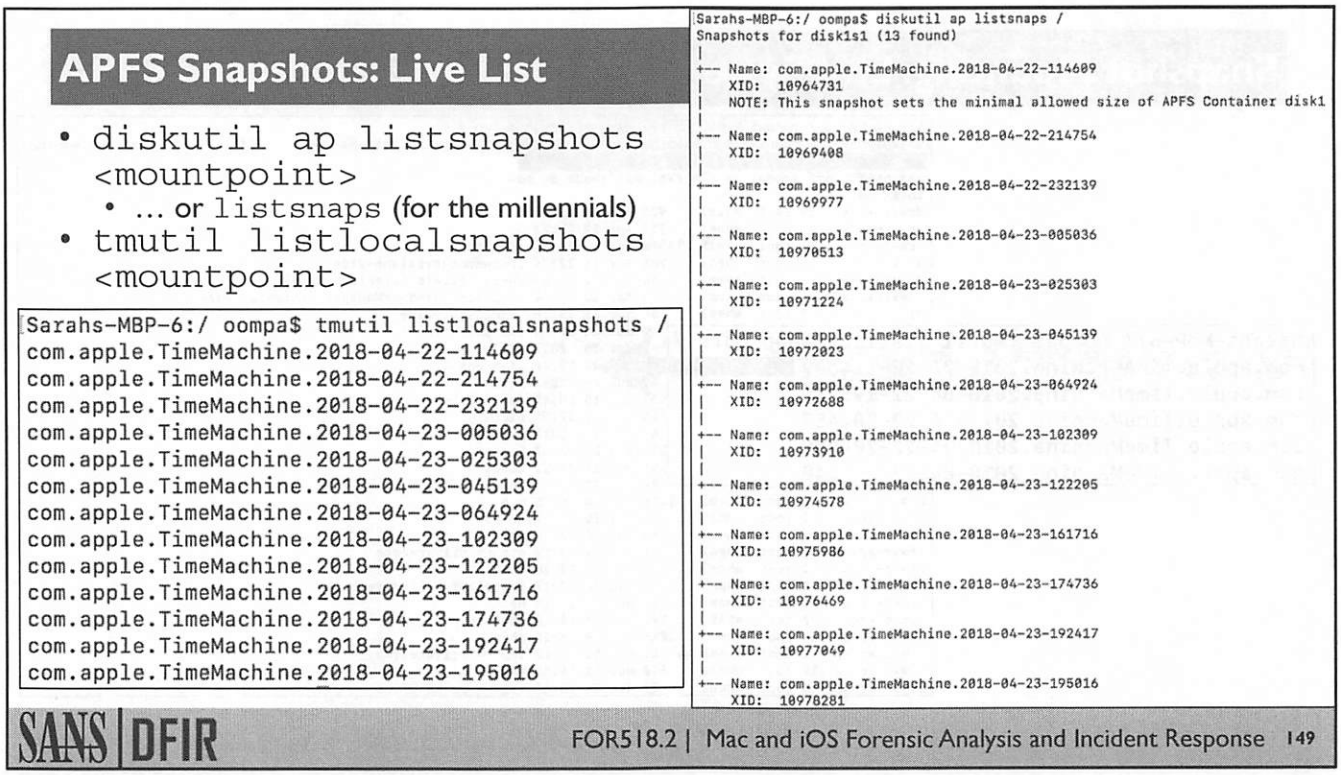
APFS Snapshots

- Read-only snapshot of the file system
- Efficient backups (Time Machine)
- Time Machine local snapshots created: Once an hour and before macOS updates
- Time Machine local snapshots kept for 24 hours
- Create on-demand snapshot 'tmutil localsnapshot'
- Use tmutil and mount_apfs on live system

```
Snapshot - An APFS Snapshot represents a read-only copy of its parent APFS
Volume, frozen at the moment of its creation.  An APFS Volume can have zero or
more associated APFS Snapshots.

APFS Snapshots are neither listed nor discoverable when their Volume is not
mounted. Snapshots are uniquely identified within their parent Volume's names-
pace by either a numeric identifier (preferred) or by their name; Snapshots
can be renamed, but APFS will never allow duplication of names (within a Vol-
ume) to occur.

APFS Snapshots are mountable; when this occurs, its mount point (separate from
and simultaneous with its parent Volume) provides a read-only historic version
of the Volume content at Snapshot creation time.
```

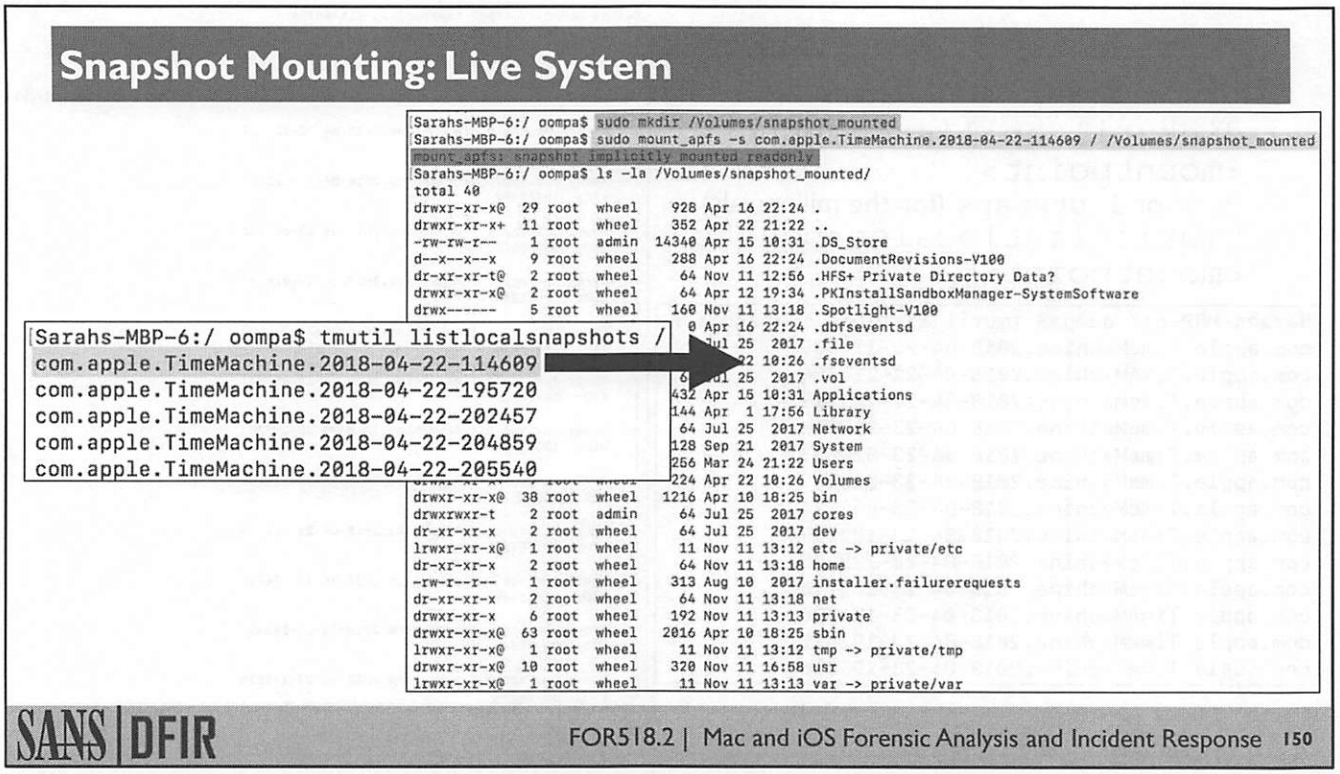

APFS Snapshots are instant copies of file system data. They are currently only used for systems that are using Time Machine. Utilities such as 'tmutil' and 'mount_apfs' can be used to interact with these snapshots, as shown on the next slide.

Reference:
About Time Machine local snapshots: `https://support.apple.com/en-us/HT204015`



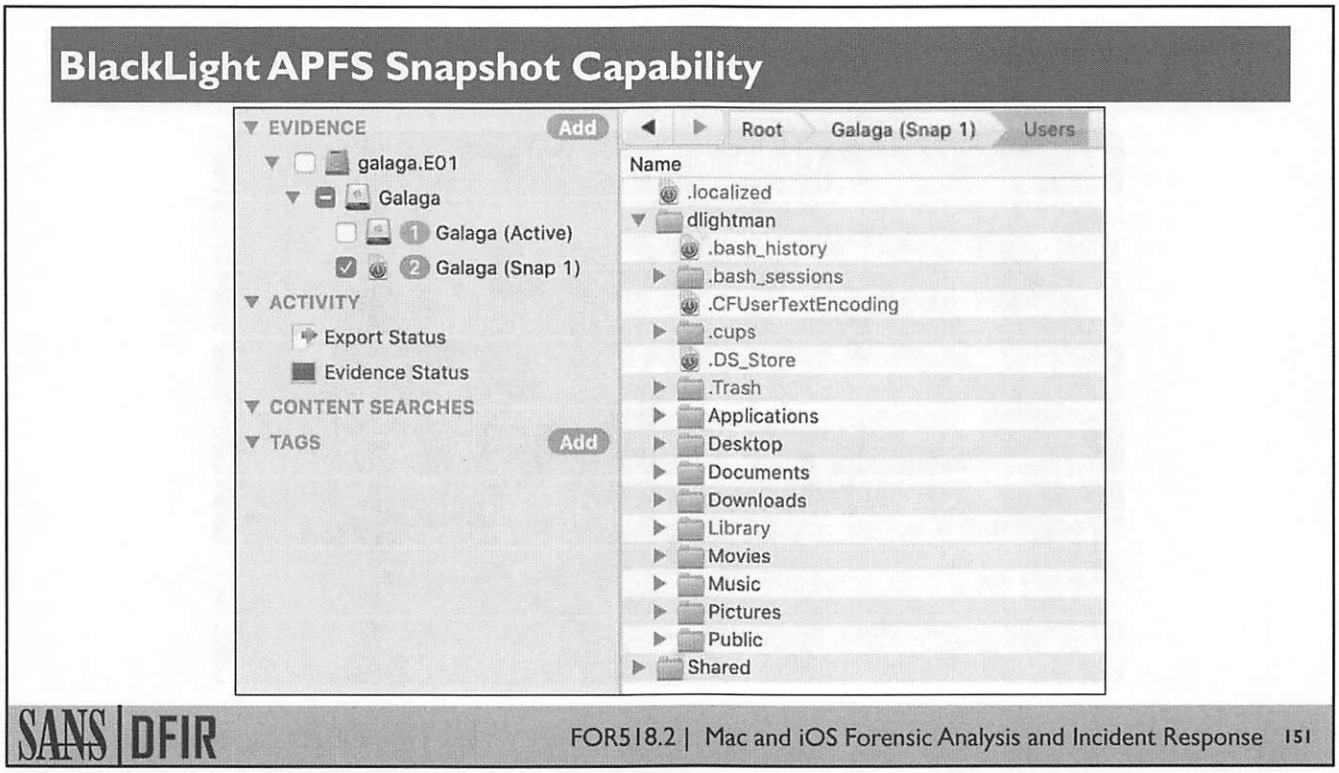
'diskutil ap listsnapshots' (or 'diskutil ap listsnaps') can be used to view what APFS Snapshots are available to use.

The 'tmutil' utility can then be used to mount one of these local snapshots to a point where it can be browsed and analyzed as any other mounted volume on the system.



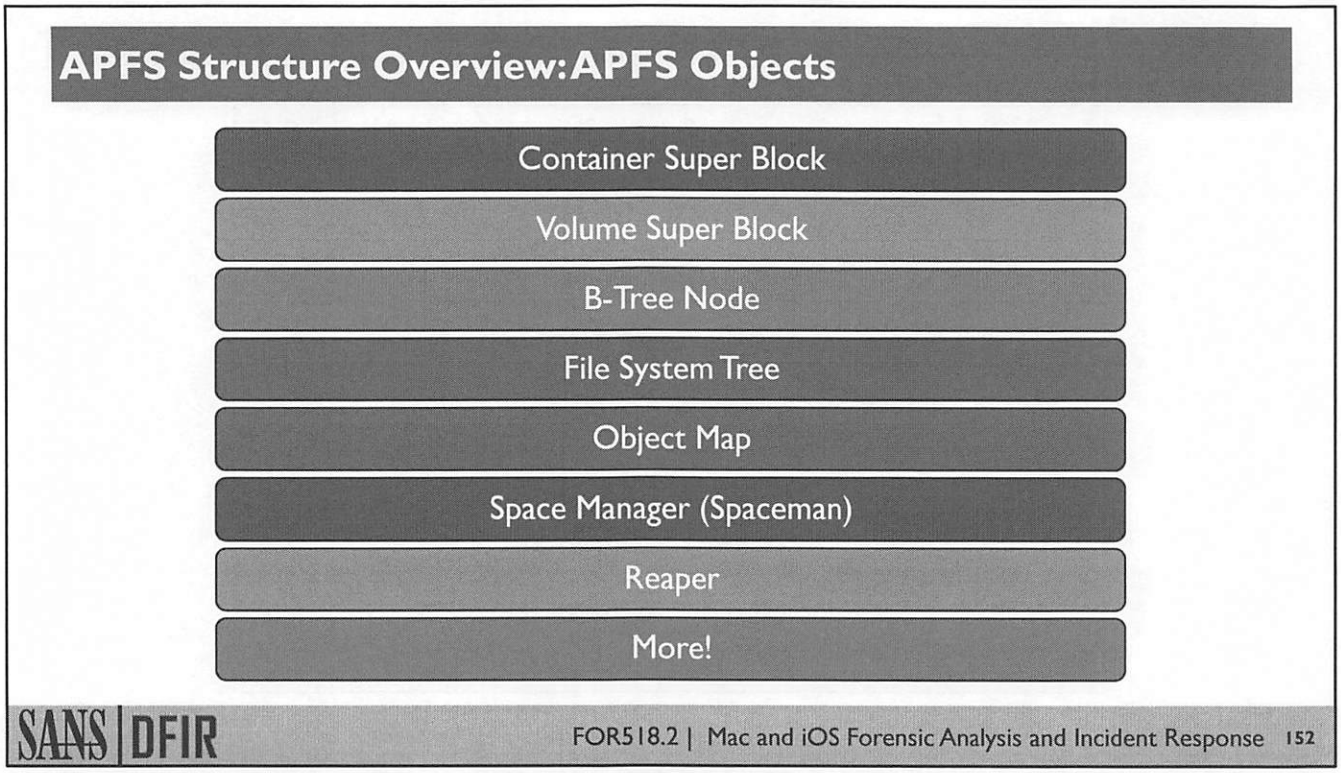
The live APFS snapshots can be mounted using the mount_apfs command just like regular APFS disk images can. The only difference is using the '-s' argument, which specifies the snapshot to mount. This can give the user a historical view of a previous version of the user's system.

The snapshots are mounted in a read-only state on the specified mount point.



BlackLight has capability to parse the APFS Snapshots; you'll find this option in the Advanced Parsing area.

Once parsed, you'll have an additional Volume (Snap 1), depending on how many snapshots were parsed from the APFS volume.



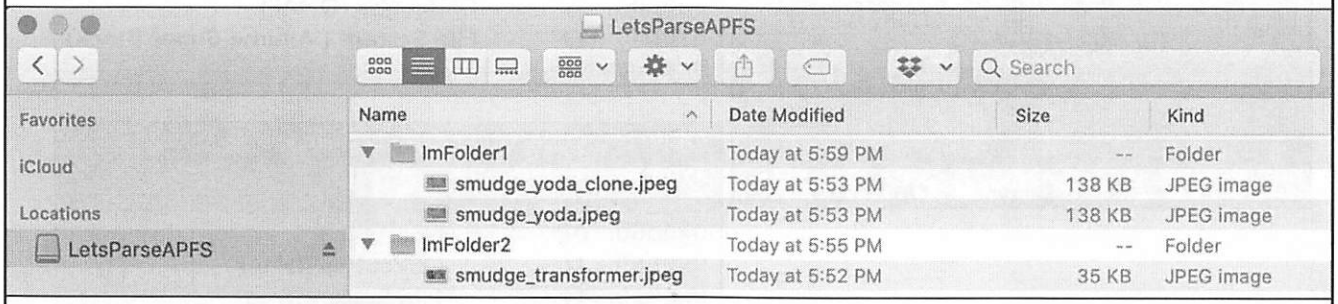
Unlike HFS+, there are no "Special Files" on APFS. All the file system structures are embedded in the file system as objects.

Objects are stored on disk in blocks; a common block size is 4096 bytes. Therefore, you can assume you have a new object every block. The organization of the blocks do not matter, as that is rectified in various file system objects keeping track of where everything is.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Structure Overview

- Example AFS Container
  - One Volume: "LetsParseAPFS"
  - Two Directories
  - Cloned File



In the next few slides, we will be diving into the structures associated with the "LetsParseAPFS" DMG file shown above.

This DMG file is simple, yet it shows many of the features you will find in a full file system APFS disk.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Object Header and Types [obj_phys_t]: 32 Bytes

Offset	Size (in bytes)	Field	Notes
0	8	o_cksum	Fletcher 64 Checksum
8	8	o_oid	Object ID
16	8	o_xid	Transaction ID
24	2	o_type.type	**Object Type**
26	2	o_type.flags	Object Flags
28	4	o_subtype	**Object Subtype**

Object Type (Hex)	Object Type (Dec)	Object Type
0x0100	1	**Container Super Block**
0x0200	2	B-Tree
0x0300	3	**B-Tree Node**
0x0500	5	Spaceman
0x0B00	11	Object Map (OMAP)
0x0D00	13	**File System (Volume Super Block)**
0x1100	17	Reaper

Object Type (Hex)	Object Type (Dec)	Object Subtype
0x0000	0	None
0x0B00	11	Object Map (OMAP)
0x0E00	14	**File System Tree**

Each object in APFS has a 32-byte object header. This header describes what kind of object you are looking at. A few of these objects are listed above.

We will be dissecting a Container Super Block, a Volume Super Block, and a B-Tree Node (with a File System tree).

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Container Super Block (nx_superblock_t) [1]

Size (in bytes)	Field	Notes
32	obj_phys_t	Object Header (Object Type = 1)
4	nx_magic	Container Magic Number: 0x4E585342 = "NSXB"
4	nx_block_size	Block Size (i.e., 4096)
8	nx_block_count	Block Count (Block Count*Block Size = Container Size in Bytes
8	nx_features	Features
8	nx_read_only_ compatible_features	Read-only Compatible Features
8	nx_incompatible_features	Incompatible Features
16	nx_uuid	Container UUID (`diskutil info /dev/disk#`)
8	nx_next_oid	Next Object ID (OID)
8	nx_next_xid	Next Transaction ID (XID)
4	nx_xp_desc_blocks	Blocks used by Checkpoint Descriptor Area
4	nx_xp_data_blocks	Blocks used by Checkpoint Data Area

The structure of the Container Super block is above. This object keeps track of what is going on in an APFS Container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Container Super Block (nx_superblock_t) [2]

Size (in bytes)	Field	Value and Notes
8	nx_xp_desc_base	Base address of Checkpoint Descriptor Area or Physical Object ID
8	nx_xp_data_base	Base address of Checkpoint Data Area or Physical Object ID
4	nx_xp_desc_next	Next Index for Checkpoint Descriptor Area
4	nx_xp_data_next	Next Index for Checkpoint Data Area
4	nx_xp_desc_index	Index for first item in Checkpoint Descriptor Area
4	nx_xp_desc_len	Number of blocks in Checkpoint Descriptor Area Used
4	nx_xp_data_index	Index for first item in Checkpoint Data Area
4	nx_xp_data_len	Number of blocks in Checkpoint Data Area Used
8	nx_spaceman_oid	Space Manager Object ID (OID)
8	nx_omap_oid	Container Object Map Object ID (OID)
8	nx_reaper_oid	Reaper Object ID (OID)
4	nx_test_type	Reserved for Testing
4	nx_max_file_systems	Maximum Number of Volumes in this Container
8	nx_fs_oid[0]	Array of OIDs for Volumes in this Container
...	...	*Continuation of Container Super Block Structure*

The structure of the Container Super block is above. This object keeps track of what is going on in an APFS Container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[1]

Size (in bytes)	Field	Notes
32	obj_phys_t	Object Header (Object Type = 13)
4	apfs_magic "APSB"	Volume Magic Number 0x41505342 = "APSB"
4	apfs_fs_index	Index in Volume Array
8	apfs_features	Features
8	apfs_readonly_compatible_features	Read-only Incompatible Features
8	apfs_incompatible_features	Incompatible Features
8	apfs_unmount_time	Timestamp when volume was last unmounted
8	apfs_fs_reserve_block_count	Block Pre-allocated for Volume (Default is none)
8	apfs_fs_quota_block_count	Maximum Block Allocated (Default is none)
8	apfs_fs_alloc_count	Number of blocks currently allocated

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[2]

Size (in bytes)	Field	Notes
2	wrapped_crypto_state_t. wrapped_crypto_state.major_version	Key Encryption Metadata: Major Version
2	wrapped_crypto_state_t. wrapped_crypto_state.minor_version	Key Encryption Metadata: Minor Version
4	wrapped_crypto_state_t. wrapped_crypto_state.cpflags	Key Encryption Metadata: Encryption State Flags
4	wrapped_crypto_state_t. wrapped_crypto_state.persistent_class	Key Encryption Metadata: Protection Class
4	wrapped_crypto_state_t. wrapped_crypto_state.key_os_version	Key Encryption Metadata: Creator OS Version 0x39004313 = 19 C 57: 19C57: Catalina 10.15.2
2	wrapped_crypto_state_t. wrapped_crypto_state.key_revision	Key Encryption Metadata: Key Version
2	wrapped_crypto_state_t. wrapped_crypto_state.key_len	Key Encryption Metadata: Key Size (0 for no Encryption)
0	wrapped_crypto_state_t. wrapped_crypto_state.persistent_key	Key Encryption Metadata: Wrapped Key No Key field is null; see key_len above

158

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[3]

Size (in bytes)	Field	Notes
4	apfs_root_tree_oid_type	Type of Root File System Tree = B-Tree
4	apfs_extentref_tree_oid_type	Type of Extent Reference Tree = B-Tree, Physical
4	apfs_snap_meta_tree_oid_type	Type of Snapshot Metadata Tree = B-Tree, Physical
8	apfs_omap_oid	Physical Object ID (OID) of Object Map
8	apfs_root_tree_oid	Virtual Object ID (OID) of Root File System Tree
8	apfs_extentref_tree_oid	Physical Object ID (OID) of Extent Reference Tree
8	apfs_snap_meta_tree_oid	Virtual Object ID (OID) of Snapshot Metadata Tree
8	apfs_revert_to_xid	Transaction ID (XID) that volume will revert to
8	apfs_revert_to_sblock_oid	Virtual Object ID (OID) of Volume Superblock to revert to
8	apfs_next_obj_id	Next Object ID (OID)
8	apfs_num_files	Number of Regular Files
8	apfs_num_directories	Number of Directories
8	apfs_num_symlinks	Number of Symbolic Links
8	apfs_num_other_fsobjects	Number of Other Files
8	apfs_num_snapshots	Number of Snapshots

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

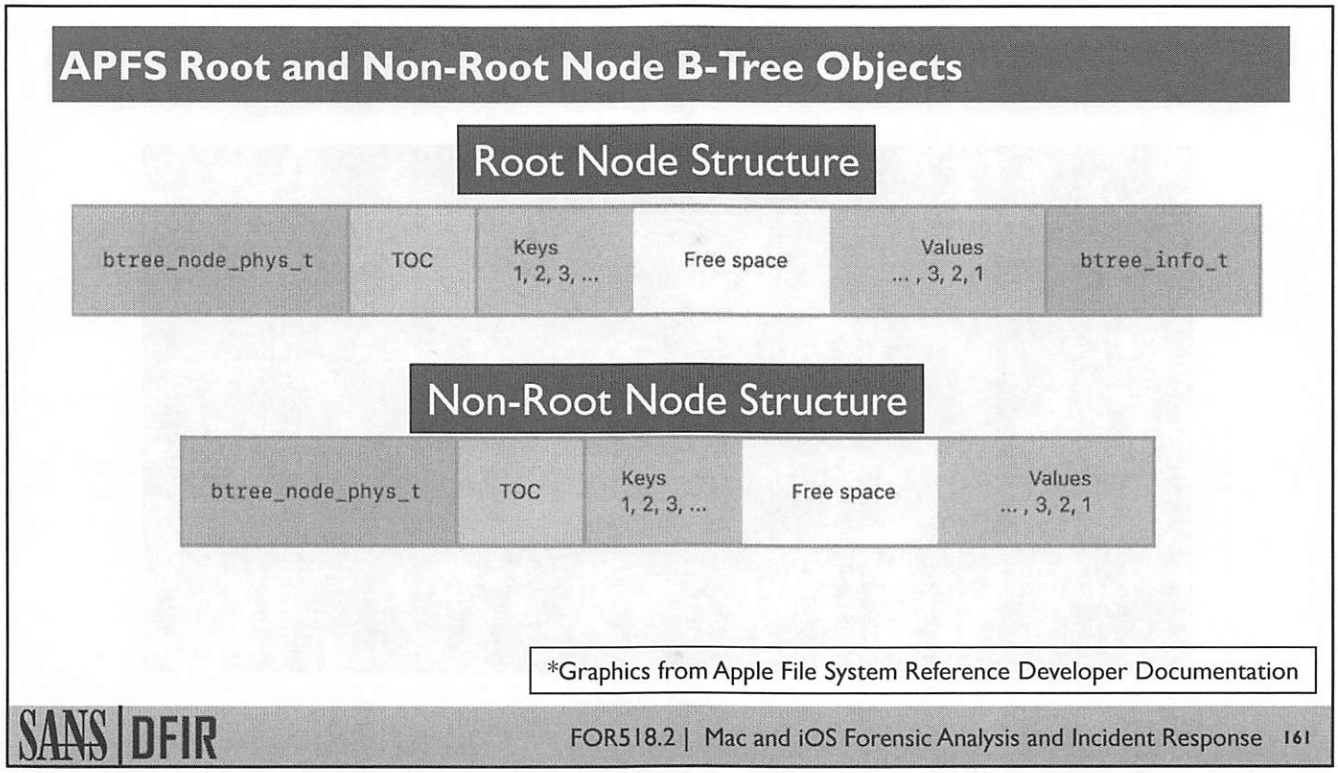
Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS Volume Super Block (apfs_superblock_t)[4]

Size (in bytes)	Field	Notes
8	apfs_total_blocks_alloced	Blocks Allocated by Volume
8	apfs_total_blocks_freed	Blocks Freed by Volume
16	apfs_vol_uuid	Volume UUID (diskutil info /dev/disk#s# [Volume])
8	apfs_last_mod_time	Last Modified Timestamp
8	apfs_fs_flags	Flags
32	apfs_modified_by_t.formatted_by.id[]	Format Program and Version
8	apfs_modified_by_t.formatted_by. timestamp	Format Timestamp
8	apfs_modified_by_t.formatted_by. last_xid	Format Transaction ID (XID)
32	apfs_modified_by_t.modified_by.id[]	Last Modified Program and Version
8	apfs_modified_by_t.modified_by. timestamp	Last Modified Timestamp
8	apfs_modified_by_t.modified_by. last_xid	Last Modified Transaction ID (XID)
336	apfs_modified_by_t.modified_by[1-7]	Array of apfs_modified_by_t[8]
256	apfs_volname	APFS Volume Name
4	apfs_next_doc_id	Next Document ID
2	apfs_role	APFS Role (None, System, Data, Preboot, VM, Recovery)
2	apfs_reserved	Reserved
8	apfs_root_to_xid	Transaction ID (XID) of Snapshot to Root
8	apfs_er_state_oid	Current State of Encryption/Decryption 160

The structure of the Volume Super block is above. This object keeps track of what is going on in an APFS Volume. An APFS container may have more than one APFS volume embedded in it. There may be multiple Volume Super Blocks for each volume in the container.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf



The file system metadata for each file, directory, extended attribute, etc. is stored in a B-Tree Object, much like that of an HFS+ file system.

The structure that we will look at today is a Non-Root Node B-Tree. This B-Tree contains leaf nodes that hold all the file system metadata for each file system entry.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree Node (btree_node_phys_t)

Size (in bytes)	Field	Value & Notes
32	obj_phys_t	Object Header (Object Type = 3, B-Tree Node, Subtype = 14, File System Tree)
2	btn_flags	Flags (Leaf Node)
2	btn_level	Number of Child Levels below this Node
4	btn_nkeys	Number of Keys
2	btn_table_space.off	Offset to Table of Contents (after btree_node_phys_t)
2	btn_table_space.len	Length of Table of Contents
2	btn_freespace.off	Offset Key/Value Free Space
2	btn_freespace.len	Length of Key/Value Free Space
2	btn_key_free_list.off	Offset to Free Key Space
2	btn_key_free_list.len	Length of Free Key Space
2	btn_val_free_list.off	Offset to Free Value Space
2	btn_val_free_list.len	Length of Free Value Space

Each B-Tree Node contains a structure to various pieces of the B-Tree. In this structure, you have how many key entries are stored in this B-Tree, along with where the keys and values for the entries are found in the tree.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Table of Contents (TOC)

TOC Entry (8 bytes)	key_offset (2 bytes)	key_length (2 bytes)	value_offset (2 bytes)	value_length (2 bytes)	Object ID (Inode #)
1	0x0000 = 0	0x1800 = 24	0x1200 = 18	0x1200 = 18	01 private-dir
2	0x1800 = 24	0x1100 = 17	0x2400 = 36	0x1200 = 18	01 root
3	0x2900 = 41	0x0800 = 8	0x9000 = 144	0x6C00 = 108	02
4	49	22	162	18	02
5	71	22	180	18	02
6	93	23	198	18	02
... continues on for number of keys (btn_nkeys) ...					

The Table of Contents (TOC) section of the B-Tree node contains the offsets and lengths for each key/value pair in the tree.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Table of Contents (TOC)—Example

B-Tree Offset in Example	TOC Entry	key_offset (2 bytes)	key_length (2 bytes)	value_offset (2 bytes)	value_length (2 bytes)	Object ID (Inode #)
... continues on for number of keys (btn_nkeys) ...						
264	27	555	8	1405	168	0x14 = 20
272	28	563	36	1425	20	0x14 = 20
280	29	599	31	1486	61	0x14 = 20
288	30	630	28	1516	30	0x14 = 20
296	31	658	8	1520	4	0x14 = 20
304	32	666	16	1544	24	0x14 = 20
... continues on for number of keys (btn_nkeys) ...						

The Table of Contents (TOC) section of the B-Tree node contains the offsets and lengths for each key/value pair in the tree.

The example above contains the TOC contents for a particular file—Inode 20.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: File System Keys (Inode 0x14 [20])

B-tree Offset	Entry	Offset	Size (in bytes)	Object ID (Inode #, 7 bytes)	Entry Kind (Highest byte)	Entry Type	Value and Notes
995	27	555	8	0x140000000 00000 = 20	0x30	Inode	N/A
1003	28	563	36	0x140000000 00000 = 20	0x40	Xattr	com.apple.lastuseddate#PS [2-byte size before, 1 byte padding after]
1039	29	599	31	0x140000000 00000 = 20	0x40	Xattr	com.apple.quarantine [2-byte size before, 1 byte padding after]
1070	30	630	28	0x140000000 00000 = 20	0x40	Xattr	com.dropbox.attrs [2-byte size before, 1 byte padding after]
1098	31	658	8	0x140000000 00000 = 20	0x60	Data Stream	N/A
1106	32	666	16	0x140000000 00000 = 20	0x80	File Extent	0x0000000000000000

Inode 20 (0x14) has six key/value pairs associated with this file. Each pair is a specific type of data associated with the file. In this example we have an Inode, three extended attributes, a data stream, and a file extent. These are the Keys.

B-Tree Offset = 440 bytes + TOC determined offset within Keys Section.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: File System Values (Inode 0x14 [20])

B-tree Offset (start)	Entry	Offset (from end)	Size (in bytes)	Entry Type	Value and Notes
2691	27	1405	168	Inode	**File Metadata (See Example Slide)**
2671	28	1425	20	Xattr	Flags = 0x0200, Length = 0x1000 Data = 0x28C72C5E0000000096AC181700000000
2610	29	1486	61	Xattr	Flags = 0x0200, Length = 0x3900 Data = "0083;5e2cc6e1;Safari;3F80AE91-AAF8-441F-8030-CFEA67C12747"
2580	30	1516	30	Xattr	Flags = 0x0200, Length = 0x1A00 Data = 0x0A120A1059C45688BCFCFFB4000000000007C9FC108FDC8E8E0F
2576	31	1520	4	Data Stream	0x02000000: 2 Reference Counts (Cloned File)
2552	32	1544	24	File Extent	**File Extent (See Example Slide)**

The values in this key/value pair are stored in the bottom of the B-Tree Node. These values contain the forensic goods that analysts are looking for: The contents of the extended attributes, the file/directory metadata, and file extent details.

B-Tree Offset = 4096 bytes - TOC determined offset within Values Section.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [1]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2691	0	8	parent_id	0x1300000000000000 Parent Inode Number = 19
2699	8	8	private_id	Inode Number = 20 0x1400000000000000
2707	16	8	create_time	0x008A47D31443ED15 = 1579992801000000000 = 2020-01-25 22:53:21 UTC
2715	24	8	mod_time	0x008A47D31443ED15 = 1579992801000000000 = 2020-01-25 22:53:21 UTC
2723	32	8	change_time	0x7B6984933743ED15 = 157999295025410290 7 = 2020-01-25 22:55:50.254103 UTC
2731	40	8	access_time	0x00D08ECE2C43ED15 = 1579992904000000000 = 2020-01-25 22:55:04 UTC

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [2]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2739	48	8	internal_flags	0x1084000000000000 (Cloned + others)
2747	56	4	nchildren or nlink	0x01000000 = 1 link
2751	60	4	default_protection_class	0x00000000
2755	64	4	write_generation_counter	0x03000000
2759	68	4	bsd_flags	0x00000000
2763	72	4	owner	0xF5010000 = 501
2767	76	4	group	0x14000000 = 20
2771	80	2	mode	0xA481 = 1010010010000001 Byte Flip = 1000 000 110 100 100 1000 = 8 (Regular File) 000 = SetUID, SetGID, Sticky bits 110 = 6 (rw-) User Permissions 100 = 4 (r--) Group Permissions 100 = 4 (r--) Other Permissions
2773	82	2	pad1	0x0000
2775	84	8	pad2	0x0000000000000000

168

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

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

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Metadata (Inode 0x14 [20]) [3]

B-tree Offset	Inode Entry Offset	Size (in bytes)	Field	Value and Notes
2783	92	2	xf_num_exts	Number of Extended Fields = 0x0200 = 2
2785	94	2	xf_used_data	Extended Fields Data Used = 0x4000 = 64 bytes
	96	x_field_t = 8 bytes total	Extended Field: x_type (1 byte), x_flags (1 byte), x_size (2 bytes)	
2787	96	4	0x04 = 4, 0x02 (Do Not Copy), 0x1100 = 17 (File Name)	
2791	100	4	0x08 = 8, 0x20 (System Field), 0x2800 = 40 – Data Stream	
2795	104	{17}	File Name	smudge_yoda.jpeg (w/1 padding bytes 0x00), 17 total bytes
2812	120	{40}	Data Stream (Size: First 8 bytes, Allocated: Next 8 bytes)	0x00000000000000 – 7 unused bytes Size: 0x261C020000000000 = 138278 bytes Allocated: 0x0020020000000000 = 139264

The file and directory metadata is stored in the inode key/value pair. The structure contains the same details as we had on HFS+ and other file systems.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

APFS B-Tree: Inode File Extent (Inode 0x14 [20]) [1]

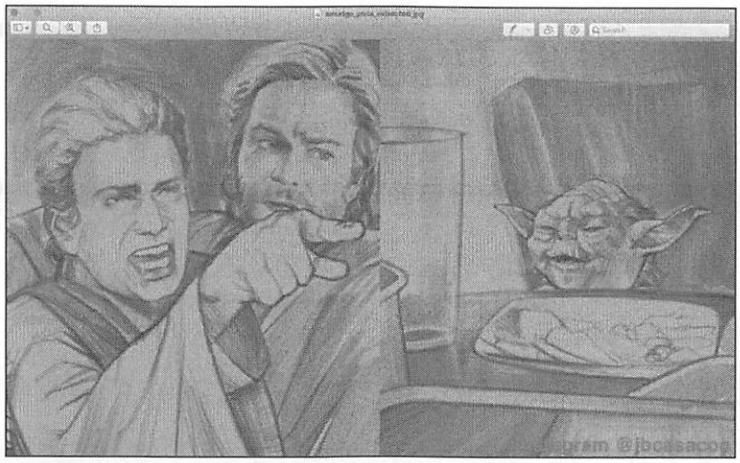
File Extent Value: 0x0020020000000000 6900000000000000 0000000000000000	
File Size (8 bytes)	0x0020020000000000 = 138264 bytes
Physical Block Location (8 Bytes): Physical Block Number from start of container (add 5 (20,480) blocks for start of disk) (# * 4096) + 20,480 = start of file location in bytes	0x6900000000000000 = 105 = Physical Block Number (105 * 4096) + 20,480 = 450560 bytes
Crypto_ID (8 Bytes)	Encryption Key: 0x0000000000000000

The Inode File Extent information can be used to find the file in the disk image.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf

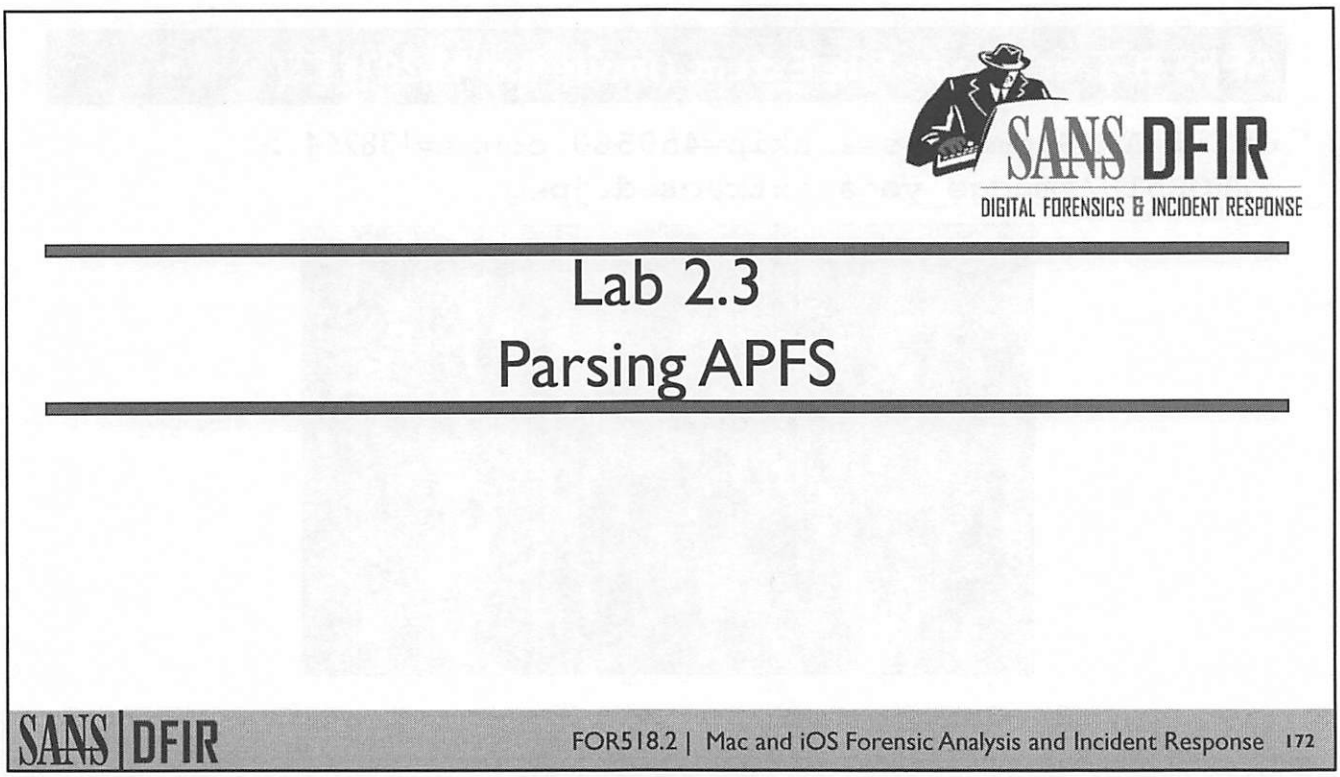
APFS B-Tree: Inode File Extent (Inode 0x14 [20]) [2]

```
dd if=APFS.dmg ibs=1 skip=450560 count=138264 >
~/FOR518/smudge_yoda_extracted.jpeg
```



The Inode File Extent information can be used to find the file in the disk image. The command above is used to extract the file that we have been viewing in this example/demo.

Reference:
Apple File System Reference: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf



This page intentionally left blank.

Annex 1
HFS+

This page intentionally left blank.

Volume Header and Example

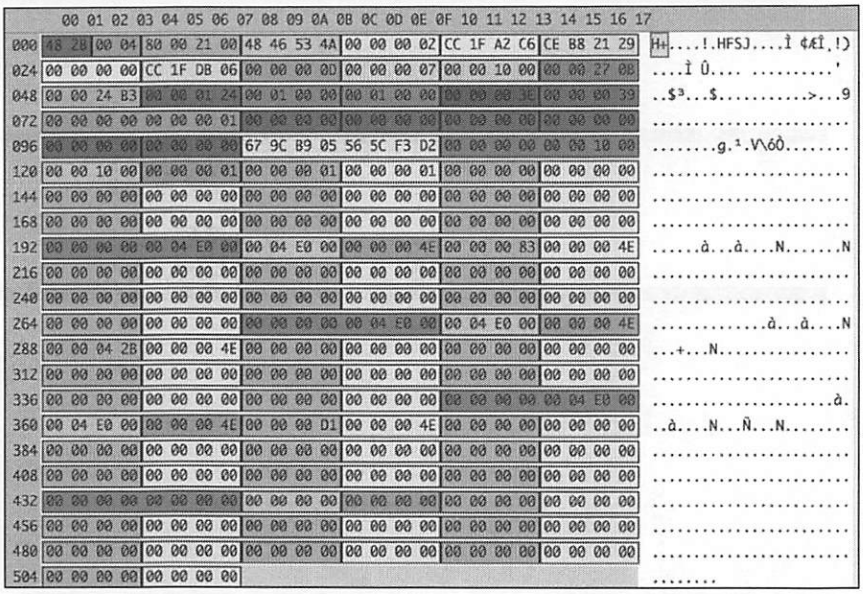
Contains data such as:

- Signature (H+)
- HFS+ Version
- Volume Creation (Local Time), Modified (GMT), Backup (GMT/Unused), and Checked (GMT) Dates
- File and 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



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".

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 is 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 and Location
192	80	Extents File Size and Location
272	80	Catalog File Size and Location
352	80	Attributes File Size and Location
432	80	Startup File Size and 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

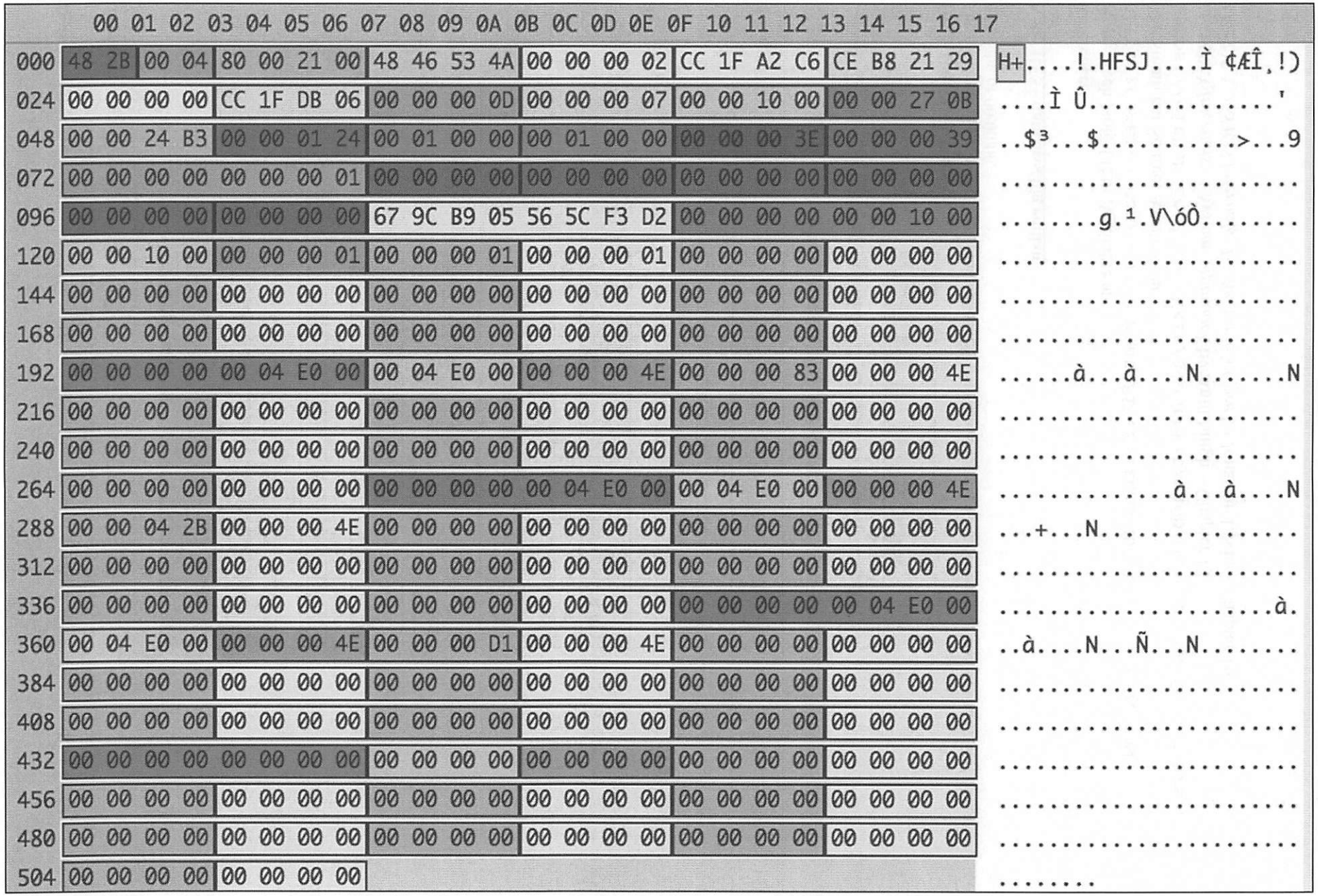
HFS+ Dates and Times

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

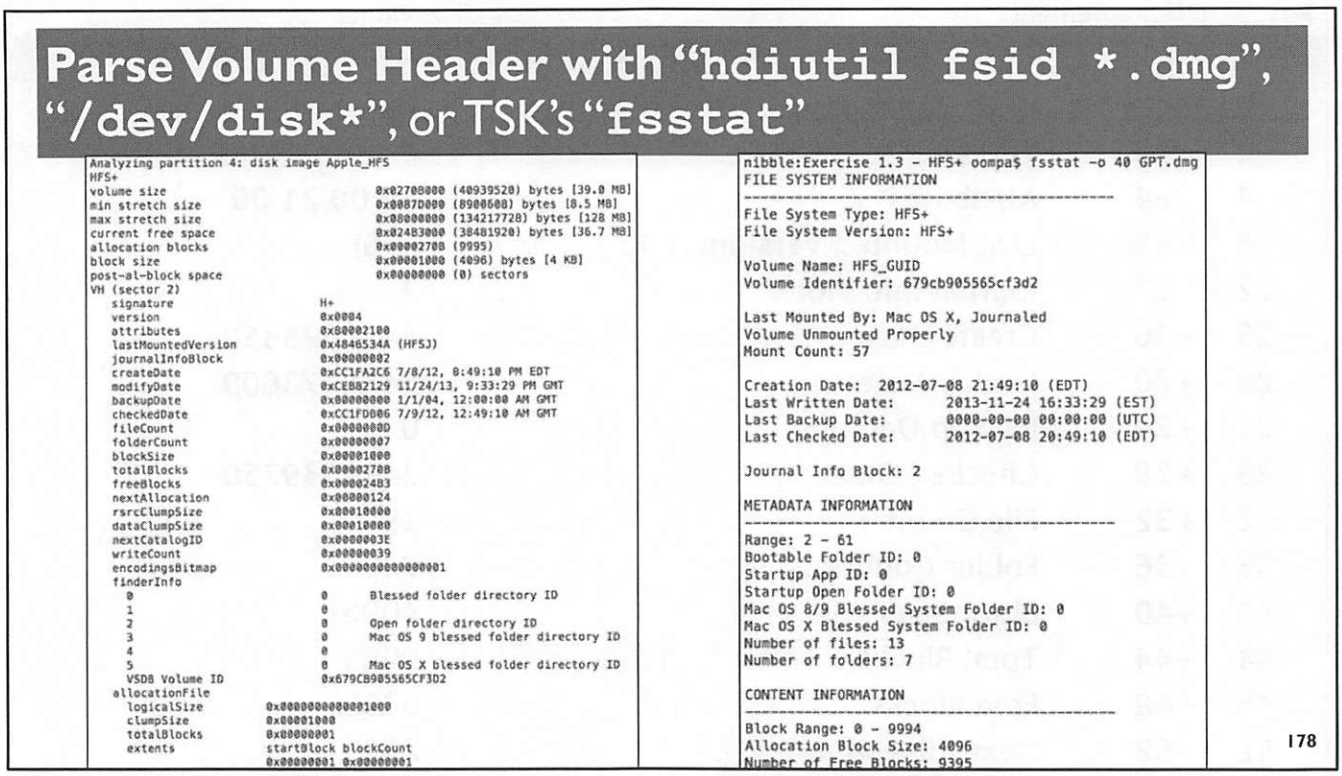
References:

Apple Tech Note 1150: Available at
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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	3E	00	00	00	39
072	00	00	00	00	00	00	00	01	00	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	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	00	4E	00	00	00	83	00	00	00	4E
216	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	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	04	E0	00	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	04	E0	00
360	00	04	E0	00	00	00	00	4E	00	00	00	D1	00	00	00	4E	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																

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]	



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; the rest can be seen on the next page.

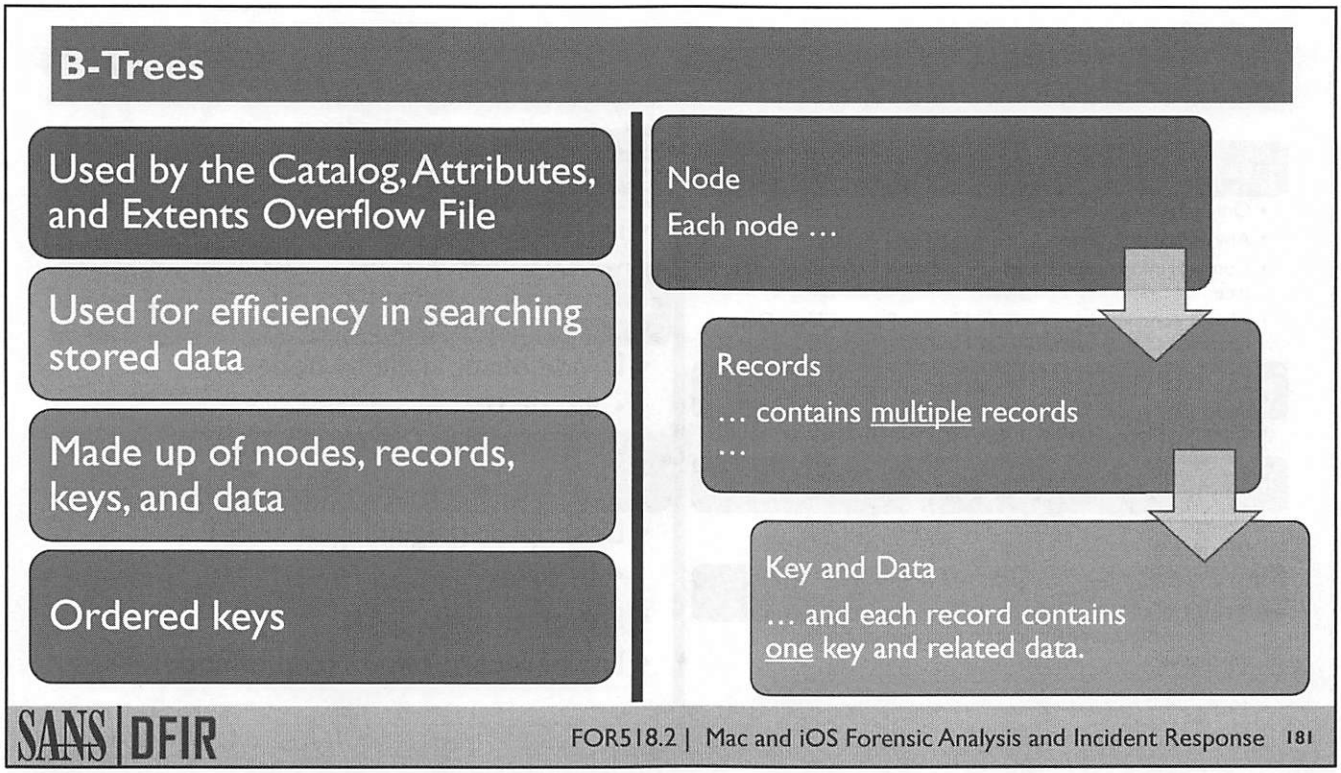
Another way we can view some of the volume header information is to use The Sleuth Kit `fsstat` command. While this does not contain the size and location of the HFS+ special files, it does give us a concise view of the volume, including timestamps, volume name, format, and file/folder information.

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

References:
`hdiutil` Man Page
`fsstat` 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
                          0x0000042B 0x0000004E
   attributesFile
     logicalSize          0x000000000004E000
     clumpSize            0x0004E000
     totalBlocks          0x0000004E
     extents              startBlock blockCount
                          0x000000D1 0x0000004E
   startupFile
     logicalSize          0x0000000000000000
     clumpSize            0x00000000
     totalBlocks          0x00000000
     extents              startBlock blockCount
```

```
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
```



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

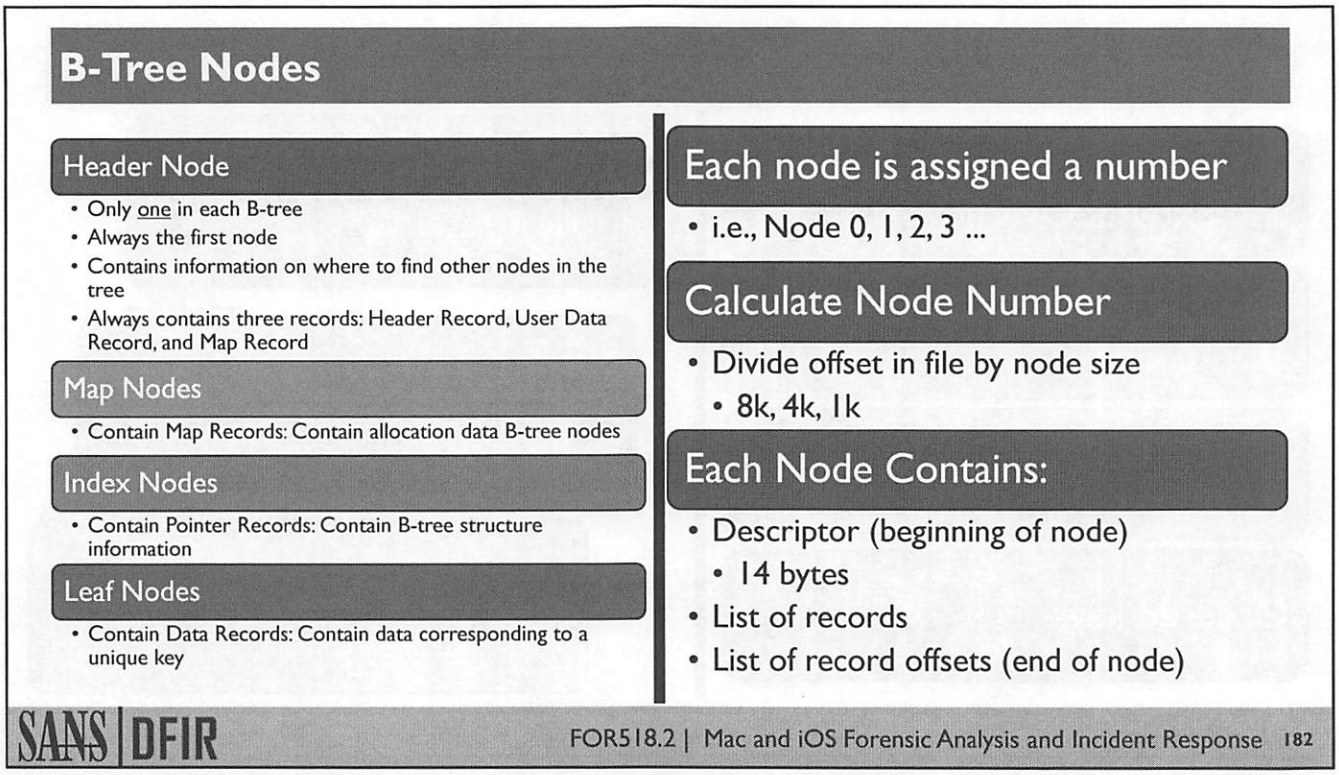
- Catalog File
- Attributes File
- Extents Overflow File

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 it takes time and patience to grasp the structure of B-trees and their purpose in the HFS+ file system.

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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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



A B-tree uses four types of nodes:

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

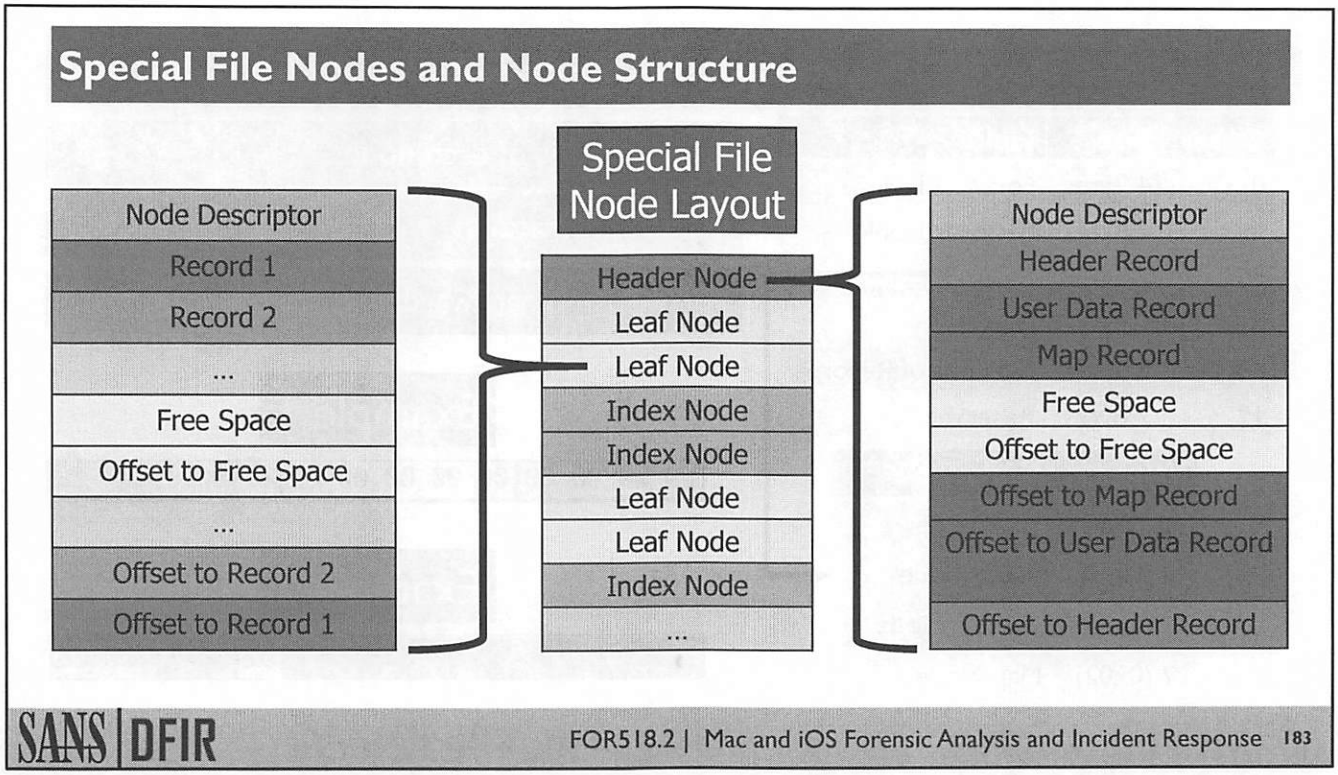
There is only one **Header Node** in each B-tree. There is **only one** Header Node for each of the Catalog, Attributes, and Extents Overflow files. The Header Node is always the first node in the B-tree; it contains information on how to find other nodes in the B-tree. **Map Nodes** contain **Map Records**.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

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

References:
Apple Tech Note 1150: Available at
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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

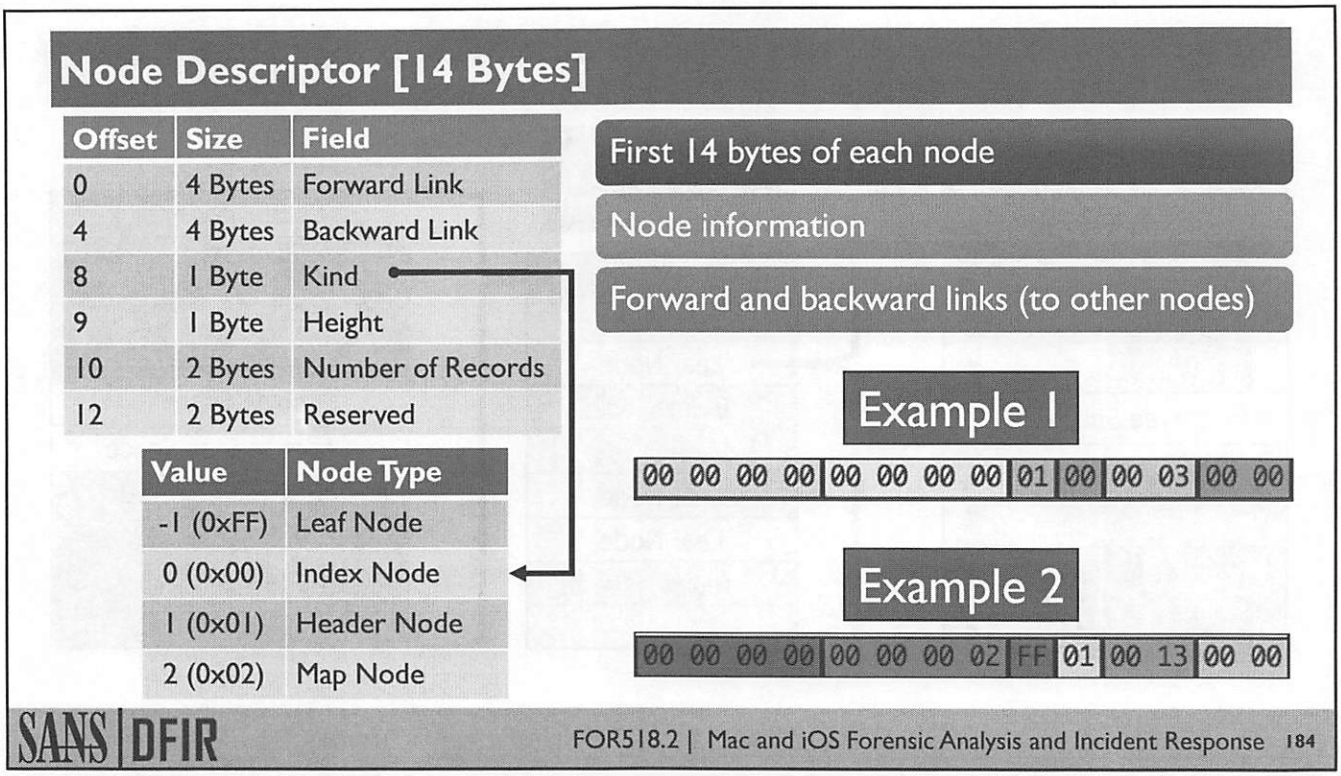


Each special file using a B-tree (Catalog, Attributes, Extents Overflow) will be comprised of ONE Header Node and multiple Leaf, Index, and possibly Map Nodes. Each node size is determined by data found in the Header Node (Node Size); this may be 4,096 bytes for each node, for example.

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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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 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 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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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

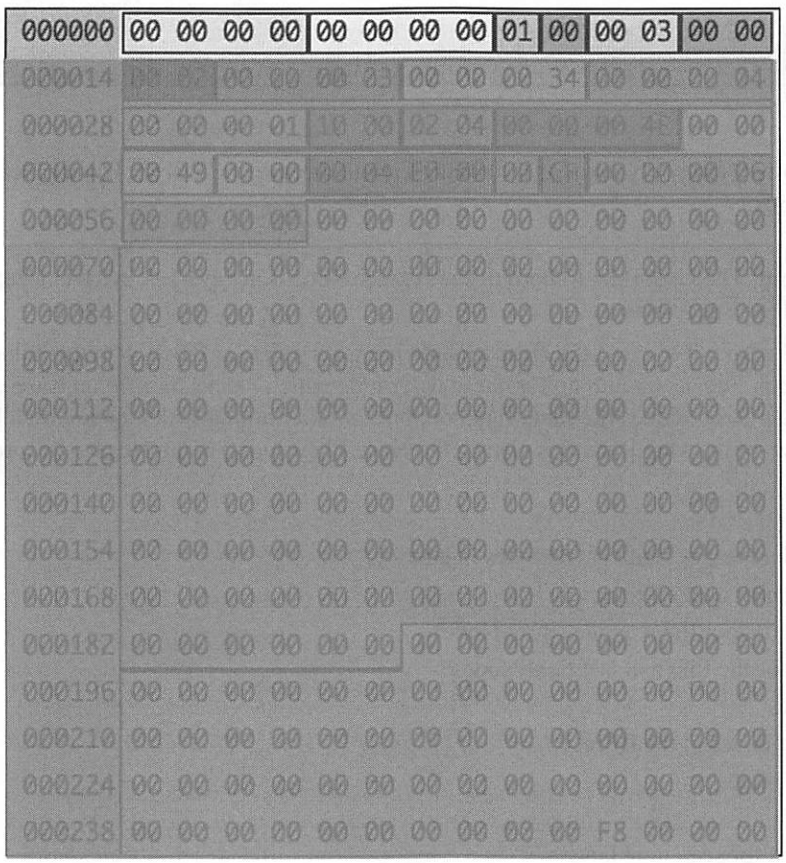
Example 1:

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

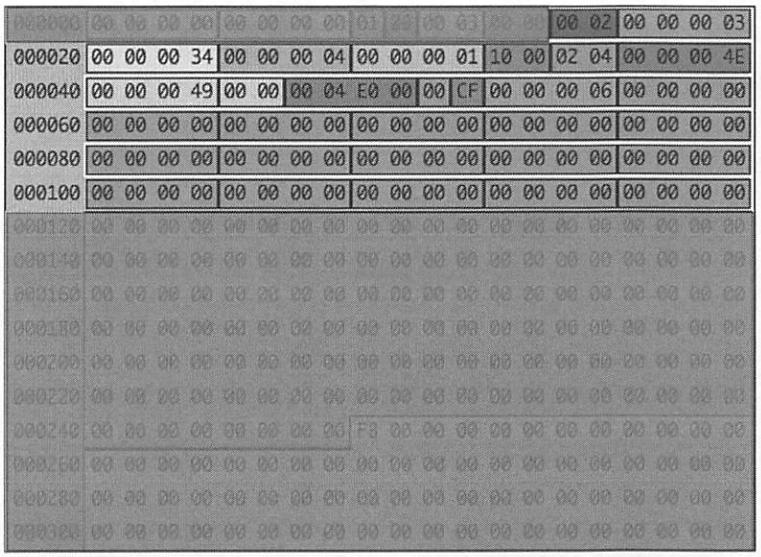
Example 2:

- 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



Header Record Example



Offset	Size	Field
0	2 Bytes	Tree Depth
2	4 Bytes	Root Node
6	4 Bytes	Leaf Records
10	4 Bytes	First Leaf Node
14	4 Bytes	Last Leaf Node
18	2 Bytes	Node Size
20	2 Bytes	Max Key Length
22	4 Bytes	Total Nodes
26	4 Bytes	Free Nodes
30	2 Bytes	Reserved
32	4 Bytes	Clump Size
36	1 Byte	B-tree Type
37	1 Bytes	Key Compare Type
38	4 Bytes	Attributes
42	4 Bytes	Reserved [16]: 64 bytes

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: The User Data Record of the Header Node is always 128 bytes in size; however, it is unused in the HFS+ special files.
- Map Record: 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 determining 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.

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

The Header Record contains data such as:

- 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.

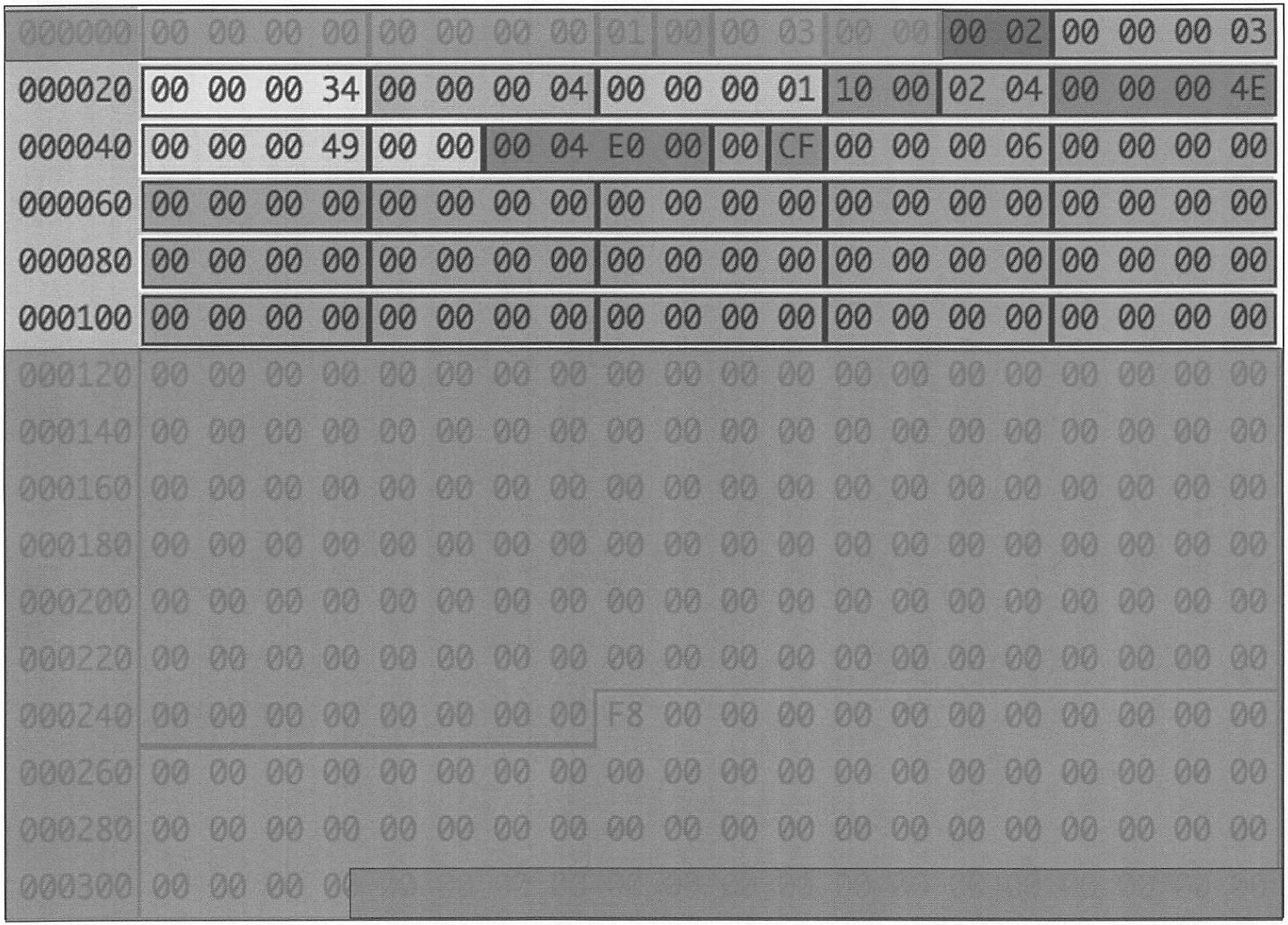
Note: HFS+ data uses big endian

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]

References:

Apple Tech Note 1150: Available at
http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html
The Sleuth Kit Source: Available at
https://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

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 1-byte padding before and/or after Data to align on 2-byte boundary to create an even number of bytes

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

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

Example 1 (Node 4, from previous example):

- Forward Link: `0x00000002` = 2
- Backward Link: `0x00000005` = 5
- Kind: `0xFF` = 0 (Leaf Node)
- Height: `0x01` = 1
- Number of Records: `0x000A` = 10 Records
- Reserved: `0x0000`

Record 1:

- Key Length: `0x0010` = 16 bytes
- Key: (Parent Catalog Node ID (CNID)) - `0x00000002` = 2
- Data:
  - Data Size: `0x0005` = 5 (10 bytes Unicode)
  - Data: `0x0053007400750066` = "`Stuff`"
- *... Continue on to Catalog File/Folder Records ...*

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

Note: HFS+ data uses big endian

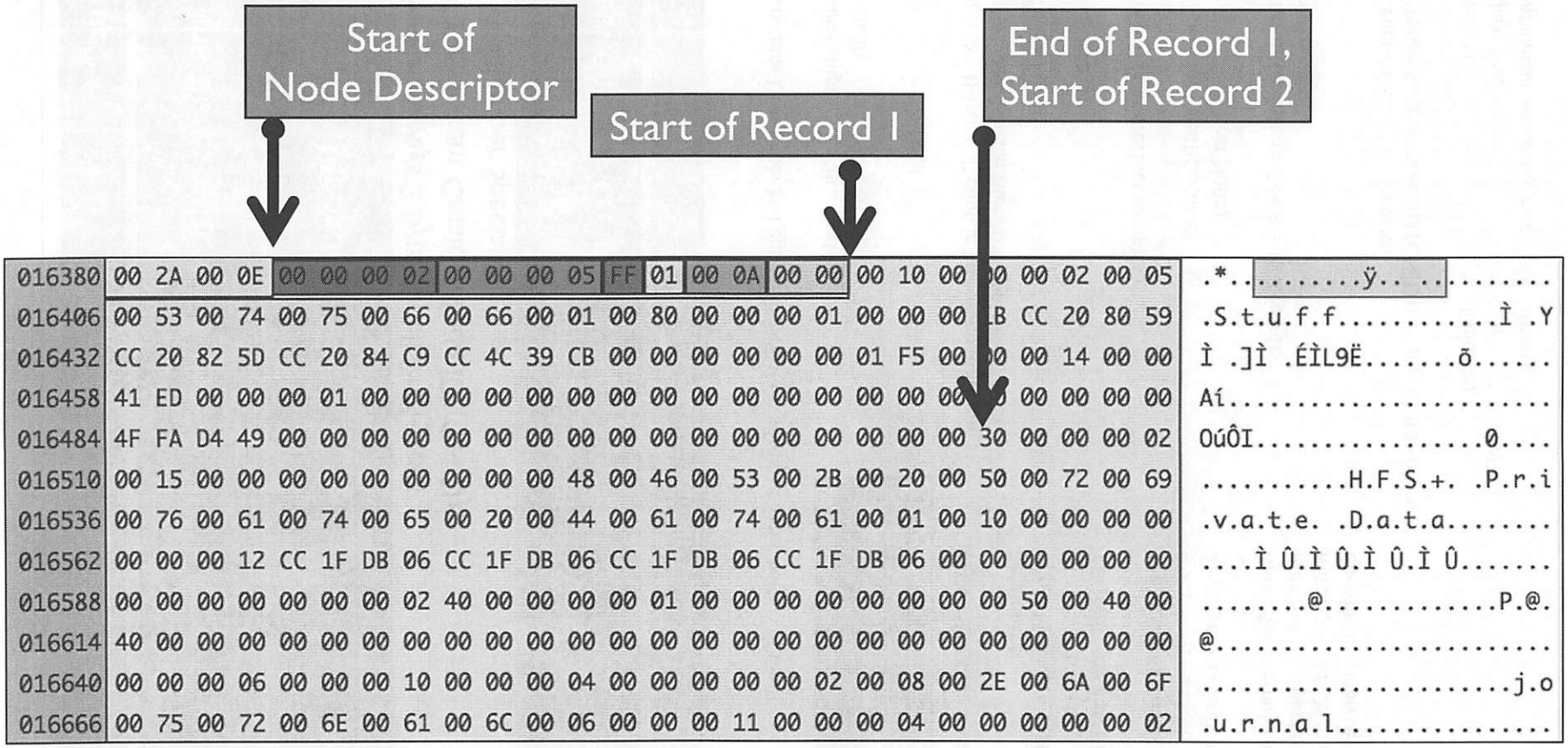
References:

Apple Tech Note 1150: available at `http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`

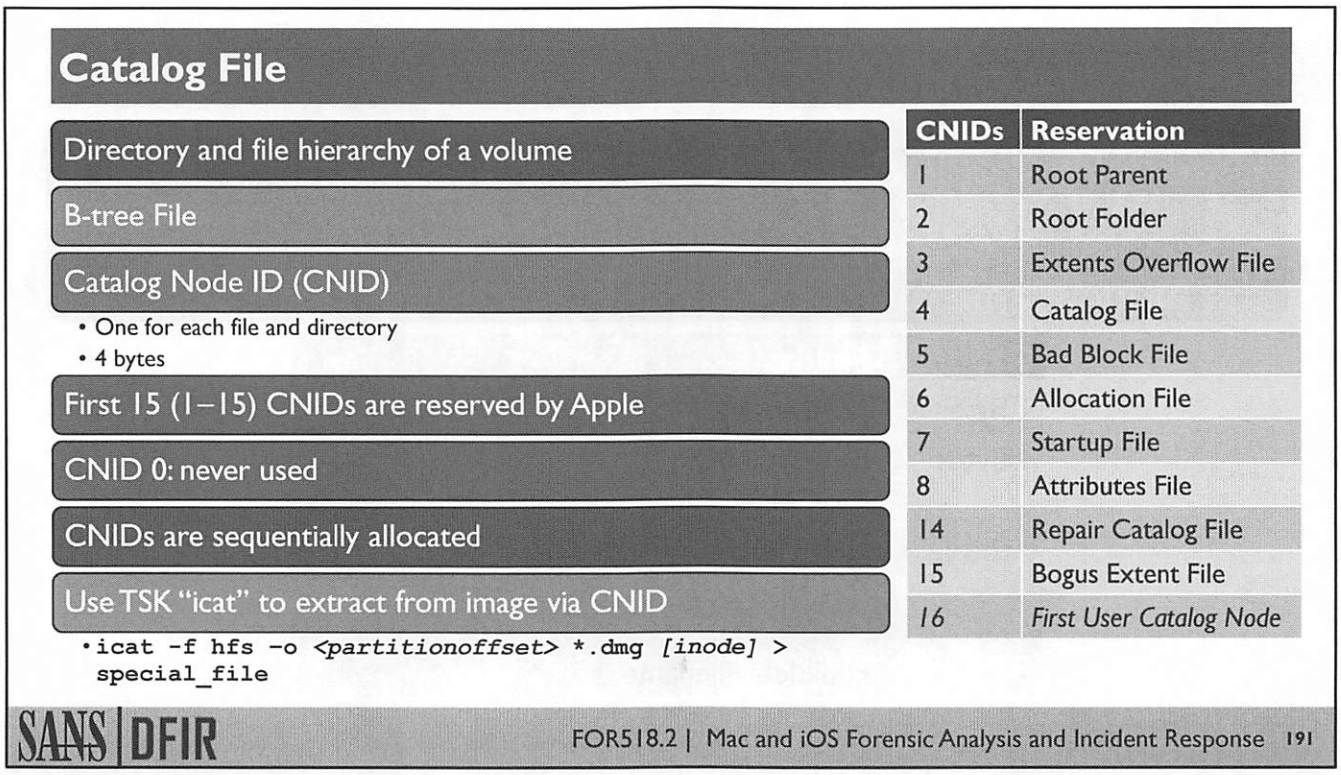
The Sleuth Kit Source: available at `https://github.com/sleuthkit/sleuthkit/blob/master/tsk/fs/tsk_hfs.h`

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



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	1B	CC	20	80	59
016432	CC	20	82	5D	CC	20	84	C9	CC	4C	39	CB	00	00	00	00	00	00	01	F5	00	00	00	14	00	00
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
016484	4F	FA	D4	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	30	00	00	02
016510	00	15	00	00	00	00	00	00	00	00	00	48	00	46	00	53	00	2B	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	61	00	01	00	10	00	00	00	00
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	00
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
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	00
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
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	02



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.

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.

Each of the HFS+ special files has their own "inode", or Catalog Node ID (CNID). 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 macbook.dmg 6 > allocation_file
```

References:

Apple Tech Note 1150: Available at
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`

The Sleuth Kit Source: Available at
`https://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

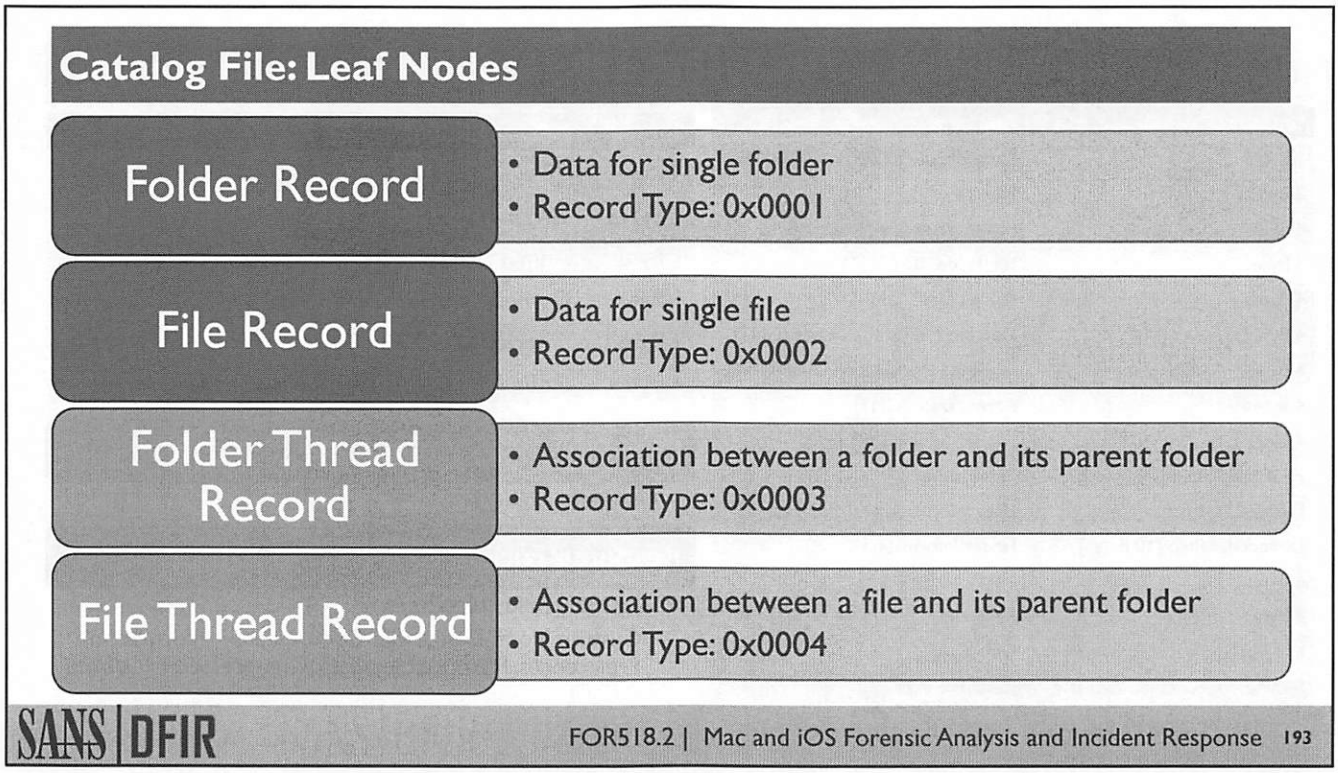
In searches, the CNID of the parent is compared first, then the Node Name

Size	Field
2 Bytes	Key Length
4 Bytes	Parent CNID (or CNID of file/folder of Thread Record)
Variable	Node Name (Unicode) • Empty string in Thread Records (0x0000) • Folder/filename

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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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: File/Folder Records [248 (+ 160 Bytes—File Only)]

Size	Field
2 Bytes	Record Type (0x0001 or 0x0002)
2 Bytes	Flags
4 Bytes	Folder Valence or Reserved
4 Bytes	File/Folder ID (CNID)
4 Bytes	Create Date (GMT)
4 Bytes	Content Modification Date (GMT)
4 Bytes	Attribute Modification Date (GMT)
4 Bytes	Access Date (GMT)
4 Bytes	Backup Date (GMT)
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

Data Fork

- Generally holds data

Resource Fork

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

SANS | DFIR FOR518.2 | Mac and iOS Forensic Analysis and Incident Response 194

The File and Folder 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, while a Folder Record will have 0x0001.

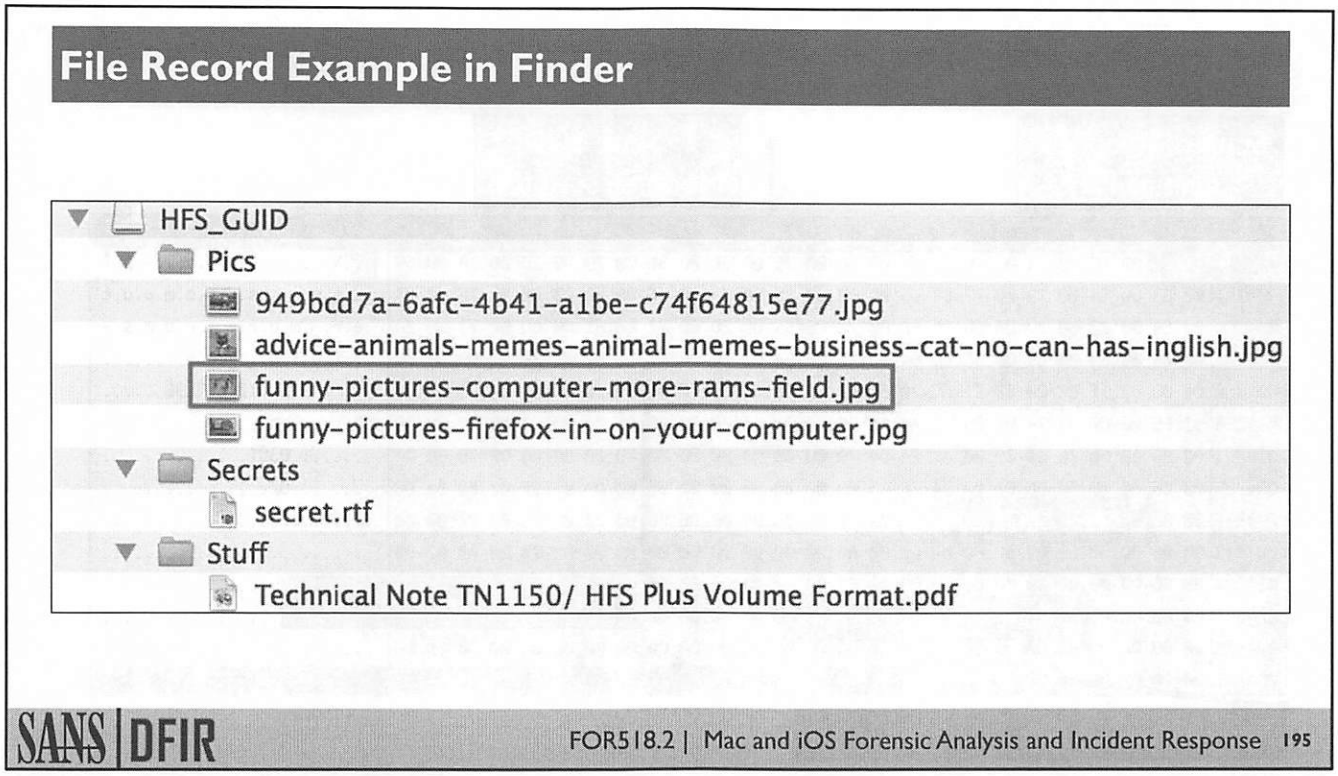
Each File/Folder Record contains five dates (HFS+ Timestamp 1/1/1904 00:00:00: in GMT):

- 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

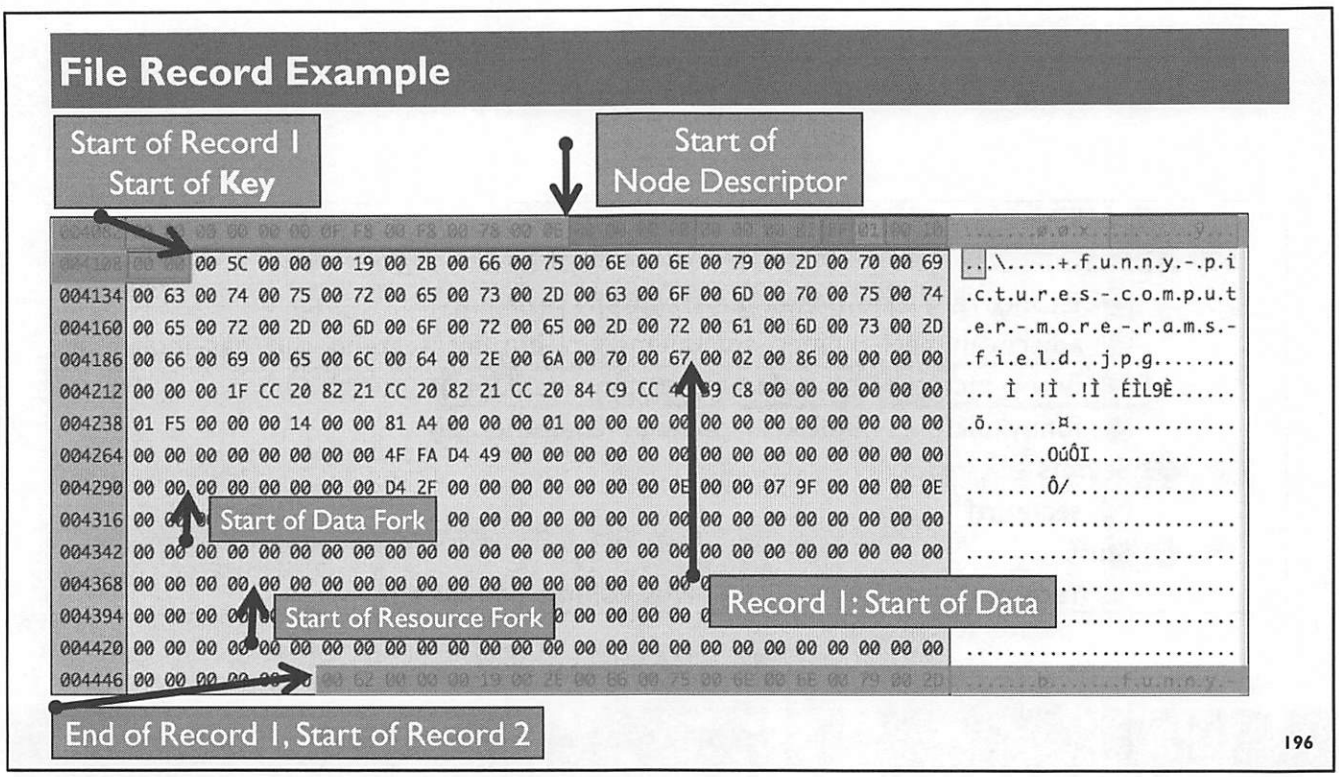
The table to the right contains the format for HFS+ `permissions` for the file labeled as HFSPlusBSDInfo in the spec. The highlighted sections are the main differences between File and Folder Records. Folder Records will have a valence (the number of items in this directory), while this is reserved for File Records. File Records will have two additional fields, Data and Resource Forks of 80 bytes each, while Folder Records will not have these fields.

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 file data and metadata.

References:
Apple Tech Note 1150: Available at
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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



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—`0x00660075006E006E0079002D00700069006300740075007200650073002D0063006F006D00700075007400650072002D006D006F00720065002D00720061006D0073002D00660069006500 6C0064002E006A00700067` = "`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`

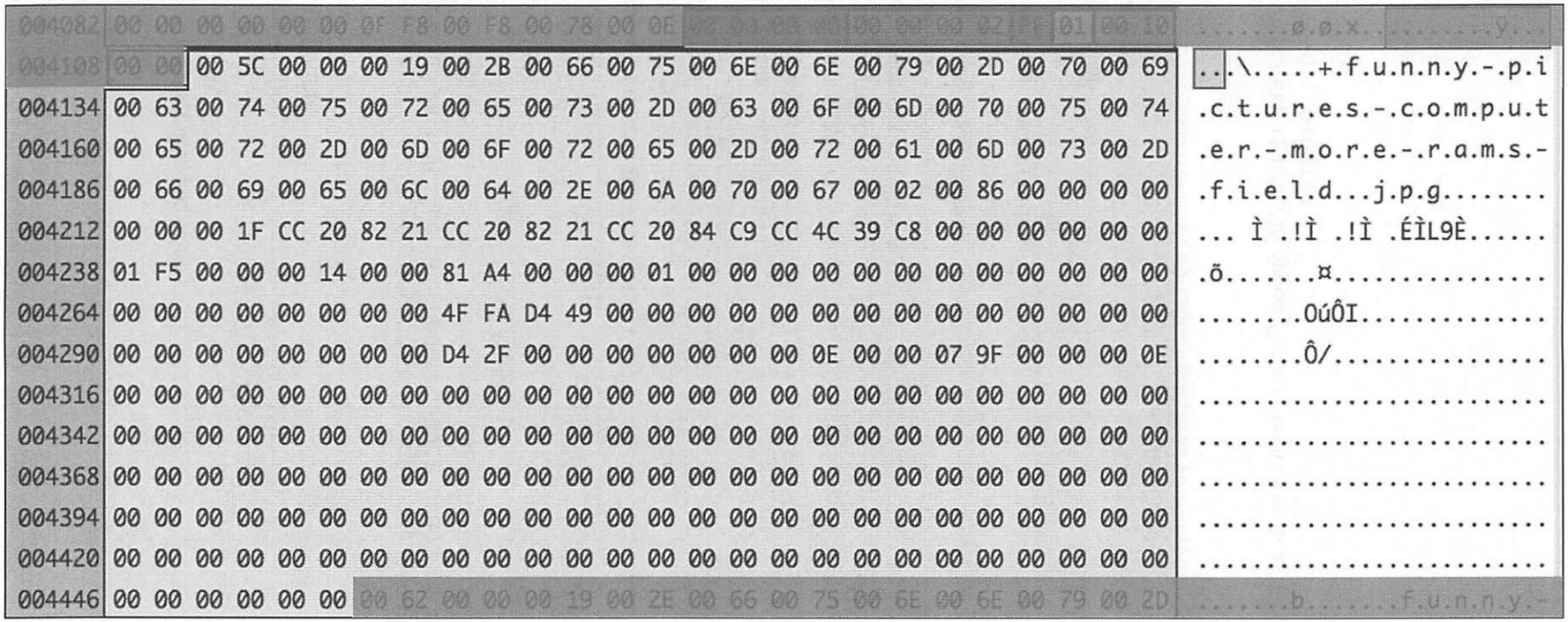
... continued on next page ...

- Permissions: 0x000001F50000001400081A400000001
  - Owner ID: 0x000001F5 = 501
  - Group ID: 0x00000014 = 20
  - Admin Flags: 0x00
  - Owner Flags: 0x00
  - File Mode: 0x81A4 (See decoding below)
  - Link Count: 0x00000001
- User Information: 0x00000000000000000000000000000000
- Finder Information: 0x000000004FFAD4490000000000000000
- Text Encoding: 0x00000000
- Reserved: 0x00000000
- Data Fork:
  0x000000000000D42F000000000000000E0000079F0000000E000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000000000000
  000000000000
  - Logical Size: 0x000000000000D42F = 54319 bytes
  - Clump Size: 0x00000000
  - Total Blocks:0x0000000E = 14
  - Extent 1: Start Block – 0x0000079F = 1951
  - Extent 1: Block Count – 0x0000000E = 14
  - Extent 2: Start Block – 0x00000000
  - Extent 2: Block Count – 0x00000000
  - Extent 3: Start Block – 0x00000000
  - Extent 3: Block Count – 0x00000000
  - Extent 4: Start Block – 0x00000000
  - Extent 4: Block Count – 0x00000000
  - Extent 5: Start Block – 0x00000000
  - Extent 5: Block Count – 0x00000000
  - Extent 6: Start Block – 0x00000000
  - Extent 6: Block Count – 0x00000000
  - Extent 7: Start Block – 0x00000000
  - Extent 7: Block Count – 0x00000000
  - Extent 8: Start Block – 0x00000000
  - Extent 8: Block Count – 0x00000000
- Resource Fork – (Empty)
  0x000000000000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000000000000
  000000000000

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
```

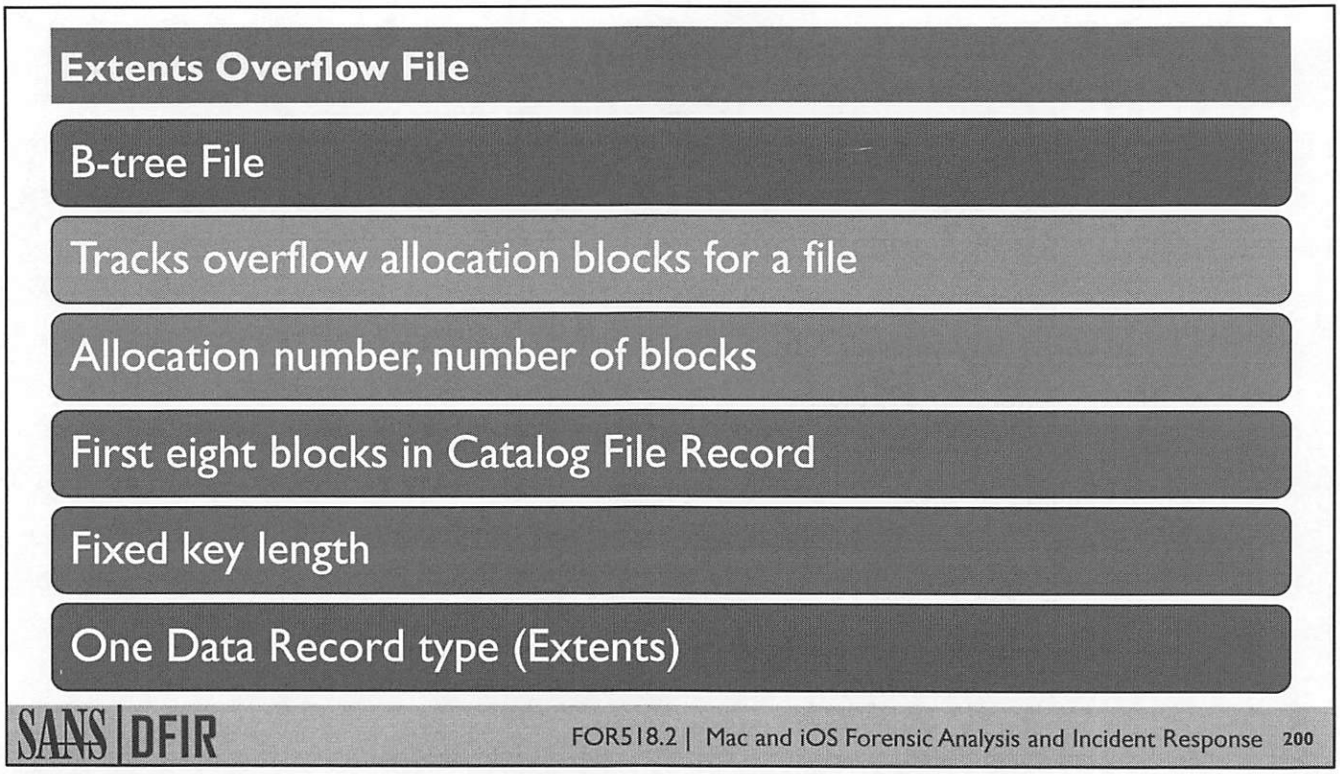


004082	00	00	00	00	00	00	0F	F8	00	F8	00	78	00	0E	[illegible]						01	00	10	.....ø.ø.x.....y.
004108	00	00	00	5C	00	00	00	19	00	2B	00	66	00	75	00	6E	00	6E	00	79	00	2D	00	70	00	69	..\.....+.f.u.n.n.y.-.p.i
004134	00	63	00	74	00	75	00	72	00	65	00	73	00	2D	00	63	00	6F	00	6D	00	70	00	75	00	74	.c.t.u.r.e.s.-.c.o.m.p.u.t
004160	00	65	00	72	00	2D	00	6D	00	6F	00	72	00	65	00	2D	00	72	00	61	00	6D	00	73	00	2D	.e.r.-.m.o.r.e.-.r.a.m.s.-
004186	00	66	00	69	00	65	00	6C	00	64	00	2E	00	6A	00	70	00	67	00	02	00	86	00	00	00	00	.f.i.e.l.d...j.p.g.....
004212	00	00	00	1F	CC	20	82	21	CC	20	82	21	CC	20	84	C9	CC	4C	39	C8	00	00	00	00	00	00	... Ì .!Ì .!Ì .ÉÌL9È.....
004238	01	F5	00	00	00	14	00	00	81	A4	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	.õ.....¤.....
004264	00	00	00	00	00	00	00	00	4F	FA	D4	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....OúÔI.....
004290	00	00	00	00	00	00	00	00	D4	2F	00	00	00	00	00	00	00	0E	00	00	07	9F	00	00	00	0E	.....Ô/.....
004316	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	.....
004342	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	.....
004368	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	.....
004394	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	.....
004420	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	.....
004446	00	00	00	00	00	00	00	62	00	00	00	19	00	2E	00	66	00	75	00	6E	00	6E	00	79	00	2D	.....b.....f.u.n.n.y.-

File Record Example: Sanity Check

```
nibble:Exercise 1.3 - HFS+ oompa$ 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 12:42:09 (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
```


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



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
`http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html`
The Sleuth Kit Source: Available at
`https://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: Key and Data

B-tree File

Variable Key Length

Three Record Types

- **Inline Attributes: 0x00000010**
- Fork Data Attributes: 0x00000020
- Extension Attributes: 0x00000030

Key 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

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

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
- Attribute 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

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 throughout the course are extended attributes, which are where you will find these types 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
http://dubeiko.com/development/FileSystems/HFSPLUS/tn1150.html

The Sleuth Kit Source: Available at
https://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

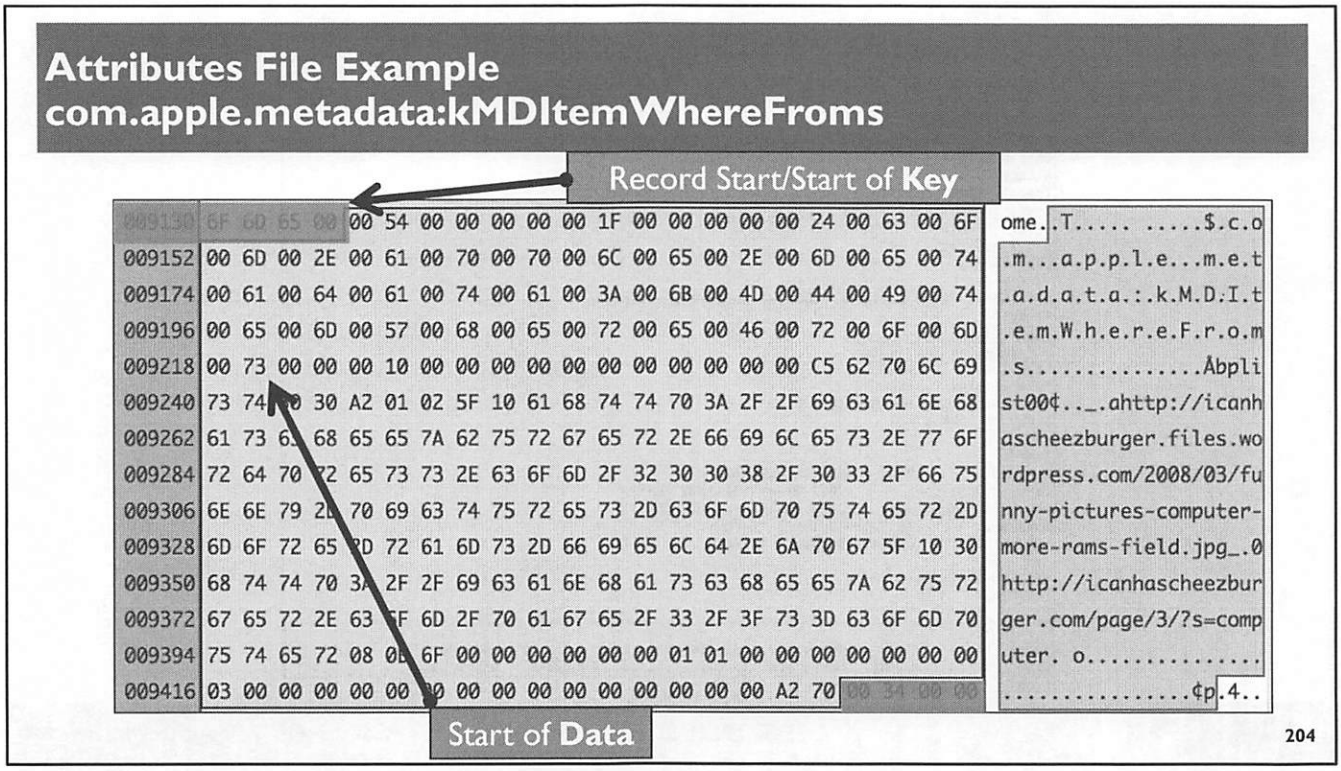
Attributes File Examples: Two Extended Attributes

```
nibble:Exercise 1.3 - HFS+ oompa$ 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 12:42:09 (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
```


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`



Attributes File Example: (**Note: HFS+ data uses big endian**)

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

Attribute 1:

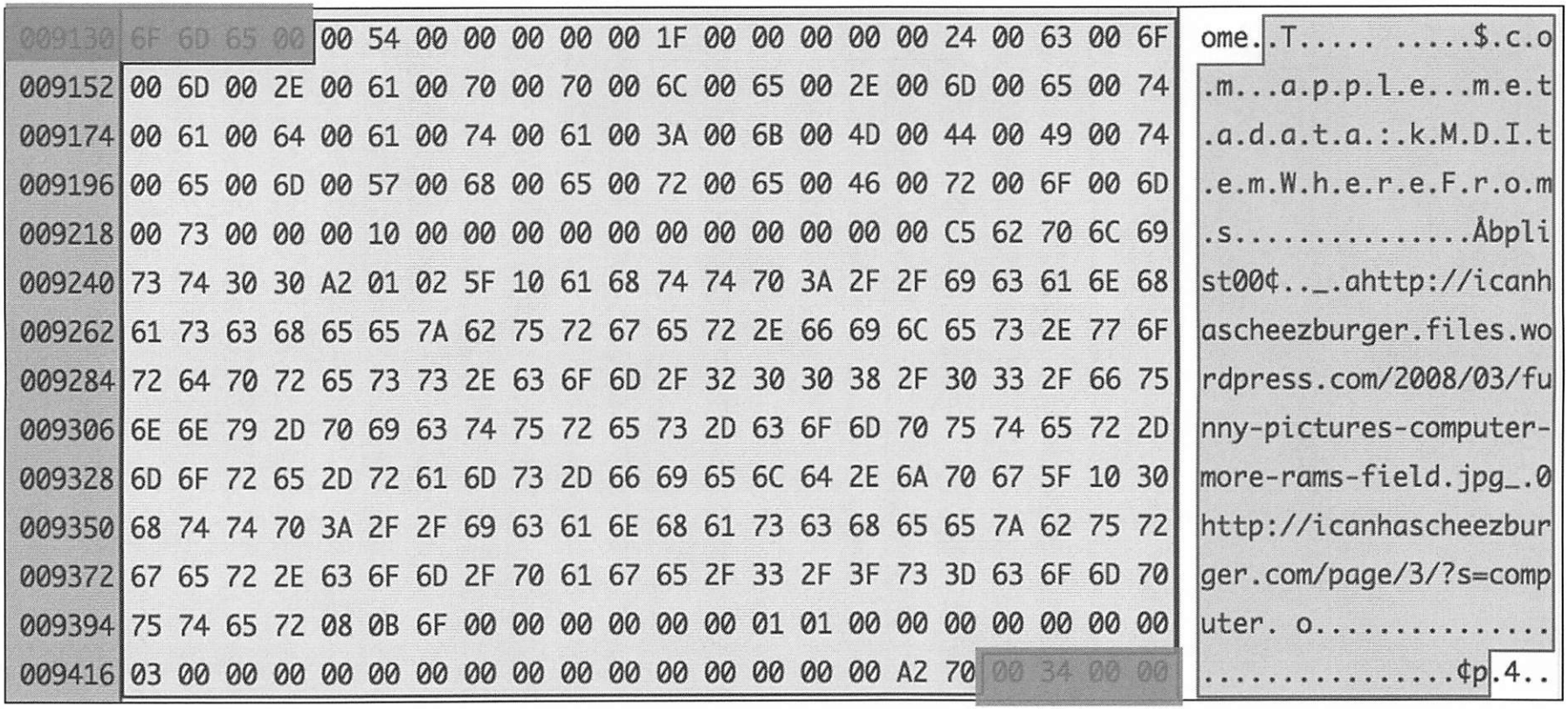
KEY:

- Key Length: `0x0054` = 84 bytes
- Pad: `0x0000`
- File ID (CNID): `0x0000001F` = 31
- Start Block: `0x00000000` = 0
- Attribute Name Length: `0x0024` = 36 bytes (72 bytes Unicode)
- Attribute Name: "`com.apple.metadata:kMDItemWhereFroms`"

DATA:

- Record Type: `0x00000010` – Inline Attribute
- Reserved: `0x0000000000000000`
- Attribute Size: `0x000000C5` = 197 bytes
- Attribute Data: Binary Property List containing information where this picture was downloaded from.

```
0x62706C6973743030A201025F1061687474703A2F2F6963616E686173636865657A6275
726765722E66696C65732E776F726470726573732E636F6D2F323030382F30332F66756E
6E792D7069637475726573732D636F6D70757465722D6D6F72652D72616D732D6669656C64
2E6A70675F1030687474703A2F2F6963616E686173636865657A6275726765722E636F6D
2F706167652F332F3F733D636F6D7075746572080B6F000000000000010100000000000000
0003000000000000000000000000000000A270
```



009130	6F	6D	65	00	00	54	00	00	00	00	00	1F	00	00	00	00	00	24	00	63	00	6F	ome.|.T..... .....$.c.o
009152	00	6D	00	2E	00	61	00	70	00	70	00	6C	00	65	00	2E	00	6D	00	65	00	74	.m...a.p.p.l.e...m.e.t
009174	00	61	00	64	00	61	00	74	00	61	00	3A	00	6B	00	4D	00	44	00	49	00	74	.a.d.a.t.a.:.k.M.D.I.t
009196	00	65	00	6D	00	57	00	68	00	65	00	72	00	65	00	46	00	72	00	6F	00	6D	.e.m.W.h.e.r.e.F.r.o.m
009218	00	73	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	00	C5	62	70	6C	69	.s..............Åbpli
009240	73	74	30	30	A2	01	02	5F	10	61	68	74	74	70	3A	2F	2F	69	63	61	6E	68	st00¢.._.ahttp://icanh
009262	61	73	63	68	65	65	7A	62	75	72	67	65	72	2E	66	69	6C	65	73	2E	77	6F	ascheezburger.files.wo
009284	72	64	70	72	65	73	73	2E	63	6F	6D	2F	32	30	30	38	2F	30	33	2F	66	75	rdpress.com/2008/03/fu
009306	6E	6E	79	2D	70	69	63	74	75	72	65	73	2D	63	6F	6D	70	75	74	65	72	2D	nny-pictures-computer-
009328	6D	6F	72	65	2D	72	61	6D	73	2D	66	69	65	6C	64	2E	6A	70	67	5F	10	30	more-rams-field.jpg_.0
009350	68	74	74	70	3A	2F	2F	69	63	61	6E	68	61	73	63	68	65	65	7A	62	75	72	http://icanhascheezbur
009372	67	65	72	2E	63	6F	6D	2F	70	61	67	65	2F	33	2F	3F	73	3D	63	6F	6D	70	ger.com/page/3/?s=comp
009394	75	74	65	72	08	0B	6F	00	00	00	00	00	00	00	01	01	00	00	00	00	00	00	00	uter. o..............
009416	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	A2	70	00	34	00	00	..............¢p.4..

"As usual, SANS courses pay for themselves by Day 2. By Day 3, you are itching to get back to the office to use what you've learned."
Ken Evans, Hewlett Packard Enterprise - Digital Investigation Services

SANS Programs
sans.org/programs

GIAC Certifications
Graduate Degree Programs
NetWars & CyberCity Ranges
Cyber Guardian
Security Awareness Training
CyberTalent Management
Group/Enterprise Purchase Arrangements
DoDD 8140
Community of Interest for NetSec
Cybersecurity Innovation Awards

Search SANSInstitute

SANS Free Resources
sans.org/security-resources

- E-Newsletters
  NewsBites: Bi-weekly digest of top news
  OUCH!: Monthly security awareness newsletter
  @RISK: Weekly summary of threats & mitigations
- Internet Storm Center
- CIS Critical Security Controls
- Blogs
- Security Posters
- Webcasts
- InfoSec Reading Room
- Top 25 Software Errors
- Security Policies
- Intrusion Detection FAQ
- Tip of the Day
- 20 Coolest Careers
- Security Glossary

SANS Institute
8120 Woodmont Avenue | Suite 310
Bethesda, MD 20814
301.654.SANS(7267)
info@sans.org