Annie Schweikert
CINE-GT 1807 Digital Preservation
Nicole Martin
14 December 2018

A Guide to the QuickTime File Format

"The QTFF is an ideal format for the exchange of digital media between devices, applications, and operating systems, because it can be used to describe almost any media structure." —Apple, "Introduction to QuickTime File Format Specification"[1]

Apple's QuickTime architecture has produced a series of media players, a codec, two wrappers, and the inspiration for several more file formats. Its popularity and flexibility as a file format has led archives to adopt it at each level of storage, as preservation, mezzanine, and access files. Given its popularity, knowledge of the format's structure and history, as well as ways to edit and investigate its content, are essential skills for managing digital audiovisual files.

What's in a QuickTime file?

A brief history of the QuickTime file format[2]

With the release of QuickTime 1.0 in 1991, Apple packaged a self-contained system of video creation. Users could encode a video file in the "QuickTime" codec,[3] wrap the encoded video in the QuickTime file format (or wrapper), and play the resulting file in the QuickTime media player. At the time, digital video creation typically required complex video hardware, and QuickTime's release—allowing the synchronization of audio and video—represented a large step forward for both Apple and desktop digital video.[4]

While Apple's codec was soon superseded by higher-resolution encodings, the QuickTime file format grew in popularity over the next decade due to its flexibility and interoperability. In 1998, when the International Organization for Standardization's Moving Picture Experts Group issued a call for proposals to help define a developing standard for digital media, it was in such broad use that a coalition of technology companies (not just Apple, but IBM, Netscape, Sun Microsystems, and others) submitted a joint proposal advocating for the QuickTime file format to serve as a "unified digital media storage format" within the specification.[5]

This "unified file" format became the first version of the MPEG-4 file format (more commonly referred to as "MP4 version 1"), published as ISO/IEC 14496-1:2001 in 2001. In 2003, MP4 was revised into "version 2" (ISO/IEC 1446-14:2003). MPEG and ISO then used this version of MP4 to derive a generalized structure for a wide-ranging family of file formats.[6] What had finally emerged from this collaboration was not technically a standalone file format, but a standard

[1] "Introduction to QuickTime File Format Specification," *Documentation Archive,* Apple, Apr. 2006, https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFPreface/qtffPreface.html
[2] With thanks to fellow MIAP student Becca Bender, whose paper "Too Many Quicktimes! (and other related confusions): Thoughts from an Emerging Audiovisual Archivist" (Dec. 2017) informed much of this section's history.
[3] Technically, the name of the codec packaged in QuickTime 1.0 was "Apple Video" (also known as Road Pizza) but it is colloquially referred to as "QuickTime." See Bender, pg. 2-3.
[4] Jeanette Borzo, "Apple's QuickTime slated for release this month," *InfoWorld*, 4 Nov. 1991, pg. 8, https://books.google.com/books?id=Xz0EAAAAMBAJ&pg=PA8#v=onepage&q&f=false
[5] "ISO Adopts QuickTime File Format as Starting Point for Developing Key Component of MPEG-4 Specification," *PR Newswire,* February 11, 1998, https://tech-insider.org/digital-video/research/1998/0211.html
[6] David Singer and Thomas Stockhammer, "White paper on an Overview of the ISO Base Media File Format" (presentation), ISO/IEC JTC1/SC29/WG11 MPEG2018/N18093, Oct. 2018, https://mpeg.chiariglione.org/sites/default/files/files/standards/docs/w18093_0.zip

intended to be extended into a range of similar, interoperable containers. This structure, which was published as the ISO Base Media File Format (ISOBMFF), or MPEG-4 Part 12 (ISO/IEC 14496-12:2004), is the direct descendent of the original QuickTime specifications.[7]

Since its codification in 2004, ISOBMFF has provided the starting point for many familiar file formats. ISOBMFF profiles, or fully functional file formats extended from the basic ISOBMFF specification, include MP4, Motion JPEG 2000, and 3GP, among others.[8] One of those file format extensions is the modern specification for QuickTime—now technically the QuickTime profile of MPEG-4 Part 12. The modern QuickTime file format is very similar to the original QuickTime file format, despite having gone through the intermediary processes of standardization into the MP4, the ISOBMFF, and then an extension back into the "QuickTime file format."

The structure of the QuickTime file format

Atoms are the building blocks of matter; they are also the building blocks of QuickTime files. All the data and metadata in QuickTime files are contained in subdivisions called "atoms." This structure affords QuickTime a high degree of interoperability and backwards compatibility. If a media player or architecture does not recognize a certain atom, it can simply ignore it and parse the rest of the file anyway.[9] QuickTime files are also flexible; as few atoms are required for a compliant file, QuickTime files can range from simple to complex.

Every MPEG-4-compliant file must contain two atoms: the "mdat" (movie sample data) atom and the "moov" (movie resource metadata) atom.[10] [11] The mdat atom is a container for the data, stored in samples, while the moov atom is a container for all the metadata that describes this bitstream, its locations, and how it should be presented. Together, these atoms contain all the information in the file, and are essential to its interpretation. Without data, there is no file to play, but without metadata the data is uninterpretable.

Nested beneath the moov atom are a series of other atoms arranged in a hierarchical structure. These atoms may be "container" atoms that contain other atoms, such as the moov atom, or "leaf" atoms that contain actual data or metadata, such as the mdat atom. While atoms (with some exceptions) do not have to be stored in a specific order within their containers, the hierarchy of these containers is important to interpretation. In some cases, the parent atom of an atom defines the scope of the atom. For example, a "user data" atom stores descriptive metadata, and may be repeated several times within a single QuickTime file; if the user data atom's parent atom is a moov atom, the metadata it stores is relevant to the entire file, but if the parent atom is a "track" atom (describing the audio, video, or data tracks), the metadata is only relevant to that track.[12]

---

[7] Ibid.

[8] A full list of registered extensions to MPEG-4 Part 12 may be viewed at the MPEG-4 Registration Authority website: http://mp4ra.org

[9] "Introduction to QuickTime File Format Specification."

[10] Maxim Levkov, "Understanding the MPEG-4 Movie Atom," *Adobe,* 18 Oct. 2010, https://www.adobe.com/devnet/video/articles/mp4_movie_atom.html

[11] "Overview of QTFF," *Documentation Archive,* Apple, Apr. 2006, https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap1/qtff1.html

[12] Ibid.

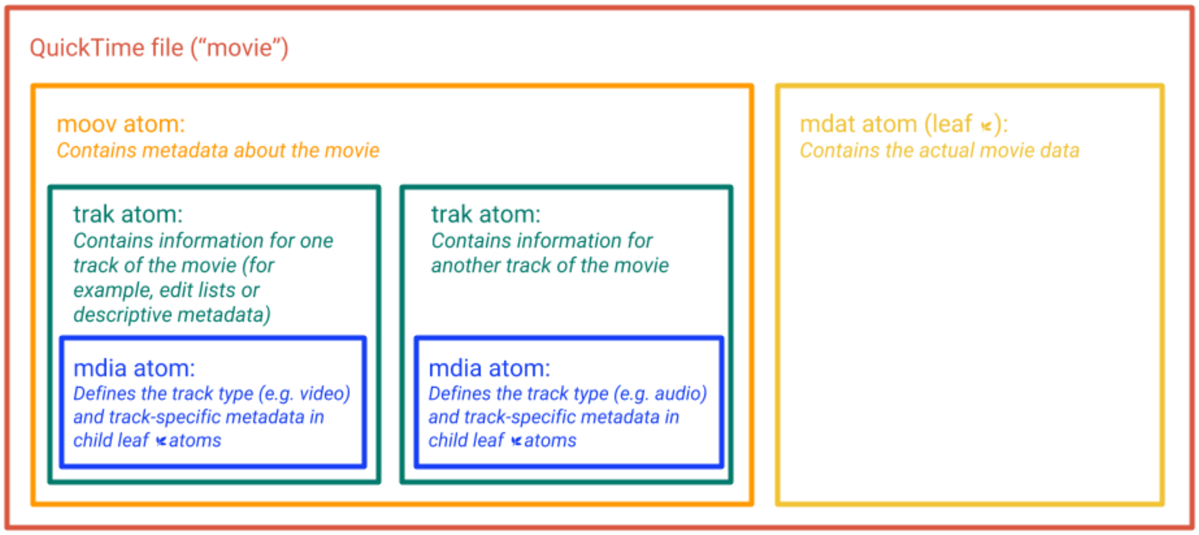

QuickTime file ("movie")
moov atom:
Contains metadata about the movie
trak atom:
Contains information for one track of the movie (for example, edit lists or descriptive metadata)
mdia atom:
Defines the track type (e.g. video) and track-specific metadata in child leaf atoms
trak atom:
Contains information for another track of the movie
mdia atom:
Defines the track type (e.g. audio) and track-specific metadata in child leaf atoms
mdat atom (leaf):
Contains the actual movie data

Figure 1. A very basic QuickTime file structure illustrating the hierarchical container nature of atoms.[13]

The atom structure of the QuickTime file format allows for playback and analysis by many different players and tools, but it also presents the potential for the files to be interpreted and presented very differently depending on those tools. To take one example, the information in the moov atom may be compressed into a single child "cmov" (compressed movie) atom. Some players recognize the cmov atom and can unpack the information it contains; other players will skip the cmov atom and report a failure in playback. Even playing these files in a QuickTime player does not guarantee identical presentation. In order to decode different video formats, QuickTime players depend on "component libraries" stored separately from the application itself. These component libraries may be changed by other applications, meaning that two machines with otherwise identical versions of QuickTime may still present the same file differently.[14]

Atoms and their content

The content of an atom is structured into three or more fields. In most atoms, the first field of four bytes is reserved for the size and the next four-byte field holds an atom type, expressed as a four-character code. The rest of the atom contains one or more fields of data (if a leaf atom) or other atoms (if a container).

[13] Images and screenshots that are otherwise unattributed were created by Annie Schweikert, Dec. 2018.
[14] Dave Rice, "Sustaining Consistent Video Presentation," *Tate Research,* 2015, https://www.tate.org.uk/about-us/projects/pericles/sustaining-consistent-video-presentation

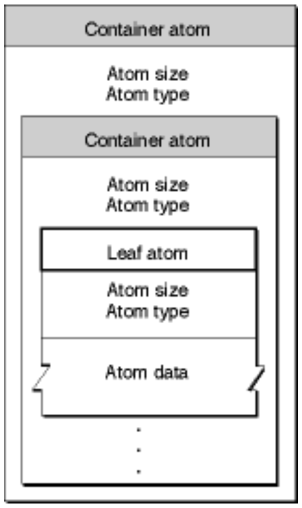


Figure 2. "A sample atom," from the QuickTime documentation at https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap1/qtff1.html

Identifying the atom size first allows a media player to skip to the end of the atom if it does not recognize the type. The eight bytes allotted to store the atom's size enforces a size limit of 4,294,967,296 bytes (around four gigabytes), but it is possible to insert a 64-bit "extended size" field into the atom, increasing the size limit to 18 exabytes.[15]

The actual information contained within the atoms can describe a wide range of timing, display, and navigation information. While a full description is outside of the scope of this paper,[16] one example hierarchy beginning with the moov atom might include:

- moov (movie) atom: Container atom for information describing the entire file, or "movie."
    - trak (track) atom: Container atom for information describing one track (audio, video, or data) of the movie, such as different edits and relationships between different tracks (e.g., synchronization of audio and video tracks).
        - mdia (media) atom: Container atom for information describing the media within a track, such as its duration, type, and volume.
            - minf (media information) atom: Container atom for information about how the type-specific handler (audio, video, etc.) should process the media data handler.[17]
                - stbl (sample table) atom: Container atom describing the time, location, and interpretation of the file's data samples.
                    - (Only leaf atoms past this point.)

Note that at each level of this hierarchy, atoms provide detail about the interpretation of smaller and smaller units of data, but that none of these atoms actually stores data—they just point at the location of the data, stored separately in the mdat atom.[18]

[15] "Overview of QTFF."

[16] For more detail on QuickTime atoms, see Apple's documentation, specifically "Movie Atoms": https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html and "Media Data Atom Types": https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap3/qtff3.html

[17] Apple supplies different "handlers" to interpret different types of media, including video, audio, text, and timecode, among others. See "About Media Handlers" in Apple's documentation for more, at https://developer.apple.com/library/archive/documentation/QuickTime/RM/MediaTypesAndHandlers/MHFundamentals/B-Chapter/2AboutMediaHandlers.html

[18] Atom hierarchy and definitions informed by "Movie Atoms," *Documentation Archive,* Apple, Apr. 2006, https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html

QuickTime in the archive

The QuickTime file format is ubiquitous in archives and is widely adopted at every level of storage and delivery. It is adaptable to many different types of codecs, from highly detailed, uncompressed video for preservation to codecs compressed for web and phone streaming. Apple has marketed QuickTime as a natural pair for popular codecs such as uncompressed video as well as Apple's own ProRes, staking associations with video in archival and production contexts, and QuickTime's association with the ISOBMFF family of formats opens up another vector of markets.

QuickTime for preservation

QuickTime is often used as a default wrapper for a slate of uncompressed video codecs. "Uncompressed video" is a blanket term for a group of codecs differentiated by sampling depth and chroma subsampling; two of the most common types are "v210" (10-bit 4:2:2 video) and "2vuy" (8-bit 4:2:2 video).[19] Uncompressed video is popular for its simplicity; the bitstream information is contained without compression and is therefore easy to reverse engineer (though not all uncompressed video is stored in the same bitstream layout, meaning uncompressed codecs are not inherently compatible).[20]

However, uncompressed video's simplicity is also its downfall. Uncompressed video is not self-descriptive; many important display characteristics are simply not contained within the encoded data, meaning that the wrapper must be responsible for documenting such basic presentation as color space and aspect ratio. Without this presentation metadata, uncompressed video can look significantly different between players.

Apple endorsed uncompressed video as a codec compatible with QuickTime as early as 1999, when it published specifications for uncompressed video codecs and their interaction with the QuickTime wrapper. The specifications attempt to ensure consistent playback of uncompressed files wrapped in QuickTime, listing a series of atoms required for compliance with consistent playback and presentation, including "colr" (color parameters), "fiel" (field and frame ordering in storage and playback), "pasp" (pixel aspect ratio), and "clap" (clean aperture, or the cropping of the frame in playback).[21] These values represent a minimum set of conditions for accurate storage as well as an imperative—with minimal display information in the file to fall back on, presentation depends singularly on the strength of the wrapper and its interpretation. Any application that cannot fully parse QuickTime will fail to correctly present the file itself.

Unfortunately, compliance with TN2162 is sometimes ambiguous, as players and reporting tools may try to infer or assume values for these atoms even if they are not present. For example, it is impossible to present a video without assigning some value for the aspect ratio of each pixel; if there is no value stored in the pasp atom (or no pasp atom at all), a player must arbitrarily assign this value before it can display the image. Reporting tools that make the same assumption may report a value when it is not actually present in the file itself. [22]

QuickTime for mezzanine delivery and production

---

[19] "Technical Note TN2162: Uncompressed Y´CbCr Video in QuickTime Files." *Apple Documentation Archive,* 14 Dec. 1999, https://developer.apple.com/library/archive/technotes/tn2162/_index.html

[20] Peter B., Hermann Lewetz and Marion Jaks, "Comparing video codecs and containers for archives: Codec: Uncompressed," *Oesterreichische Mediathek*, 13 Aug. 2015, http://download.das-werkstatt.com/pb/mthk/info/video/comparison_video_codecs_containers.html#codec_uncompressed

[21] Dave Rice, "Audiovisual Adherence," *PERICLES,* 2017, https://www.tate.org.uk/about-us/projects/pericles/audiovisual-adherence

[22] "Technical Note TN2162."

Another family of codecs often paired with QuickTime is Apple's proprietary ProRes. An intermediate, lossy series of codecs, ProRes encodings range from efficient compressed video at a low bit rate (ProRes 422 Proxy) to near-lossless 12-bit video (ProRes 4444 XQ) and raw video (ProRes RAW).[23] ProRes is widely used in video post-production—even some cameras support ProRes—and is especially common in broadcast contexts. Despite its proprietary nature, ProRes has been reverse-engineered and can be encoded and decoded with open source tools such as FFmpeg, ensuring a measure of sustainability for the future.[24]

ProRes has been associated with QuickTime since its release in 2001, when the two were bundled together in Apple's Final Cut software.[25] Though certainly a product of Apple's own marketing, the pairing makes sense from a structural standpoint as well, as Apple designed ProRes to mimic QuickTime's atomic structure. In a ProRes video, the data is divided into frames holding one or two fields (in progressive and interlaced video, respectively). Each frame begins with a frame container atom, which stores four bytes of size information and four bytes describing the type—always "icpf," the four-character code reserved for a ProRes frame. After the frame container comes the frame header, which stores metadata describing the frame's display. Following the frame header is the picture, which itself stores a header, a table describing the subdivision of the picture data, and the picture data itself (subdivided into "slices").[26]

In notable contrast to uncompressed video, ProRes is self-descriptive and does not rely on the container for interpretation. Some information required in the wrapper by TN2162 is stored in the frame header, such as the color matrix, color transfer function, and field ordering.[27] In other cases, interpretation information is integrated into a set of stricter rules for ProRes video. For example, square pixels are required, removing the need to store the pixel aspect ratio anywhere.[28] A potential problem that arises from using QuickTime as a ProRes wrapper, therefore, is the possibility of metadata collisions—conflicting values stored in the video essence and in the container that attempt to describe the same presentation aspects. The interpretation of these conflicts will vary from player to player.

Adapted QuickTime for access and delivery

Finally, though QuickTime is not typically used in the creation of access files, it is the direct predecessor to MP4, an international standard for web and mobile streaming.[29] The first MP4 file format was developed by the Moving Pictures Expert Group based directly on Apple's original QuickTime specifications. Later, when MP4 was revised as MPEG-4 Part 14, it retained several key structural similarities that were then generalized into MPEG-4 Part 12 (the ISOBMFF)—from which the modern QuickTime file format is extended.[30]

---

[23] "Apple ProRes White Paper," *Apple*, Apr. 2018, https://www.apple.com/final-cut-pro/docs/Apple_ProRes_White_Paper.pdf
[24] "Apple ProRes 422 Codec Family," *Sustainability of Digital Formats,* Library of Congress, 27 July 2017, https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml
[25] Ibid.
[26] "Apple ProRes," *Multimedia Wiki,* 23 May 2013, https://wiki.multimedia.cx/index.php?title=Apple_ProRes
[27] Ibid.
[28] Larry Jordan, "Understanding ProRes 422," *LarryJordan.com,* 15 May 2011, https://larryjordan.com/articles/understanding-prores-422/
[29] ISO/IEC 14496-14:2003, published 15 Nov. 2003, https://github.com/OpenAnsible/rust-mp4/blob/master/docs/ISO_IEC_14496-14_2003-11-15.pdf
[30] "MPEG-4 File Format, Version 1," *Sustainability of Digital Formats,* Library of Congress, 27 July 2017, https://www.loc.gov/preservation/digital/formats/fdd/fdd000037.shtml

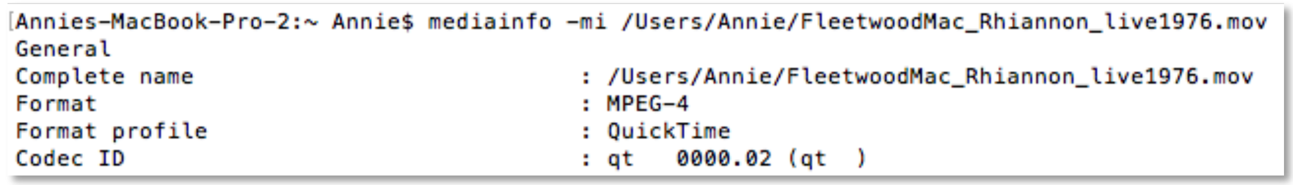
An MP4 file is made up of "boxes," building blocks that are "functionally identical" to atoms.[31] The MP4 standard differs from QuickTime most notably in its stricter requirements for metadata, and specifically in its requirements for "object descriptors."[32] The MP4 specification defines the "Object Descriptor Framework" as an infrastructure that governs a presentation's (video) objects and streams;[33] an object descriptor box groups and defines the relationships, synchronization, and description of the video data stream.[34]

MP4 files are often paired with the lossy H.264 codec, also known as AVC (Advanced Video Coding) and standardized as MPEG-4 Part 10 (ISO/IEC 14496-10). As another MPEG-4 standard, it is optimized to comply with the MP4 file format, and yet another MPEG-4 standard (MPEG-4 Part 15, ISO/IEC 14496-15) governs the storage of H.264 in ISOBMFF file formats.[35] The use of MP4 over MOV is partly spurred by the stricter guidelines for MP4 description, which lends itself to streaming presentations.[36]

Identifying and evaluating QuickTime files

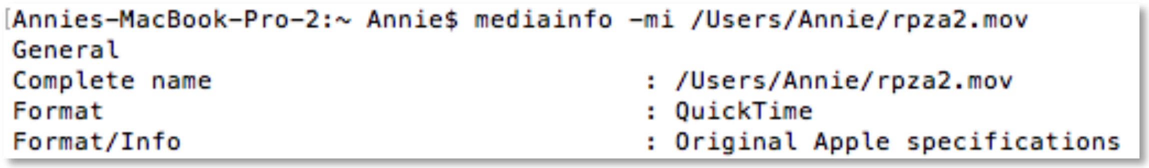
Identifying QuickTime files

QuickTime files are highly structured but can be confusing to identify on first glance. While most QuickTime files have consistent extensions (.mov) and MIME types (video/quicktime), these characteristics are not conclusive. Generic reporting tools, such Mac's Finder and the QuickTime inspector tool, will report information without enough granularity to allow sure identification. The quickest reliable way to identify QuickTime files is by using a reporting tool optimized for audiovisual media, such as MediaInfo, which will distinguish between early and modern QuickTime formats.



```
Annies-MacBook-Pro-2:~ Annie$ mediainfo -mi /Users/Annie/FleetwoodMac_Rhiannon_live1976.mov
General
Complete name                            : /Users/Annie/FleetwoodMac_Rhiannon_live1976.mov
Format                                   : MPEG-4
Format profile                           : QuickTime
Codec ID                                 : qt   0000.02 (qt  )
```

Figure 3. The first part of the MediaInfo report for a modern QuickTime file (a QuickTime profile of MPEG-4 Part 12).

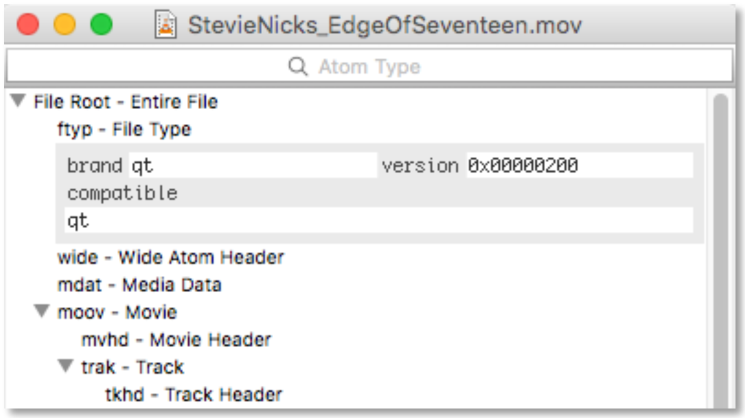


```
Annies-MacBook-Pro-2:~ Annie$ mediainfo -mi /Users/Annie/rpza2.mov
General
Complete name                            : /Users/Annie/rpza2.mov
Format                                   : QuickTime
Format/Info                              : Original Apple specifications
```

Figure 4. The first part of the MediaInfo report for a "classic" QuickTime file.

Though MediaInfo is a specialized audiovisual tool, tools also exist for QuickTime specifically. Atom Inspector, a legacy Mac application, reads the atom structure of a QuickTime file and displays both the hierarchy and the contents of the atoms.

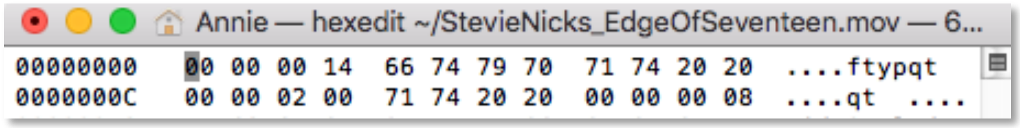
[31] "Overview of QTFF."
[32] "MPEG-4 File Format, Version 2," *Sustainability of Digital Formats,* Library of Congress, 21 Feb. 2017, https://www.loc.gov/preservation/digital/formats/fdd/fdd000155.shtml
[33] ISO/IEC 14496-14:2003.
[34] Glenn Pearson, "Methods to Store Metadata within Motion JPEG 2000 Files" (preprint), U.S. National Library of Medicine (NIH/HHS), 6 June 2005, Appendix 2, https://lhncbc.nlm.nih.gov/system/files/pub-2005-104.pdf
[35] David Singer and Thomas Stockhammer.
[36] Dave Rice, personal communication, 14 Dec. 2018.



```
▼ File Root - Entire File
    ftyp - File Type
      brand qt                    version 0x00000200
      compatible
      qt
    wide - Wide Atom Header
    mdat - Media Data
  ▼ moov - Movie
      mvhd - Movie Header
    ▼ trak - Track
        tkhd - Track Header
```

Figure 6. A QuickTime file viewed in Atom Inspector. Note the ftyp atom and its value ("qt "), as well as the mdat atom and the hierarchy that begins under the moov atom.

In some cases, an archivist may want to bypass a reporting tool and investigate the file itself, which is possible with a hex editor. When opening a QuickTime file in a hex editor (for example, the hexedit command-line utility or the Hex Fiend application for Mac), the ASCII (plain text) side of the editor will display the atoms by four-character code. In the below screenshot, the ftyp (filetype) atom is visible at the very beginning of the file. The ftyp atom was introduced as part of the ISOBMFF specification in 2004 and so only appears in QuickTime files created since then. It is mandated to appear first in the file—one of very few atoms whose position is defined—and is easy to use as positive identification. If the file conforms to the QuickTime extension of ISOBMFF, its value will be "qt " (including the two spaces, which makes up a four-character code and which is the same value reported by MediaInfo).[37]



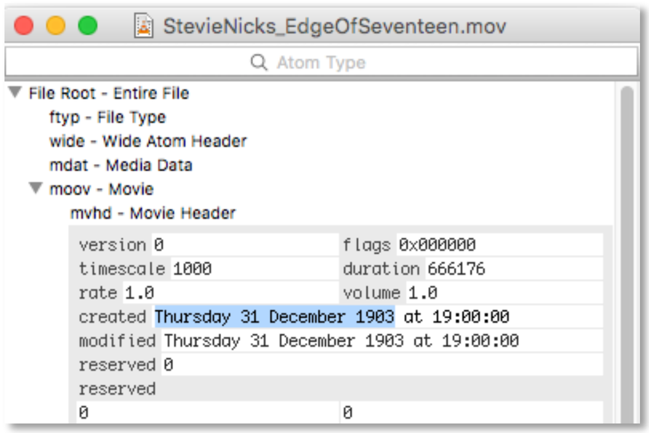
```
00000000   00 00 00 14  66 74 79 70  71 74 20 20  ....ftypqt
0000000C   00 00 02 00  71 74 20 20  00 00 00 08  ....qt  ....
```

Figure 6. The beginning of a QuickTime file as viewed in a hex editor. Note the ftyp atom and its value ("qt ") on the ASCII side of the viewer.

Editing QuickTime files

The ability to edit QuickTime files allows the archivist to fix incorrect information and add descriptive information. QuickTime files can be edited in a number of applications, including Atom Inspector (shown above) as well as QuickTime 7 Pro, Atom Inspector, Dumpster, and Metadata Hootenanny. It is often as simple as clicking within the atom's text field and writing over the existing data.[38]

[37] "Overview of QTFF."
[38] Dave Rice, personal communication, 26 Nov. 2018.



```
▼ File Root - Entire File
    ftyp - File Type
    wide - Wide Atom Header
    mdat - Media Data
  ▼ moov - Movie
      mvhd - Movie Header
        version 0                          flags 0x000000
        timescale 1000                     duration 666176
        rate 1.0                           volume 1.0
        created Thursday 31 December 1903 at 19:00:00
        modified Thursday 31 December 1903 at 19:00:00
        reserved 0
        reserved
        0                                  0
```

Figure 7. We didn't have digital video in 1903. Luckily I can change this.

It is much harder to add new atoms than it is to edit the existing atom data. Size information governs the management of a QuickTime file's atoms; any new atom would change the size information in the parent container atom, in the parent atom's parent atom, and so forth up the chain. Adding atoms without changing this information could break the file. Editing atoms directly in a hex editor is also fraught, as a single bit change in size may have the same ripple effects.[39]

Evaluating QuickTime files

QuickTime files are most important to evaluate when they represent an institution's preservation-level files. This note, Technical Note (TN) 2162, has been put into practice as a baseline test at cultural heritage institutions. Such a test can be performed with software that probes the bit-level structure of the file and checks the values it finds in the moov atom with a set of rules written to comply with TN2162. The software, MediaConch, is open source, downloadable for free, and comes with a built-in profile for TN2162.[40] [41]

In some other cases, especially at production archives, QuickTime/ProRes files represent a large proportion of digital accessions. Since ProRes is a far more self-descriptive codec,[42] presentation metadata need not be stored in the QuickTime wrapper; it is more important to ensure that the QuickTime metadata does not contradict the ProRes information, and to ensure that the ProRes values are reflected in the presentation. It is possible to investigate and edit atom values in the QuickTime wrapper in the same ways presented for uncompressed video.

Finally, the structure of a QuickTime or MP4 file has direct implications on access and delivery, and evaluating QuickTime files for access often takes the form of optimizing the files for streaming. It is impossible to begin playing a file without reading the entire moov atom. For this reason, files intended for streaming should be encoded with the moov atom at the beginning of the file; if the moov atom is at the end of the file, playback and scrubbing is disabled until the entire file has been downloaded.[43]

This distinction is relevant in archives, where QuickTime files are often accessioned with the moov atom at the end. This structure usually shows up when signals are captured in real time, either in digitized or raw born-digital materials; capture software cannot anticipate the length of the signal

[39] Dave Rice, personal communication, 26 Nov. 2018.
[40] Dave Rice, "Audiovisual Adherence."
[41] "MediaConch," *MediaArea,* MediaArea, 2018, https://mediaarea.net/MediaConch
[42] Dave Rice, personal communication, 26 Nov. 2018.
[43] Levkov.

it's capturing and so cannot write the moov atom until it has all of this information. Many websites that host video content, such as YouTube, will automatically transcode files to be optimized for streaming. However, archivists creating their own derivatives should export files with this in mind, and check for options for "fast start" or "streaming."

Conclusion

QuickTime is a ubiquitous file format that inspired and shaped a whole class of file formats used widely by archivists today. Knowledge of its structure, and how to manipulate it, can help troubleshoot difficulty at all levels of storage: preservation, intermediary delivery, and access. Being able to edit a QuickTime file may not come in handy on a day-to-day basis, but it will help illuminate what's happening when a file won't play because of a "missing moov atom"—and it may even provide enough knowledge to fix it.