MAGNET AXIOM AND macOS/APFS

A GUIDE TO APPLE'S LATEST FILE SYSTEM

MAGNET FORENSICS®

MAGNET AXIOM AND macOS/APFS

EXECUTIVE SUMMARY

The Apple File System (APFS) is the latest file system to come from Apple, Inc. for their family of Macintosh computers, as well as iPhone, iPad, Apple TV, and Apple Watch. It supersedes the aging Hierarchal File System Plus (HFS+), adding many significant new features found in other modern file systems such as ZFS or XFS, including Copy-on-Write (CoW), encryption, and cloning.

The purpose of this paper is to provide a high-level overview of some of the more prominent APFS features of interest to digital forensic examiners working with APFS-aware tools such as Magnet AXIOM. HFS+ is referenced where appropriate to illustrate the differences found in the two file systems. To keep the exploration reasonably brief and focused on APFS, it is assumed the target audience has a fundamental understanding of HFS+ and its associated structures, i.e. volume header, allocation file, catalog file, etc. Where APFS structures and functionality overlap or duplicate HFS+, explanations may only include common definitions when they are appropriate for clarity of discussion. Otherwise, it appears APFS has more in common with other UNIX-like file systems than it does with HFS.

APPLE FILE SYSTEM (APFS) OVERVIEW

The top level APFS structure is the container and is described in the GUID Partition Table (GPT) for Apple Extensible Firmware Interface (EFI) disks. Containers are subdivided into one or more volumes. Information regarding volumes is found in the container's metadata. The container possesses the following primary structures:

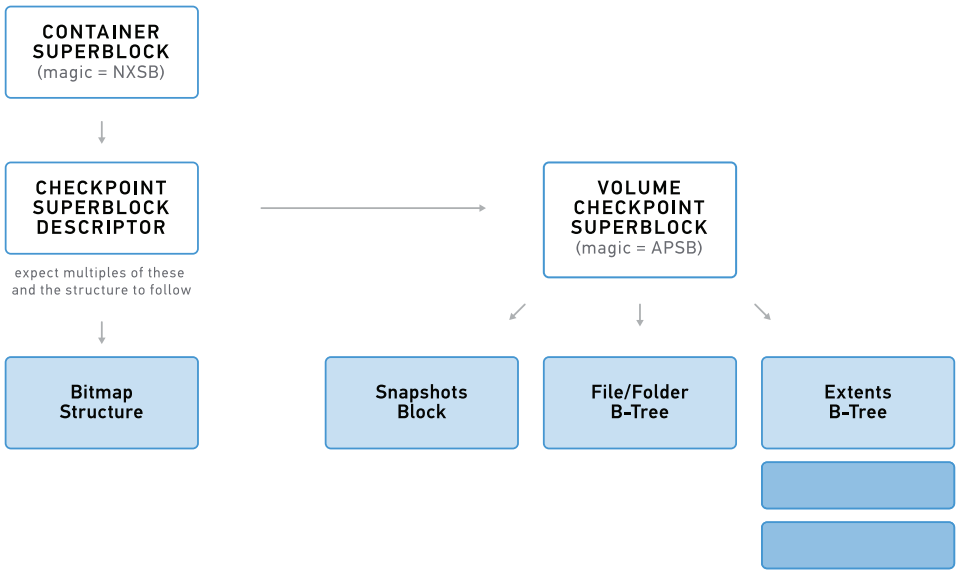

Figure 1- Simplified APFS Diagram

Container Superblock = Main superblock with metadata information about the container (blocks and checkpoints).

Checkpoint Superblock Descriptor = Metadata information about a specific Checkpoint Superblock. The container has its own Checkpoint Superblock Descriptor.

Bitmap Structure = Allocation status of blocks (Similar to Allocation File in HFS+).

Volume Superblock = Volume information.

File/Folder B-Tree = File records (similar to Catalog File in HFS+), including data location by offset and length.

Extents B-Tree = Records file content by offset and length similar to File/Folder B-tree, but part of Snapshots.

Snapshots = Volume state at a specific point in time and user-initiated; separate from checkpoints.

Checkpoint Superblock = A historical state of the container. Up to 100 of these. Similar to Time Machine in that disk states can be preserved for rollback. Created each time the file system data is written to disk. Current state should be the last Checkpoint.

APFS volumes in a container all share space within that container. This is different from typical partition schemes where each partition or logical volume is exclusive to itself. Note that the Bitmap Structure is identified in the container's Checkpoint Superblock Descriptor and tracks the allocation status of all blocks in the container. Therefore, volumes in a container will all report the same amount of free space, which is free space of the container as a whole. In this way APFS provides space sharing among volumes. A forensic issue here is that storage blocks don't belong to a specific volume until they're allocated to it. When a block is unallocated, it returns to the container pool. It may be extremely difficult or impossible to tell when data is recovered from unallocated blocks which volume the data belonged. This can be an issue for data carving, for example.

Checkpoints themselves are a file system operation and provide a comprehensive means to managing current and historical system states. The forensic significance here is that the common issue of limited-time file recovery (undeletion) in HFS+ can be all but disregarded. The issue related to balancing or "B-Trees" used in HFS+ is that when a file name is removed from the system, the balancing nature of a B-Tree quickly overwrites information as the tree is balanced. B-Trees are still used in APFS, but having checkpoints statically committed to disk means that the information may be available over a longer period of time since checkpoints other than the current one aren't in use.

APFS is engineered to support up to 9 quintillion files, whereas HFS+ can support 4 billion. Put another way, it would require over 2 billion HFS+ volumes to equal the potential file count capacity of a single APFS volume. Additionally, APFS uses 64-bit address spacing compared to 32-bit address space in HFS+. Further, APFS makes use of nanosecond time stamps over the less granular HFS+ to-the-second time stamps. APFS uses the UNIX epoch of 01 January 1970 as the basis for time calculation as do many other UNIX and UNIX-like file systems. Unlike almost all of those other file systems, however, APFS is immune from the 17 January 2038 problem. Other file systems that are not immune cannot store time values past the 2038 date. Other features will be detailed to follow.

APFS makes use of nanosecond time stamps over the less granular HFS+ to-the-second time stamps. APFS uses the UNIX epoch of 01 January 1970 as the basis for time calculation as do many other UNIX and UNIX-like file systems.

APFS isn't a journaling file system as it does not maintain a journal in the sense of HFS+. Rather it uses Apple's atomic safe-save and Copy-on-Write (CoW) to accomplish this. It appears that APFS was designed specifically for solid state media, hence CoW, but its intended use does include traditional platter disks where CoW can be disabled. Since safe-save operations are atomic, disk writes either happen in totality or not at all in either case. CoW fundamentally dictates that when a resource (file, folder, data, etc.) is copied, unless changes are made to the resource, then the resource data can be effectively shared between the copy and the original and data is not actually copied. Only when changes occur (on write) is the original copied and changes then made.

Closely related to CoW is the concept of APFS file cloning. In HFS+, when changes occur, the write is made in place. In other words, the file data to be changed is overwritten with the new data. APFS file cloning doesn't actually create a new copy. Rather, it records and stores the change data using delta encoding. That is, instead of writing out a new data file, the changes are stored as differences (deltas). For example, when a document file is opened, the original source document is read with the deltas abstracted over so the user sees only the newest document. This applies only to content data. Metadata is copied and unique to both the original and copy. As with the concept of hard links, content data remains on disk until all linked references are removed, so deleting a file doesn't absolutely imply that the storage blocks will be unallocated or that any storage space will be freed unless deletion occurred for the only remaining reference. Additionally, see the section on snapshots to follow. This saves storage space and contributes to failure recovery. Forensically it means that several historical versions of a file may be recoverable. Both atomic safe-saves and CoW can be more efficient than journaling, which enforces a write-twice mechanism. For the forensic examiner, it means new options exist for file recovery.

Encryption is part of APFS design, but its use isn't strictly enforced. If encryption is used, implementation can be either single-key or multi-key. Multi-key encryption provides the possibility to encrypt the whole disk with one key, then encrypt user files with a separate key. Forensically, this means that access to the encrypted APFS container doesn't guarantee access to user file data. Encryption modes are either AES-XTS or AES-CBC and chosen based on hardware.

Encryption considerations at this point should also include Apple's T2 Security Chip. This chip is an amalgamation of other controllers that were typically discrete chip packages on earlier Macintosh hardware, as well as new features specifically targeting security and authentication. Documentation of chip internals is scarce and much of what is available is based on theory and experimentation. Fundamentally, the T2 feature set includes a flash memory controller, a Secure Enclave (SE) coprocessor, and an encryption engine. The SE coprocessor stores a digest of installed system software, machine ID, and startup volume ID that's cryptographically signed by Apple when the operating system is installed. The SE coprocessor then uses this signed digest to validate the system at startup on Secure Boot-enabled systems. If it this process returns invalid, system software is downloaded and installed again, a new digest is created and then sent to Apple for signing. When the signed digest is returned, it replaces the previous signed digest and the process repeats at each boot time.

APFS disk encryption is handled entirely in hardware using 256-bit AES. For encrypted disks, APFS containers themselves do not require a password as the Media Encryption Keys (MEKs) never leave the T2 chip. Similar to disks encrypted with BitLocker and the Trusted Platform Module (TPM chip) on IBM-based personal computers, removing the APFS disk from the machine of origin renders the contents unreadable on a different machine. Moreover, if the machine of origin subsequently has a replacement disk installed and configured for booting, the previous disk will no longer be readable by that machine, either. Theoretically, the signed digest stored in the SE coprocessor will have changed and will no longer match the first disk. The takeaway for forensic examiners is that special consideration and handling must be applied in cases involving T2 chip/Secure Boot equipped hardware.

APFS Snapshots are a user-initiated capture of system state at a specific point in time and not totally unlike snapshots commonly used to save the state of virtual machines. Time Machine also creates and uses local snapshots, making a degree of restoration available even if the Time Machine backup disk isn't available. In either case, APFS Snapshots are different from the system's use of Checkpoints (Checkpoint Superblock and Checkpoint Superblock Descriptor). With APFS Snapshots, files are tracked by an Object ID and a Transaction ID where the Transaction ID is incremented each time the file is updated. APFS Snapshots record these two index numbers to track state at that point in time. Once a

snapshot is made, included files are protected from actual deletion or overwrite as long as the snapshot exists. This is true even if the user elects to delete a file in Finder, for example. This can be an excellent source of historical data, especially taken together with the temporal nature of the snapshot feature. Combined with the persistence of original associated data can provide a very clear picture of user activity over time.

To simply demonstrate this, below is a small 15 MB APFS volume:

```
Container spanning 14.96 MB (3830 blocks) with 1/1 volumes
nextCSBID: 0x1
Volume omap OID: 0x70
Volume 1: (Block 0x6e)  (XID: 0x5)
        Label: 'untitled'
case-insensitive          Contains 3 files, 1 directories, and 0 symlinks
      Size:    32.0 KB (8 blocks)
Volume 'untitles' autoselected
FD: 5
```

Figure 2 – APFS Container information

There are no user created directories or files on this volume to start with. An examination of the blocks displays both Object IDs (OID) and Transaction IDs (XID) by block:

```
FSleuth(untitles:/)> blockmap
NUM BLOCKS 3830
Block 0x0 (OID: 0x1, XID: 0x5) is container
Block 0x1 (OID: 0x1, XID: 0x5) is checkpoint
Block 0x2 (OID: 0x1, XID: 0x5) is container
Block 0x3 (OID: 0x3, XID: 0x2) is checkpoint
Block 0x4 (OID: 0x1, XID: 0x2) is container
Block 0x5 (OID: 0x5, XID: 0x3) is checkpoint
Block 0x6 (OID: 0x1, XID: 0x3) is container
Block 0x7 (OID: 0x7, XID: 0x4) is checkpoint
Block 0x8 (OID: 0x1, XID: 0x4) is checkpoint
Block 0x9 (OID: 0x400, XID: 0x1) is space manager
Block 0xa (OID: 0x401, XID: 0x1) is reaper
Block 0xb (OID: 0x400, XID: 0x2) is space manager
Block 0xc (OID: 0x403, XID: 0x2) is B-tree(subtype space manager free queue)
Block 0xd (OID: 0x403, XID: 0x2) is B-tree(subtype space manager free queue)
Block 0xe (OID: 0x401, XID: 0x2) is reaper
Block 0xf (OID: 0x400, XID: 0x3) is space manager
Block 0x10 (OID: 0x403, XID: 0x3) is B-tree(subtype space manager free queue)
Block 0x11 (OID: 0x405, XID: 0x3) is B-tree(subtype space manager free queue)
```

Figure 3 – Object IDs and Transaction IDs

Further down the list displays the OID and XID of the volume itself (Block 0x59, OID 0x402, XID 0x2):

```
Block 0x59 (OID: 0x402, XID: 0x2) is volume
Block 0x5a (OID: 0x5a, XID: 0x2) is object map
Block 0x5b (OID: 0x5b, XID: 0x2) is B-tree(subtype object map)
Block 0x5c (OID: 0x404, XID: 0x2) is B-tree(subtype fstree)
Block 0x5e (OID: 0x5e, XID: 0x3) is B-tree(subtype block ref tree)
```

Figure 4 – Volume Object ID (0x402) and starting Transaction ID (0x2)

A simple text file was created for this example titled “TEST1.txt”:

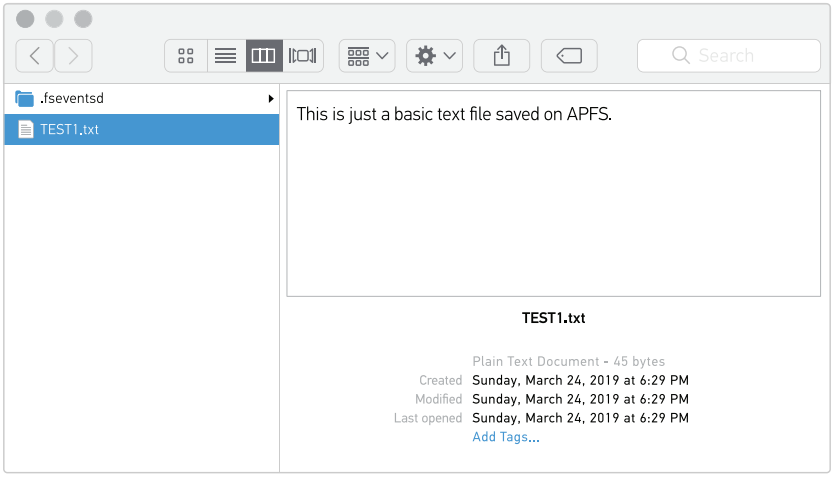

Figure 5 – Creating a sample file for testing

Saving it to the volume, together with various other system actions, causes transactional incrementation. Note that the Object ID is the same (0x402), but the Transaction ID has incremented (0x7):

```
Block 0x7e (OID: 0x7e, XID: 0x7) is B-tree(subtype object map)
Block 0x7f (OID: 0x402, XID: 0x7) is volume
Block 0x80 (OID: 0x80, XID: 0x7) is object map
Block 0x81 (OID: 0x81, XID: 0x7) is B-tree(subtype object map)
```

Figure 6 – Volume Transaction ID has incremented

Running a file copy command to overwrite the file's contents with new text, i.e. "cp TEST2.txt TEST1.txt", results in file content change:

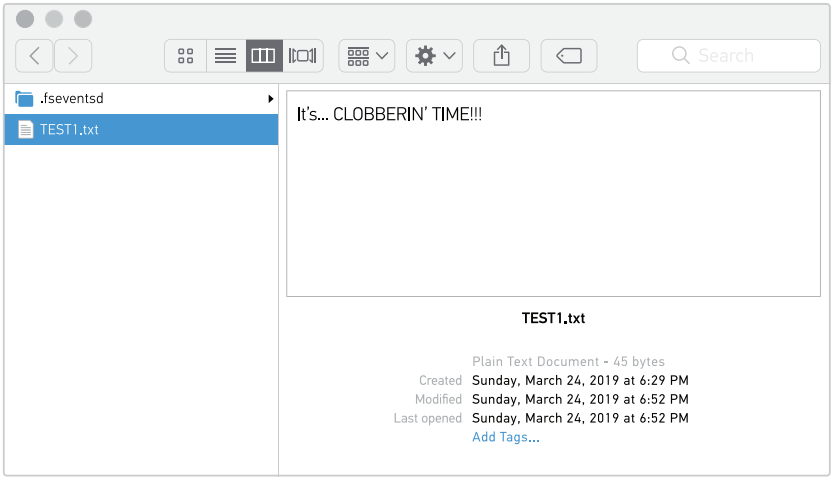

Figure 7 – Changing the test file's content

It also results in XID incrementation for OID 0x402:

```
Block 0x86 (OID: 0x86, XID: 0x8) is B-tree(subtype object map)
Block 0x87 (OID: 0x402, XID: 0x8) is volume
Block 0x88 (OID: 0x88, XID: 0x8) is object map
```

Figure 8 – Content change increments Transaction ID

The new data content displayed in Figure 8 can also be seen at the byte level beginning at offset 0x88000:

```
00087FF0  06 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................
00088000  49 74 27 73 2e 2e 2e 20  43 4c 4f 42 42 45 52 49  |It's... CLOBBERI
00088010  4e 27 20 54 49 4d 45 21  21 21 00 00 00 00 00 00  |N' TIME!!!......
```

Figure 9 – Changed content visible at byte level

This operation would normally be expected to overwrite the previous data content. However, it still exists on disk as seen at offset 0x7f000:

```
0007f000  54 68 69 73 20 69 73 20  6a 75 73 74 20 61 20 62  |This is just a b
0007f010  61 73 69 63 20 74 65 78  74 20 66 69 6c 65 20 73  |asic text file s
0007f020  61 76 65 64 20 6f 6e 20  41 50 46 53 2e 00 00 00  |aved on APFS....
0007f030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................
```

Figure 10 – Original content still exists on disk

This paper is a very high-level overview of APFS with only a brief excursion into just one of the internal mechanisms that set it apart from the Apple file systems that came before. Many hours to date have been devoted to the study of the structural intricacies and internal nuances that make this file system unique.

APPLE FILE SYSTEM (APFS) OVERVIEW

Magnet AXIOM version 3.0.0 continues the legacy to prove itself to be a dependable solution in the field of Apple device forensics with its unparalleled forensic support for MacOS and iOS devices running APFS. This support includes decryption of FileVault2 containers and volumes, together with dozens of new user and operating system artifacts for macOS and iOS, as well as browsing support for APFS in AXIOM Examine's File System Explorer. New macOS artifacts include Apple Accounts, Bash Sessions, Bluetooth Devices, Calendar (ICS), CoreAnalytics, Daily Logs, Deleted Accounts, Dock Items, File System Events, Finder MRU, Finder Sidebar Items, and many more.

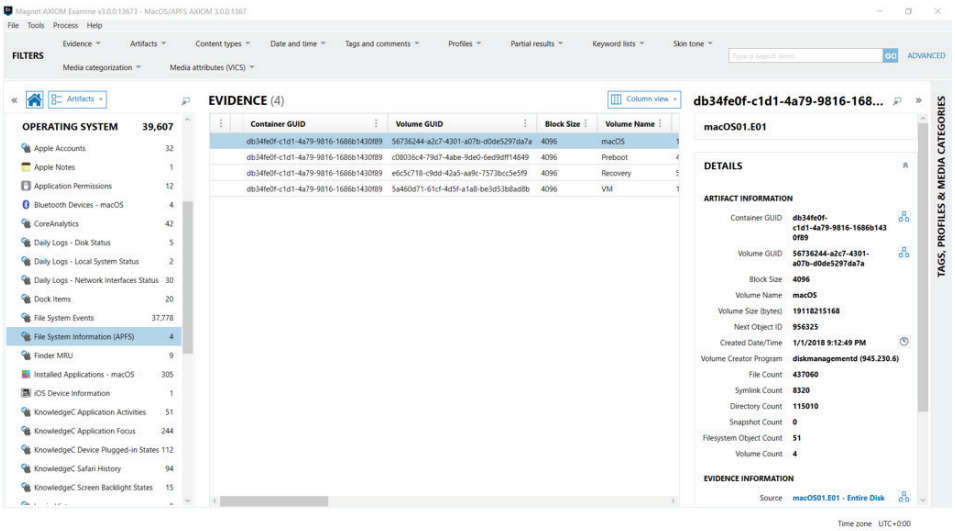

Figure 11 – MacOS and APFS artifacts in AXIOM

Artifacts Explorer in AXIOM Examine provides detailed filtering, tagging, and reporting on APFS and MacOS artifacts through the use of the Filters Bar, together with the Navigation, Evidence, and Details panes without surprises. Leveraging the "artifact first" approach for MacOS/APFS evidence sources does not require additional learning to master AXIOM's interface. If an examiner knows how to capitalize on those features to examine any evidence source, they already know how to do the same for macOS/APFS evidence sources.

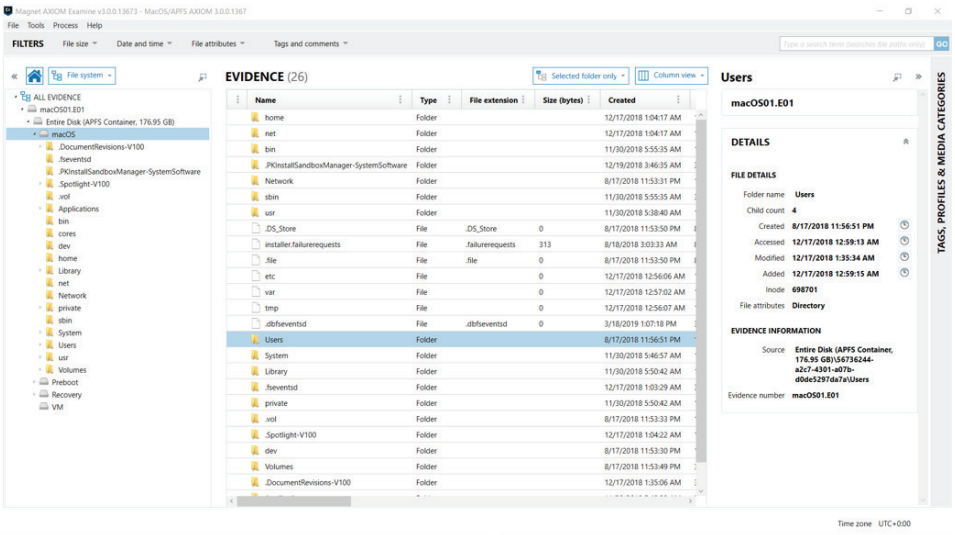

Figure 12 – Browsing APFS in AXIOM

File System Explorer provides a fast, intuitive way to quickly navigate the directory structure of an APFS volume while providing a complete tree so the examiner can easily see to which container the volume belongs. Taking advantage of source linking in Artifacts Explorer is the fastest way to locate and browse locations in File System Explorer specific to artifacts and related items of interest.

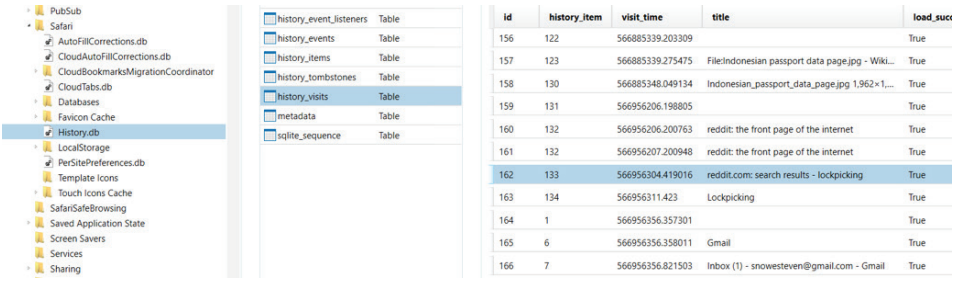

Figure 13 – Browsing an SQLite database in AXIOM

File System Explorer continues to provide a stable method for viewing the contents of SQLite databases. This is especially helpful when manually verifying the information displayed in Artifacts Explorer without having to save the database to an external location or open it using third-party tools unless that option is specifically desired.

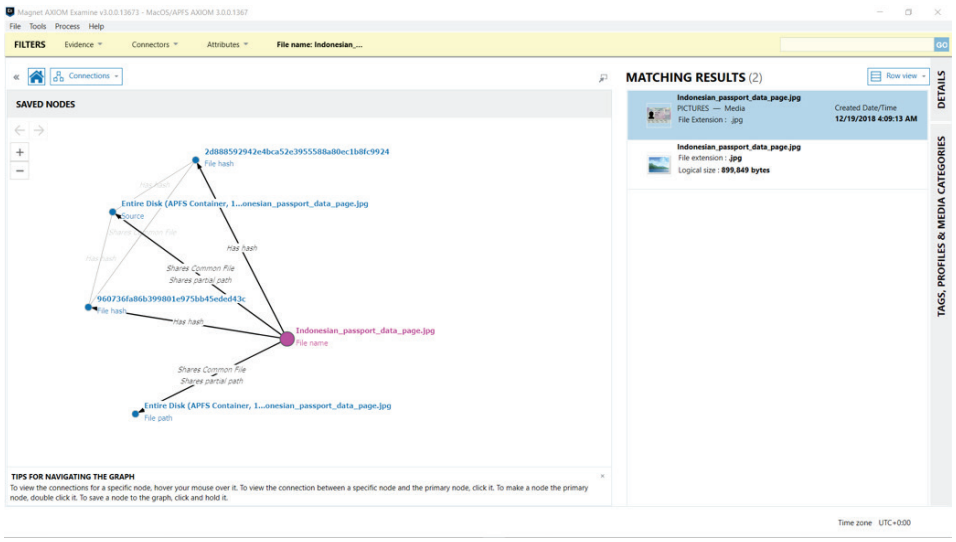

Figure 14 – APFS Connections in AXIOM

AXIOM Connections fully supports macOS/APFS artifacts with all the functionality examiners have come to expect. Again, following the rule of least surprise, if a practitioner knows how to leverage the power of AXIOM Connections in any case, they already know how to do it with APFS evidence sources.

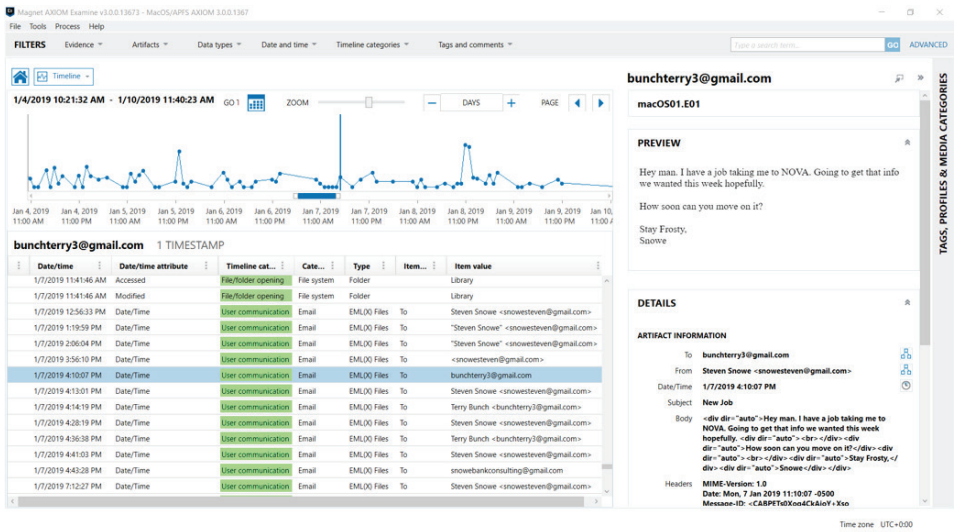

Figure 15 – Timeline Explorer for APFS artifacts in AXIOM

Timeline Explorer is a newly introduced feature in AXIOM 3.0.0. It replaces and greatly improves upon the Artifact Explorer's Timeline View in previous versions of AXIOM. By changing the functionality from a simple view option for the Evidence Pane to a full featured explorer in its own right, Timeline Explorer now provides examiners with both the power and flexibility to navigate temporally significant MacOS/APFS artifacts and tell the story the evidence wants to tell. Examiners who understand how to exploit Timeline View in previous versions of AXIOM will find the new Timeline Explorer very intuitive with a shallow learning curve.

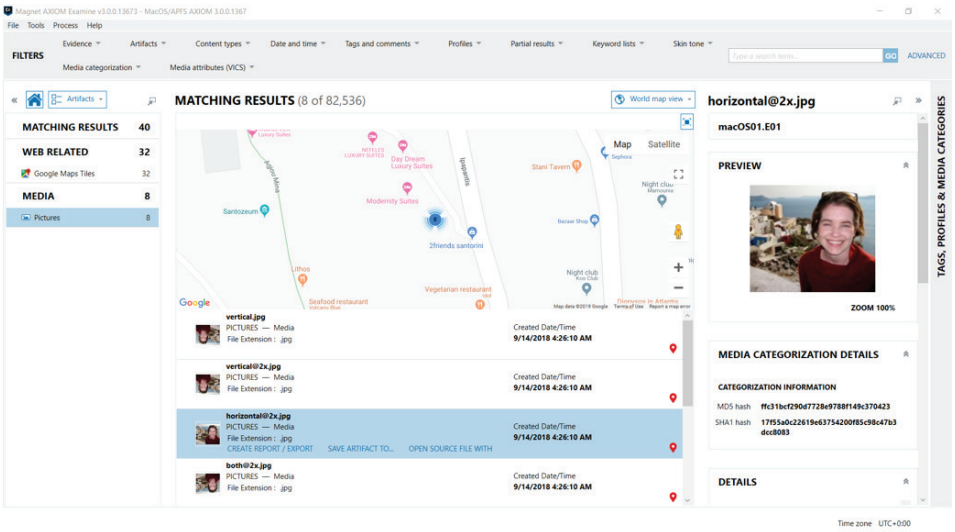

Figure 16 – World Map View for MacOS/APFS artifacts in AXIOM

macOS and APFS artifacts with geolocation data are easily mapped and located using World Map View for the Evidence Pane in Artifacts Explorer. The functionality found in this part of AXIOM can also prove indispensable for iOS and Apple iCloud artifacts as well. By combining related evidence sources within the same case, examiners can have a much more complete understanding of events and locations, and how those relate to each other.

CONCLUSION

APFS, and to a lesser extent macOS, will prove to be a challenge for developers of digital forensic software for some time to come. New aspects and features yet to be implemented by Apple will be discovered as proliferation and use of these technologies increase. Apple has made clear that APFS is their file system of both today and tomorrow, so we should expect to encounter it in our work for many years to come. Magnet Forensics is dedicated to the continued pursuit of knowledge and understanding of APFS and can be counted upon to deliver the benefits of our investment to digital forensic practitioners just as timely and thoroughly as our global customers and partners have come to expect. For in-depth training on AXIOM and digital forensics, including macOS/APFS, please visit magnetforensics.com/training-overview/.

SEE AXIOM IN ACTION FOR YOURSELF

If you'd like to learn more about Magnet AXIOM and how it can help you run smoother investigations, visit magnetaxiom.com. While you're there, you can learn more about the product, request an in-depth personal demo from an AXIOM expert, and request a free 30-day trial version.

Learn more at magnetforensics.com

For more information call us at 1-844-638-7884
or email sales@magnetforensics.com