# Glossary of Digital Video Codecs and Container Formats

## Video Codecs

### Uncompressed Video (RAW Video)

Uncompressed video refers to recording or storing each frame as full image data without any compression. It often uses color formats like **YCbCr 4:2:2 or 4:4:4** at 8-bit or 10-bit depth, which preserve all image detail[[1]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=A%20digital%2C%20color,or%20Y%27UV)[[2]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Uncompressed%204%3A2%3A2%20video%20picture%20streams,provides%20codes%20for%2030%20packed). Typical use cases include professional video capture (e.g. SDI outputs from cameras or film scanners) and lossless archiving, since no information is discarded. In preservation, uncompressed video offers maximum fidelity and avoids generation loss, but produces very large file sizes and high bitrates, requiring substantial storage and bandwidth. It is widely supported and **format-neutral**, reducing future decoding risks – essentially the video is stored as raw pixel data[[2]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Uncompressed%204%3A2%3A2%20video%20picture%20streams,provides%20codes%20for%2030%20packed). Common container formats for uncompressed video include **AVI, QuickTime (.mov), Matroska (.mkv), and MXF**, often identified by specific FourCC codes (e.g., v210 for 10-bit 4:2:2)[[2]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Uncompressed%204%3A2%3A2%20video%20picture%20streams,provides%20codes%20for%2030%20packed)[[3]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Relationship%20to%20other%20formats%20Subtype,Web%20site%20at%20this%20time). Preservationists may use uncompressed encoding for critical content (or during capture) to avoid any quality loss, though more efficient lossless codecs are often employed to save space.

### FFV1 (Lossless Video Codec)

**FFV1** is an open-source, lossless intra-frame video codec developed as part of FFmpeg, aimed at archival use. It compresses video without losing any data, meaning the decoded output is bit-identical to the original source[[4]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Lossless%20Compression%20Codecs%3AFFV1%3A%20A%20key,like%20MDPI%20for%20format%20selection). FFV1 supports various color spaces (e.g. YCbCr 4:2:2, 4:4:4, and even an alpha channel) at high bit depths, making it suitable for preservation of both **television video and film scans**[[4]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Lossless%20Compression%20Codecs%3AFFV1%3A%20A%20key,like%20MDPI%20for%20format%20selection). Typical use cases include digitization projects and archival master files – for example, many archives adopt FFV1 in a Matroska container as a preservation format to significantly reduce file size compared to uncompressed video while maintaining full quality[[4]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Lossless%20Compression%20Codecs%3AFFV1%3A%20A%20key,like%20MDPI%20for%20format%20selection). Preservation considerations: FFV1 is well-documented and now even standardized as RFC 9043, with no patent restrictions. It offers features like built-in checksums for error detection and supports **multi-threading** for faster decoding. The trade-off is larger file sizes than lossy codecs and higher computational load to encode/decode, but these are acceptable in archiving contexts[[5]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Considerations%20for%20Lossless%20Codecs%3A%20They,possibly%20greater%20computer%20processing%20capabilities). FFV1 in **Matroska (.mkv)** is recommended by various institutions (it’s a “Preferred Format” at the Library of Congress) due to its openness and robustness[[6]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Format%20Overview)[[7]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Open%20Source%20and%20Affordable). Compatible containers also include AVI and MOV, but MKV is most common for preservation projects.

### HuffYUV (Lossless YUV Compression)

**HuffYUV** is a legacy lossless video codec for Windows, originally released in the late 1990s. It employs predictive coding (similar to lossless JPEG) and Huffman encoding, hence the name[[8]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%27s%20algorithm%20is%20roughly%20the,The%20predictor%20functions%20are). HuffYUV compresses each frame independently (intra-frame), and every frame remains a keyframe, making it easy to seek and edit[[9]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=,and%20decoded%20on%20its%20own). Typical use cases were **video capture and intermediate editing**: HuffYUV was popular for digitizing analog video or exporting from NLEs without quality loss when storage and CPU power were limited. It supports YCbCr 4:2:2 and RGB color spaces (8-bit) and was optimized for speed on late-90s hardware[[10]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%20is%20a%20lossless%20video,so%20its%20main%20features%20are). In preservation, one might encounter HuffYUV in older archival files or transcoded outputs, since it was one of the first widely used lossless codecs. It is open-source (GPL) and decoders remain available, so transparency is good. However, HuffYUV lacks support for higher bit depths or modern color formats, and its compression efficiency is poorer than newer lossless codecs (resulting in larger files)[[10]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%20is%20a%20lossless%20video,so%20its%20main%20features%20are). It is most often stored in an **AVI** container, as was standard on Windows[[10]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%20is%20a%20lossless%20video,so%20its%20main%20features%20are). While largely superseded by codecs like FFV1, HuffYUV played a significant role in early digital video preservation workflows and may still be encountered in legacy collections.

### Lagarith (Lossless Video Codec)

**Lagarith** is an open-source lossless video codec (Windows-based) introduced in 2004 as a fork of HuffYUV, offering improved compression at the cost of lower encoding speed[[11]](https://en.wikipedia.org/wiki/Lagarith#:~:text=Lagarith%20is%20an%20open%20source,a%20few%20aims%20in%20mind)[[12]](https://en.wikipedia.org/wiki/Lagarith#:~:text=While%20not%20as%20fast%20as,63%2C%20and%20RGBA%20colorspaces). It supports multiple color formats – including YV12 (4:2:0), YUY2 (4:2:2), and RGB(A) – and unlike some older codecs, it avoids color space conversions to prevent any rounding errors[[13]](https://en.wikipedia.org/wiki/Lagarith#:~:text=be%20slower,each%20frame%20can%20be%20separately). Lagarith uses only intra-frame compression (each frame is independent) and produces all keyframes, which simplifies editing and cuts[[14]](https://en.wikipedia.org/wiki/Lagarith#:~:text=colorspaces.%20Keyframes%20Disallowing%20inter,joining%20and%20seeking%20much%20easier). Typical use cases included **video editing and archiving on Windows** in the 2000s for those needing smaller file sizes than HuffYUV provided (e.g., compressing screen recordings or computer-generated video losslessly). In preservation contexts, one might encounter Lagarith in collections of computer gameplay videos or fan restorations. While Lagarith yields significantly smaller files than HuffYUV and maintains full quality, it is slower to encode and decode[[11]](https://en.wikipedia.org/wiki/Lagarith#:~:text=Lagarith%20is%20an%20open%20source,a%20few%20aims%20in%20mind). It is also less widely supported outside Windows, though FFmpeg can decode it. As a preservation choice today it’s largely supplanted by FFV1 and other modern codecs, but its **open, GPL-licensed nature** means content encoded with Lagarith can still be decoded in the future. Lagarith files commonly carry the **“.avi”** extension, since it was implemented as a Video for Windows codec[[11]](https://en.wikipedia.org/wiki/Lagarith#:~:text=Lagarith%20is%20an%20open%20source,a%20few%20aims%20in%20mind).

### Motion JPEG (MJPEG)

**Motion JPEG (M-JPEG)** is a video codec where each frame is compressed *individually* as a JPEG image, with no inter-frame compression[[15]](https://en.wikipedia.org/wiki/Motion_JPEG#:~:text=Prior%20to%20the%20rise%20in,demand%20%20143). Technically, it is a simple intraframe codec: every frame is like a still photo (using DCT-based JPEG compression), so quality and compression ratio per frame are similar to JPEG still images. Typical use cases: MJPEG was widely used in the 1990s and early 2000s for **digital cameras, webcams, and non-linear editing systems**[[16]](https://en.wikipedia.org/wiki/Motion_JPEG#:~:text=M,121%20and%20Microsoft%20Edge). Many early digital point-and-shoot cameras and DSLR “movie modes” recorded video as MJPEG in an AVI or QuickTime wrapper, and some broadcast/editing systems used MJPEG for its moderate compression and easy frame-accurate editing. In terms of preservation, MJPEG’s intraframe nature is advantageous for simple decoding and editing, but it produces larger files than more modern interframe codecs for equivalent quality. It is a **lossy** codec – each frame undergoes compression, which can introduce JPEG artifacts – but because there’s no temporal compression, there are no motion artifacts and no dependency between frames. Preservation issues: MJPEG is well-documented (leveraging the ubiquitous JPEG standard) and broadly supported, so decoding it in the future is low risk. However, its compression efficiency is much lower than later formats; for example, at a given file size, MJPEG quality is typically inferior to MPEG-4 or H.264[[15]](https://en.wikipedia.org/wiki/Motion_JPEG#:~:text=Prior%20to%20the%20rise%20in,demand%20%20143). MJPEG video often appears in **AVI or QuickTime (.mov)** files, and on DVD-era hardware it was also used in some video capture cards. While largely obsolete now, archivists may encounter MJPEG in legacy camera files or edited masters, and these can be transcoded to lossless formats if needed to avoid generational loss.

### JPEG 2000 (Motion JPEG 2000)

**JPEG 2000** is an image compression standard that, in its motion form (JPEG 2000 Part 3), can be used as a video codec by encoding each frame as a JPEG 2000 image. Unlike JPEG (which is DCT-based), JPEG 2000 uses wavelet compression, offering high compression efficiency and the option of **mathematically lossless** encoding[[17]](https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf#:~:text=Archiving%20lhncbc,a%20truly%20lossless%20compression). Technical characteristics: Motion JPEG 2000 supports high bit depths (12-bit or more), wide color gamuts, and intraframe compression with no inter-frame dependency. It’s known for very good image quality, especially at high bitrates or in lossless mode, and no block artifacts (wavelet compression produces different artifact types). Typical use cases: JPEG 2000 became the standard for **Digital Cinema Packages (DCP)** – digital movie distribution – where each frame is JPEG 2000 compressed. It has also been adopted by some archives and broadcasters for **digital preservation masters** and mezzanine files[[17]](https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf#:~:text=Archiving%20lhncbc,a%20truly%20lossless%20compression). For example, archival videotape reformatting projects have used lossless JPEG 2000 in MXF wrappers as an open, standard format for long-term storage[[18]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=LC%20experience%20or%20existing%20holdings,version%20or%20profile%20or%20MXF). Preservation considerations: JPEG 2000 is an open ISO standard (ISO/IEC 15444) with no royalty requirements, which is favorable for sustainability[[17]](https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf#:~:text=Archiving%20lhncbc,a%20truly%20lossless%20compression). It provides **fully lossless** compression mode, ensuring exact preservation of the source pixels if used in that mode. However, JPEG 2000 encoding/decoding is computationally demanding, and adoption was initially slow – around 2010, few software tools and hardware devices supported it in real time, though this has improved over time. As a result, its use in broadcast archives was limited early on[[17]](https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf#:~:text=Archiving%20lhncbc,a%20truly%20lossless%20compression), but it’s now more common in niche areas like film scanning and as part of standards like **MXF Application Specification AS-07** (for preservation). Compatible container formats include **MXF** (widely used for JPEG 2000 in digital cinema and broadcasting) and the standalone **.mj2** Motion JPEG 2000 file format, though the latter is less common. Overall, JPEG 2000 offers excellent quality and longevity, and is considered a preservation-friendly codec when storage and processing resources are available.

### DV (Digital Video) – DV, DVCAM, DVCPRO Family

**DV** is a family of digital video codecs originally developed for camcorder tapes in the 1990s. It uses block-based **intra-frame DCT compression** (similar to JPEG), compressing each frame independently[[19]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=independent%20files,place%20in%20professional%20television%20production). DV operates at a fixed data rate of about 25 Mbps and typically with **4:1:1 chroma subsampling (NTSC)** or **4:2:0 (PAL)** and 8-bit depth, yielding good consumer-quality video for its time[[20]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=Clarity%20). Variants of DV were introduced for professional use: Sony’s **DVCAM** (same data rate but locked audio sync) and Panasonic’s **DVCPRO**. Higher-bit-rate extensions include **DVCPRO50** (50 Mbps, 4:2:2 chroma for broadcast SD) and **DVCPRO HD** (also known as DV100, ~100 Mbps for HD 720p/1080i video at 4:2:2)[[21]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=,data%20rate%20of%20100%20MB%2Fs). Typical use cases: DV was ubiquitous in the late 1990s–2000s for **consumer and semi-professional video recording** (MiniDV tapes) and was widely used in TV production and news gathering. Many archives hold raw DV tapes or files from this era. Preservation issues: DV is a lossy format (each frame compression discards some color detail due to 4:1:1/4:2:0 sampling and applies quantization), and it’s limited to 8-bit depth. Consequently, while DV material should be preserved in its native form if it’s the original, it is not ideal for re-digitizing higher-quality sources due to its compression and color subsampling limits. On the other hand, DV is an **openly documented standard (IEC 61834)** and widely supported in software, so decoding it is straightforward and there are no licensing hurdles[[22]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=Open%20standards,SMPTE)[[23]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=incorporated%20in%20files%20is%20also,Licensing%20and%20patents%20None%20identified). In fact, DV’s broad adoption in the 90s and 2000s means support will persist for a long time. DV files can be stored as raw **.dv streams** or wrapped in **AVI, QuickTime (.mov), or MXF** containers[[24]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=DV%20signals%20are%20formatted%20for,as%20AVI%2C%20QuickTime%2C%20and%20MXF). Notably, low-budget filmmakers sometimes used DV in QuickTime as a “preservation format” because of its small file size relative to uncompressed video[[23]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=incorporated%20in%20files%20is%20also,Licensing%20and%20patents%20None%20identified), though today’s standards favor truly lossless codecs for that role. Compatible audio is uncompressed PCM (typically 16-bit 48 kHz). In summary, DV and its extensions are **important legacy codecs**: easy to play and preserve but inherently lossy, so archives often retain DV data as is, while ensuring future migrations do not further degrade the content.

### MPEG-1 Video (H.261)

**MPEG-1 Video** (ISO/IEC 11172-2), commonly known from Video CD days, was one of the first digital video compression standards (circa 1991). It uses a *lossy*, block-based DCT compression with inter-frame prediction and was designed for about **1.5 Mbps throughput**, encoding approximately 352×240 resolution at 30 fps[[25]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Moving%20Image%20Normal%20rendering%20Good,off%20to%20make%20sure%20things)[[26]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Description%20Compression%20encoding%20for%20video,by%20%2010%2C%20QuickTime%20MPEG). Notably, MPEG-1’s design is effectively identical to the ITU-T H.261 standard (initially for video conferencing)[[26]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Description%20Compression%20encoding%20for%20video,by%20%2010%2C%20QuickTime%20MPEG). Technical characteristics: MPEG-1 supports **progressive video** and uses I-frames and P-frames (no B-frames in the base standard), with 4:2:0 chroma subsampling. Quality is moderate by modern standards – for example, a typical MPEG-1 file is **352×240 pixels** and captures only half the vertical resolution of NTSC video (each field, not full frame) to meet the bitrate target[[27]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Clarity%20%28high%20image%20resolution%29%20Moderate,compression%2C%20and%20the%20encoder%20used). Use cases: It was commonly used for **Video CDs**, early CD-ROM videos, and was extensively employed in the mid-1990s for web and multimedia content[[28]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Telecommunication%20Standardization%20Sector%29%20recommendation%20H,2%20Layer%20II%20Audio%20Encoding)[[29]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Adoption). Many early digital library projects and CD-based training videos used MPEG-1. Preservation considerations: MPEG-1 is an **open standard** with no known patent issues today (any patents have expired), and it’s widely decodable[[29]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Adoption). This makes old MPEG-1 files (e.g., .mpg or .dat on VCDs) accessible for the long term. However, the codec’s limitations in resolution and fidelity mean it does not meet modern preservation quality benchmarks except as an access copy. If an archive holds content originally encoded in MPEG-1, those files can be preserved as-is (since re-encoding won’t improve quality) but might be transcoded to a lossless format if restoration or intensive analysis is needed. MPEG-1 video is typically found in **MPEG Program Stream** files (with extensions like .mpg or .mpeg), often accompanied by MP2 audio[[30]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Tag%20Value%20Note%20Filename%20extension,2046%20Internet%20Media%20Type%20video%2Fmpv). Overall, MPEG-1’s significance is historical; it proved digital video distribution was feasible on small media, and its **wide adoption for web/PC video in the 1990s** ensures continued support in players[[29]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Adoption), even as it has been superseded by more advanced standards.

### MPEG-2 Video (H.262)

**MPEG-2 Video** (ISO/IEC 13818-2 / ITU-T H.262) is a lossy video codec that became the standard for **DVD-Video, digital cable/satellite TV, and early HDTV broadcasting**. It enhances MPEG-1 by supporting higher resolutions (SD and HD), interlaced video, and more advanced coding tools (including B-frames for better compression). Technical details: MPEG-2 is a **DCT-based, inter-frame codec** using I, P, and B frames with motion compensation[[31]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Description%20This%20description%20pertains%20to,Each%20PES%20packet%20includes%20a). Common profiles include Main Profile @ Main Level (used for standard-definition DVDs at around 720×480/576, 4:2:0 chroma) and Main Profile @ High Level or 4:2:2 Profile (used in studio applications, supporting higher resolution or 4:2:2 chroma)[[32]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Adoption)[[33]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=MPEG,at%20this%20time%3A%20SNR%20Scalable). MPEG-2 video typically is 8-bit and often 4:2:0 in distribution, although a 4:2:2 profile exists for broadcast production. Use cases: It was **widely adopted** – for example, DVD-Video uses MPEG-2 at about 4–9 Mbps, and broadcast TV systems like ATSC (US) and DVB (Europe) made MPEG-2 mandatory for their first-generation digital video[[34]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Widely%20adopted%20for%20filemaking%2C%20DVD,exist%20for%20encoding%20and%20decoding). Many TV archival files from the 2000s are in MPEG-2 (including formats like IMX 50, an intraframe 50 Mbps 4:2:2 MPEG-2 used by broadcasters). Preservation considerations: MPEG-2 is an international open standard, thoroughly documented and broadly supported in hardware and software[[35]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Disclosure%20Open%20standard,Adoption)[[36]](file://file-AH8UoofHcWxUPBGCWjywFQ#:~:text=video%2C%20typically%2050%20Mbps%20for,supported%20by%20the%20vendor%20community). This widespread adoption and standardization mean that decoding MPEG-2 in the future is very likely to be possible, reducing obsolescence risk. However, as a *lossy* codec, it does introduce compression artifacts and some loss of detail, especially at lower bitrates. For preservation, this means one wouldn’t choose to compress new master files in MPEG-2 today, but one may encounter a lot of MPEG-2 content in collections (from DVDs or broadcast masters). Archives often preserve such content in its original form to avoid generational loss, while noting that MPEG-2’s quality is lower than newer codecs at a given bitrate. One advantage is that high-bitrate MPEG-2 (e.g., 50 Mbps I-frame) can be nearly visually lossless and was sometimes used as an intermediate format for archive reformatting[[37]](file://file-AH8UoofHcWxUPBGCWjywFQ#:~:text=MPEG,2). Compatible containers: MPEG-2 video is typically stored in **MPEG-2 Program Streams** (with extension .mpg or on DVD as .vob) or in **Transport Streams** (.ts, e.g., for broadcast or AVCHD). It can also be wrapped in **MXF** for professional workflows[[38]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Spatially%20Scalable%2C%20and%20High,and%20Layer%20II%20Audio%20Encoding), or even in modern containers like Matroska. In summary, MPEG-2 video is a **cornerstone of digital video history**, with a balance of efficiency and quality that was state-of-the-art in the 90s, and it remains important for accessing a huge amount of legacy media content.

### MPEG-4 Part 2 (Visual Coding – DivX/Xvid)

**MPEG-4 Part 2 Visual** is a video compression standard developed in the late 1990s, known for implementations like **DivX, Xvid**, and Apple’s QuickTime MPEG-4. It is a successor to MPEG-2 in the MPEG family, using similar DCT compression but adding features like object-based encoding modes. In practice, the most common profiles (Simple and Advanced Simple Profile) of MPEG-4 Part 2 are fairly similar to H.263 baseline coding[[39]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Compression%20encoding%20for%20video%20identical,264%20video). Technical characteristics: It’s a lossy, inter-frame codec supporting **I-frames and P-frames** (and optionally B-frames in Advanced Simple Profile), with **4:2:0 chroma** and up to 8-bit depth. It improved compression efficiency over MPEG-2 for lower bitrates and smaller resolutions, making it popular for early internet video. Typical use cases: MPEG-4 Part 2 was widely adopted in the early 2000s for **downloadable video and computer playback**. The codec achieved fame via **DivX and Xvid** – codecs that could compress DVD-resolution video into much smaller files (a “DVD rip” could be 700 MB, for example). These were commonly stored in AVI files. It was also supported on some portable devices and early **IPTV** systems, and the video-capable iPod (2005) played back Simple Profile MPEG-4 video[[40]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Beginning%20in%20about%202005%2C%20there,See%20also%20%2022). Preservation issues: MPEG-4 Part 2 is an open standard, but it did not see as wide professional adoption as its sibling **H.264 (MPEG-4 Part 10)**, and by the mid-2000s it was largely overtaken by H.264 in new applications[[41]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Adoption). Many archives will encounter content in DivX/Xvid form, especially in personal digital collections from that era or early streaming/download services. Decoding support remains abundant (FFmpeg, etc.), so access is not a problem. Quality-wise, it’s *lossy* and roughly on par with or slightly better than MPEG-2 for a given bitrate; however, H.264 offers significantly better efficiency, so by comparison MPEG-4 Part 2 video looks more artifact-prone at low bitrates[[42]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=MPEG,at%20a%20lower%20data%20rate). Thus, for preservation, one would not transcode into MPEG-4 Part 2, but one should preserve original files encoded in it. Compatible container formats include **AVI (very common for DivX/Xvid)**, **MP4/MOV** (MP4 supports MPEG-4 Visual as a codec), and **OGM/Matroska** in some cases. In summary, MPEG-4 Part 2 (often just called “MPEG-4 ASP” for Advanced Simple Profile) was a transitional codec – widely used in its time (e.g., in millions of AVI files) but now largely legacy, overshadowed by more efficient H.264/AVC[[43]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Compression%20encoding%20for%20video%20identical,264%20video)[[41]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Adoption).

### H.264 / MPEG-4 Part 10 (AVC)

**H.264**, also known as **MPEG-4 Advanced Video Coding (AVC)**, is a widely used lossy codec that became the de facto standard for high-quality video in the 2008–2020 period. Technically, H.264 is a block-based codec with motion compensation like its predecessors, but it introduced new features such as *adaptive entropy coding (CABAC)*, *multiple reference frames*, *advanced intra prediction*, etc., enabling much higher compression efficiency than earlier standards[[42]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=MPEG,at%20a%20lower%20data%20rate). It supports a range of profiles: from Baseline (low-complexity, no B-frames) up to High 4:4:4 Predictive, which allows for 4:4:4 chroma and up to 14-bit depth. Commonly, **8-bit 4:2:0 High Profile** was used for most consumer content. Use cases: H.264 became ubiquitous – used in **Blu-ray Discs, streaming platforms, broadcast HDTV, DSLRs and smartphones** for recording, and many archival mezzanine files. For example, YouTube and Vimeo adopted H.264 for streaming around 2007–2015, and virtually all consumer devices could play H.264 MP4 files. It’s also used in production pipelines (often as AVC-Intra, an all-intra variant for cameras, or XAVC). Preservation considerations: H.264 is an **international open standard (ITU-T and ISO)**, so documentation is available, but it is heavily patented. Decoding and encoding require a license, though open-source implementations (x264, FFmpeg) have thrived. Long-term, the patent situation meant some hesitation in archival circles to name H.264 as a preservation format, but its ubiquity means support will remain high[[44]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=Adoption%20The%20underlying%20encoding%2C%20sometimes,in%202005%3B%20adoption%20not%20determined)[[45]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=and%20an%20encoding%20variously%20listed,4_AVC_BP). Being *lossy*, H.264 is not ideal for multi-generational preservation (each re-encode loses quality), and it’s computationally complex to encode/decode compared to older codecs. However, if an archive receives camera files or mastered content in H.264, those are often retained as-is given the practicality. H.264 offers excellent quality at relatively low bitrates; for example, an H.264 file at a few Mbps can surpass MPEG-2 at twice the bitrate in quality[[42]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=MPEG,at%20a%20lower%20data%20rate). It also supports **intra-only modes** (used for high-end recording) which can be near lossless. Compatible containers: most commonly **MP4** or **QuickTime (.mov)**, and also **MPEG-2 TS** for broadcast or AVCHD (MTS files), and **Matroska (.mkv)** for various content[[46]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=Relationship%20to%20other%20formats%20Used,ISO%2FIEC%20TS%2030135%3A2014). H.264 was generally a final distribution or production-delivery format (it’s even used in IMF masters in lossless mode for some studios), though for long-term archiving of valuable content, truly lossless formats are preferred. In summary, H.264/AVC is a **cornerstone codec**, combining high efficiency with broad hardware support; it will remain important for accessing a vast amount of video created in the last two decades[[44]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=Adoption%20The%20underlying%20encoding%2C%20sometimes,in%202005%3B%20adoption%20not%20determined).

### H.265 / HEVC (High Efficiency Video Coding)

**H.265**, also called **HEVC (High Efficiency Video Coding)**, is the successor to H.264, offering roughly **50% better compression** at the same quality level[[47]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=One%20of%20the%20main%20advantages,for%20hardware%3B%20capable%20of%20consistent). Approved in 2013, HEVC incorporates many advances: a flexible quadtree coding structure (allowing very large or small blocks), improved motion prediction and intra prediction, support for higher bit depths and **full 4:4:4 chroma**, and more parallelism in decoding[[48]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=and%20intra%20,HEVC%20in%20a%202017%20video)[[49]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=HEVC%20is%20a%20important%20advance,varied%20test%20set%20of%20video). Technical characteristics: HEVC can handle up to 8K resolutions, defines profiles for 8-10 bit video (Main and Main10) and beyond (up to 14-bit in High Throughput profiles), and includes profiles for screen content. It’s a lossy codec (with an unofficial lossless mode available) that, in practice, achieves about 25–50% bitrate savings over H.264 for similar objective quality[[50]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=of%20the%20reference%20software%20for,improvement). Use cases: HEVC is used for **Ultra HD (4K) broadcasting and streaming**, on 4K Blu-ray discs, and in some camera formats. For example, many 4K streaming services (Netflix, etc.) and broadcasters (DVB 4K) adopted HEVC in the late 2010s, and Apple made HEVC the default for iPhone video recording in 2017 (for space-saving)[[51]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=MPEG,a%20desired%20image%20or%20video)[[52]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=The%20formation%20of%20AOMedia%20and,HEVC%20was). Preservation issues: HEVC is an **open standard** technically, but it is *heavily encumbered by patents* – multiple patent pools have made licensing complex and expensive[[53]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=The%20underlying%20HEVC%20encoding%20is,Joins). This slowed its adoption in some domains (e.g. web browsers largely avoided HEVC in favor of AV1 due to royalties). For archives, the patent issues pose a risk: relying on HEVC might entail legal or cost barriers for long-term use in open-source tools. Nevertheless, HEVC decoding is becoming common (many hardware decoders exist, and FFmpeg supports it), so access to content is feasible. HEVC is likely to be present in collections of **born-digital 4K videos** and user-generated content from the late 2010s. Its compression is lossy, so, like H.264, archives wouldn’t choose it for intermediate/store-of-record encoding when lossless options exist. But they may need to preserve content that arrives in HEVC (ensuring they maintain high-quality profile files). Containers: typically **MP4 (.mp4)** or **MPEG-TS**/.m2ts for HEVC (e.g., 4K Blu-rays), and **Matroska/WebM** (though WebM opted for VP9/AV1, some use MKV for HEVC). Apple’s **MOV** also supports HEVC (e.g., .MOV files from iPhones). In summary, HEVC represents the state of the art (circa 2015) in compression efficiency, **supporting higher fidelity video (up to 4:4:4 12-bit) with smaller file sizes**[**[49]**](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=HEVC%20is%20a%20important%20advance,varied%20test%20set%20of%20video), but at the expense of complexity and patent concerns. As of 2025, it’s widely used in industry but archives are cautiously evaluating royalty-free alternatives for long-term use.

### AV1 (AOMedia Video 1)

**AV1** is a next-generation video codec released in 2018 by the Alliance for Open Media, designed to be **open and royalty-free** as an answer to HEVC’s patent issues[[52]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=The%20formation%20of%20AOMedia%20and,HEVC%20was). It is a lossy codec that builds upon Google’s VP9 advances and introduces new coding tools to achieve very high compression efficiency. In tests, AV1 can deliver **30-50% better compression** than H.264 and about 30% better than VP9 at the same quality[[47]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=One%20of%20the%20main%20advantages,for%20hardware%3B%20capable%20of%20consistent). Technical characteristics: AV1 supports up to 12-bit color depth and a **4:4:4 Professional profile** for lossless-quality encoding, as well as 8 or 10-bit 4:2:0 profiles for typical media[[54]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=,than%20or%20equal%20to%202). It uses techniques like larger transform blocks, adaptive quantization matrices, warp motion, and combined predictive modes from multiple reference frames. It also incorporates film grain synthesis metadata (so grain needn’t be coded as data). Typical use cases: AV1 has been adopted by **web streaming platforms** (e.g., YouTube, Netflix, Facebook) to reduce bandwidth costs – especially for high resolutions or on data-constrained networks. By 2025, many modern browsers and mobile devices support AV1 decoding, and it’s being used in some videoconferencing solutions. While not much content in archives is natively AV1 yet, it’s expected to grow for **user-generated content and web video**. Preservation aspects: being **royalty-free and openly documented**, AV1 is very attractive for long-term sustainability[[52]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=The%20formation%20of%20AOMedia%20and,HEVC%20was). There are no licensing fees to worry about, which encourages wide adoption. However, AV1 is extremely computationally intensive to encode (significantly slower to encode than H.264/H.265), which might be a factor for archives considering transcoding to AV1 for access copies. For master preservation, AV1 is still a lossy codec – one could use its lossless mode or high-bit-depth profile, but that’s not its primary design goal. It’s more suited to replacing H.264/H.265 for distribution. Nonetheless, if an archive’s incoming content is AV1 (perhaps from screen recordings or future camera formats), it can be preserved as-is with confidence in decode availability. Container format: AV1 is commonly stored in **Matroska/WebM** containers (WebM is essentially Matroska restricted to VP8/VP9/AV1 video and Opus/Vorbis audio)[[55]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=Relationship%20to%20other%20formats%20Modification,Not%20documented%20on%20this%20resource). The MP4 format has also defined support for AV1 (ISO BMFF codec code ‘av01’), so .mp4 files with AV1 exist as well. In summary, AV1 is an **important modern codec** combining top-tier compression with an open license[[56]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=Nonetheless%2C%20the%20HEVC%20patent%20situation,See%20HEVC%20for%20more%20details)[[47]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=One%20of%20the%20main%20advantages,for%20hardware%3B%20capable%20of%20consistent). For archives, its arrival is promising for creating high-quality access copies without patent worries, though its role for masters is likely limited to cases where only a lossy compressed source is available.

### Theora (Xiph.Org Video Codec)

**Theora** is a free and open lossy video codec developed by the Xiph.Org Foundation, released in 2004 as an open alternative to proprietary formats[[57]](http://www.linux-magazine.com/Online/News/Google-funds-OGG-Theora#:~:text=The%20OGG%20Theora%20codec%20is,2%2C). It is derived from On2 Technologies’ VP3 codec (which was open-sourced) and can be viewed as roughly comparable to MPEG-4 Part 2 or H.263 in capability. Theora uses DCT-based intra-frame and inter-frame compression and typically operates in **4:2:0, 8-bit**. Use cases: Theora was embraced in the mid-2000s by open-source communities and was notably used for **Wikipedia videos and other open content** when HTML5 was emerging – it was supported natively in early Firefox and Chrome builds as the video format for HTML5 <video> before H.264 or WebM took over[[57]](http://www.linux-magazine.com/Online/News/Google-funds-OGG-Theora#:~:text=The%20OGG%20Theora%20codec%20is,2%2C). It was often paired with Vorbis audio in an Ogg container (.ogv files for video). Preservation considerations: Theora is unencumbered by patents and can be implemented by anyone, aligning with archival ideals of openness. However, it never achieved compression efficiency parity with the best proprietary codecs of its time – for a given bitrate, Theora’s quality is similar to or a bit below that of contemporaneous MPEG-4 or H.264 Baseline. As a result, industry adoption stagnated once H.264 became ubiquitous, and even Xiph’s own efforts shifted toward supporting Google’s VP8/VP9. Nevertheless, Theora content in collections (from, say, open culture projects around 2005–2012) remains playable – support exists in VLC, FFmpeg, and all major open-source players. Because it’s an openly documented format, long-term decode is not a concern. Theora is rarely chosen for new encodings now, but historically it’s significant as one of the first successful royalty-free video codecs. Compatible container: **Ogg** is the primary container for Theora video[[58]](https://wiki.xiph.org/Theora#:~:text=Theora%20is%20a%20video%20codec%2C,future%20scope%20for%20encoder) (hence .ogv files), though technically Matroska could also carry Theora. In summary, Theora represents a **deprecated but historically important codec**: it demonstrated the viability of open video compression and paved the way for later codecs like VP8 and AV1. Archives might encounter it in open-license video files from the late 2000s and can preserve or transcode them knowing there are no legal hurdles to using the content.

### VP8 (WebM Video Codec)

**VP8** is a royalty-free video codec introduced by On2 Technologies and released by Google in 2010 as part of the WebM project. It’s a direct competitor to H.264 Baseline/Main in terms of compression efficiency. VP8 is a block-based, inter-frame codec (with 4:2:0 chroma, 8-bit depth) that achieves roughly the same quality as H.264 **Main Profile** at a slightly lower bitrate[[47]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=One%20of%20the%20main%20advantages,for%20hardware%3B%20capable%20of%20consistent). Use cases: VP8 was adopted for **HTML5 web video** – Google Chrome, Firefox, and Opera browsers all supported VP8 within WebM container for playing <video> content without plugins. YouTube offered VP8 WebM streams for many videos in the early-to-mid 2010s. It was also used in WebRTC for real-time video calls. Preservation considerations: VP8, being open and royalty-free, avoids licensing concerns[[52]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=The%20formation%20of%20AOMedia%20and,HEVC%20was). It saw significant adoption in the tech community, which means decoders for VP8 are widespread (FFmpeg’s libvpx, hardware support in some chips, etc.). As a result, long-term accessibility of VP8 content is very good – it can be viewed as the **“open H.264” of its era**. In terms of quality, VP8 is lossy and a generation behind the latest codecs (it’s less efficient than HEVC or AV1). Archives likely won’t use VP8 to encode new preservation masters, but may very well have VP8-encoded files in web archives or donated collections of digital footage from the 2010s. Such files can usually be kept as-is or transcoded to a lossless format if re-use/editing is needed, to avoid cumulative loss. The primary container for VP8 is **WebM (Matroska variant)** – .webm files typically contain VP8 video and Vorbis or Opus audio[[55]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=Relationship%20to%20other%20formats%20Modification,Not%20documented%20on%20this%20resource). VP8 can also be found in standard .mkv files and even in rare cases in MP4 (though MP4 didn’t officially support it widely). As an example of its role: when RealNetworks and others were looking for alternatives to proprietary codecs, VP8 was highlighted as a new open option around 2011[[59]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=alternatives.%20For%20example%2C%20MPEG,and%20RealAudio%2C%20and%20for%20related). In summary, VP8 is an **archivally friendly codec** from a licensing standpoint and is part of many modern software stacks, but it represents a generation of lossy compression suitable for distribution rather than archival mastering. It holds historical importance as the codec that pushed the industry toward open video with the WebM format.

### Windows Media Video / VC-1

**Windows Media Video (WMV)** refers to a series of Microsoft’s proprietary video codecs, culminating in **WMV9**, which was standardized as **SMPTE VC-1**. VC-1 is a DCT-based inter-frame codec in the same class as H.264’s contemporaries, offering efficient compression and support for interlaced content[[60]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Wikipedia%20VC,broadcast%20and%20video%20industry%20professionals). Technical outline: VC-1 supports advanced profiles that include B-frames, interlaced frame coding tools, and 4:2:0 color at up to 1080p/level 3. Its design evolved from earlier WMV versions and is comparable in quality to H.264 Baseline/Main at a given bitrate. Use cases: VC-1 was notably used in **HD DVD and Blu-ray Discs** as a mandatory codec (many Blu-rays contain VC-1 video)[[61]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=According%20to%20the%20Wikipedia%20VC,free%20implementation%20of%20this%20codec). It was also the codec behind many **.wmv files** used in the early-mid 2000s for web streaming and downloads (e.g., movie trailers on Microsoft’s sites, or third-party streaming services before H.264 took over). Microsoft’s Silverlight and the Xbox 360 supported VC-1 for video playback/downloads[[62]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=According%20to%20the%20Wikipedia%20VC,free%20implementation%20of%20this%20codec). Preservation: As VC-1 was standardized (SMPTE 421M), its decoding specification is published and can be implemented by anyone[[63]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Disclosure%20Decoding%20specification%20is%20an,Adoption). This means that despite being born proprietary, VC-1 is now openly documented, and tools like FFmpeg can decode it. Patents do apply (there’s a patent pool with multiple companies), but by now VC-1 is declining in use. Many archives will encounter **WMV files** from government agencies, education, or personal collections created around 2003–2010. These typically use VC-1 or its earlier cousins. Decoding and migrating those to newer formats is straightforward. Quality-wise, VC-1 at high bitrates was considered very good (some broadcasters used VC-1 for mezzanine files, and Microsoft touted it as on par or better than MPEG-4 Visual). It supports **interlaced video compression natively**, which made it attractive for broadcast professionals transitioning from MPEG-2[[64]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=H.261%2C%20H.263%2C%20MPEG,broadcast%20and%20video%20industry%20professionals). In terms of container, VC-1 is usually packaged in **ASF containers** (with .wmv extension) for files, or in **MPEG-TS** for Blu-ray and streaming. One preservation concern is that VC-1 never achieved the ubiquity of H.264; thus, long-term, there are fewer hardware decoders produced after its prime. But software support remains, and documentation exists to reimplement if needed. Also, any **DRM** that was sometimes applied to WMV files (Windows Media DRM) could complicate preservation – the files may need to be rewrapped or the DRM stripped, as keys might be unobtainable in the future. Otherwise, as a standard, **VC-1/WMV9 is well-documented and broadly supported**, and the Library of Congress even lists WMV (VC-1) as acceptable for video preservation if re-encoding isn’t feasible[[65]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Adoption)[[66]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Licensing%20and%20patents%20According%20to,exclusive%20licensing%20body). In summary, VC-1/WMV is a **once-popular format** bridging the DVD and H.264 era: it’s an important part of collections from that timeframe and can be preserved or transcoded with relatively low risk.

### RealVideo (RealNetworks Codec Series)

**RealVideo** is a proprietary video compression format (actually a series of codecs: RealVideo 1.0 through 10) developed by RealNetworks for streaming media. RealVideo was a dominant web video format in the late 1990s and early 2000s, commonly delivered in **.rm or .rmvb (RealMedia)** container files via the RealPlayer. Technical characteristics: early RealVideo versions (RV10, RV20) were based on H.263, while later ones (RV30, RV40 corresponding roughly to RealVideo 8,9,10) introduced more complex, non-standard algorithms (RealNetworks literature claims they are not purely block-based like MPEG-2)[[67]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Normal%20rendering%20Good%20support,and%20the%20encoding%20option%20selected). The codecs were highly tuned for low bitrate streaming over dial-up connections, and they often ran at reduced frame sizes or rates to accommodate bandwidth. Use cases: RealVideo was **widely adopted for streaming web content** (news clips, proprietary content portals, etc.) starting in the mid-90s[[68]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Adoption%20Widely%20adopted%20for%20streaming,A%20new%20option%20is). For example, many early internet TV services and some content from sites like the BBC and CNN were offered in RealMedia format. In many parts of Asia, RealVideo in RMVB (variable bitrate RealMedia) format was popular for distributing dramas and movies due to its compression efficiency at low bitrates. Preservation issues: RealVideo is **proprietary with limited public documentation**, and decoding support historically was closed-source (RealPlayer). However, eventually FFmpeg and others reverse-engineered later RealVideo codecs, and today tools can decode most RealVideo files. RealNetworks also provided source access to some developers via the Helix community, which somewhat opens it[[69]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Disclosure%20Proprietary%20format%20with%20little,Although%20those%20developing%20HTML5)[[70]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Licensing%20and%20patents%20Licensing%20for,source%20code%20for%20the%20codecs). The format is undeniably *obsolete*: usage declined steeply once more open or standard formats (H.264, etc.) became available and after RealPlayer fell out of favor[[68]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Adoption%20Widely%20adopted%20for%20streaming,A%20new%20option%20is). For archives, this means RealVideo files should be transcoded for access if needed, but also kept in original form if they’re the source – and ensuring a copy of the last RealPlayer or an FFmpeg-based player is available for future decoding is wise. RealMedia container (.rm/.rmvb) can include some metadata – e.g. title, author, etc., and **supports multiple bitrates in one file (SureStream)** for adaptive streaming[[71]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Full%20name%20RealVideo%2C%20Version%2010,from%20version%2010%20only%20in). Typically, a RealMedia file contains one video and one or more audio streams (often RealAudio or later AAC). One interesting preservation note is that RealMedia files often rely on the concept of a “**metafile (.ram)**” to stream – an external pointer to the .rm – but for local files the .rm/.rmvb stands alone. In practice, archives might find .rmvb files with higher quality (near-VHS/SD quality) for content that circulated around 2005-2010 especially. Ensuring these can be played (by converting to, say, MP4 H.264) is recommended, as RealVideo decoders may not be common in consumer tools going forward. Nonetheless, given that RealNetworks has ceased pushing its format and openly documented alternatives exist, the **need for proprietary streaming solutions like RealVideo has vanished**[[68]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Adoption%20Widely%20adopted%20for%20streaming,A%20new%20option%20is). RealVideo’s legacy is a reminder of the early internet video era, and archivists should be on the lookout for .rm files in old web archives or personal archives, recognizing that those represent a once “cutting-edge” attempt at low-bitrate video delivery.

### Apple ProRes (422 and 4444 Families)

**Apple ProRes** is a family of proprietary, high-quality intermediate video codecs introduced by Apple in 2007. ProRes is *intra-frame only* (each frame compressed independently) and designed to maintain excellent fidelity for **post-production workflows** while significantly reducing file size compared to uncompressed video[[72]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=Apple%20ProRes%20is%20a%20family,Apple%20ProRes%204444%20Codec%20Family)[[73]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20was%20designed%20to%20be,the%20majority%20of%20FCP%20users). Technical characteristics: The ProRes 422 family compresses 10-bit video at 4:2:2 chroma (the ratio “422” in the name) and supports any frame size from SD up to 5K[[74]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=The%20key%20character%20traits%20that,422%20family%20are%20support%20for). Variants include ProRes 422 Proxy, LT, Standard, HQ – differing in bitrate and resulting quality. The ProRes 4444 family adds support for **4:4:4:4** (full color and alpha channel) at 12-bit depth, aimed at graphic and film workflows. ProRes uses a proprietary DCT-based compression that is lossy but visual-lossless at typical bitrates (e.g., ProRes 422 HQ for 1080p is around 220 Mbps). Use cases: ProRes became *ubiquitous in professional video production*. It is the format of choice for many camera recorders, editing, and color grading workflows (especially in the Mac ecosystem but also widely supported on Windows via licensed encoders). A production might shoot in camera raw or another codec but then convert to ProRes for editing; many final masters for broadcast or archiving by studios are delivered as ProRes files. Preservation issues: ProRes is **proprietary** – for years, Apple kept tight control; decoding was widely supported (even FFmpeg can decode ProRes) but encoding was officially allowed only in Apple products or via special licensing. In 2015, Apple published a *Registered Disclosure Document* with SMPTE for ProRes[[75]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=While%20detailed%20technical%20information%20on,bitstream%20syntax%2C%20the%20bitstream%20element), partially documenting the bitstream for decoding. This improves its transparency for preservation (decoders can be reimplemented from the spec)[[76]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=While%20detailed%20technical%20information%20on,available%20for%20a%20fee). Still, it’s not an open standard, and it’s lossy, though high bitrate. Archives receive a lot of content in ProRes (e.g., mezzanine files from video producers or digitization vendors), and they often choose to keep it in that format to avoid a generation loss that re-encoding to another codec might cause. ProRes at high quality profiles (422 HQ or 4444 XQ) is considered **visually lossless**, and in fact some archives treat ProRes HQ as an acceptable preservation format for video when truly lossless options result in too much data[[77]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20codecs%20are%20usually%20contained,See%20Notes%20for%20more%20information). Compatible container: ProRes is almost exclusively wrapped in **QuickTime (.mov)** files[[77]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20codecs%20are%20usually%20contained,See%20Notes%20for%20more%20information). (Only recently, as of 2016, Apple allowed ProRes in MXF for broadcast delivery[[77]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20codecs%20are%20usually%20contained,See%20Notes%20for%20more%20information), but .mov remains dominant.) These QuickTime files can contain metadata, timecode, and audio along with the ProRes video. In summary, ProRes is a **mastering codec** balancing quality and size: it’s *not bit-perfect* to the source but good enough to round-trip through post-production multiple times. From an archiving standpoint, its widespread adoption in industry and the availability of decoders means content in ProRes can be preserved and accessed for the foreseeable future[[78]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=but%20rather%20,which%20details%20the%20frame_size%20and). The main caution is that it’s proprietary; however, given its momentum, ProRes appears to be a “born-digital” format that archives will continue to accommodate (similar to how TIFF is accepted for images despite being proprietary in origin).

### Avid DNxHD / DNxHR (SMPTE VC-3)

**Avid DNxHD** (and its extended variant **DNxHR** for higher resolutions) is a family of lossy intra-frame codecs developed by Avid Technology for professional video editing. It was designed as an open alternative to ProRes; Avid submitted DNxHD to SMPTE, which standardized it as **VC-3**[[79]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=Avid%20DNxHD%20%28,3%20standard.%5B%201). Technical characteristics: DNxHD data is 8-bit or 10-bit 4:2:2 compression for HD video (up to 1920×1080)[[80]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=DNxHD%20is%20a%20video%20codec,in%20a%20%2059%20container). Each frame is DCT-compressed independently (no inter-frame compression), at various fixed bitrates (e.g., 36, 120, 185 Mbps for 1080p). DNxHR (introduced 2014) adds support for 4K and higher resolutions and more flavors (including HQX for 12-bit 4:2:2, and DNxHR 444 for 4:4:4 content). Use cases: DNxHD was (and is) heavily used in **broadcast production and post**. It enabled real-time editing of HD footage on moderate hardware, similar to ProRes but favored in AVID Media Composer environments and by some camera recorders. Many TV networks accepted DNxHD 145 (145 Mbps) files as masters, and it was common in tapeless workflows. For archiving, institutions may have preservation masters or mezzanine files as DNxHD – for example, some digitization operations capture analog tapes as DNxHD 220 Mbps files in an MXF wrapper. Preservation considerations: A key aspect is that DNxHD was intended to be an *open* standard; SMPTE VC-3 specifications are published, meaning the codec is documented and not solely reliant on Avid’s implementation[[81]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=On%20February%2013%2C%202008%2C%20Avid,3%20standard.%5B%202). In practice, Avid initially licensed DNxHD encoding—FFmpeg could decode it, but encoding was proprietary until Avid allowed an open implementation. Over time, DNxHD/DNxHR became more open and now are widely supported (FFmpeg, Resolve, etc. can encode it under certain licenses). So, sustainability is good: it’s standardized and broadly adopted in professional circles. As a *visually lossless* intra-frame codec, it’s suitable for preserving content with minimal quality loss, though strictly speaking it is still lossy. Many archives consider DNxHD at high bitrate an acceptable preservation format if truly lossless options are impractical, because it’s well-understood and preserves full HD detail to the human eye. In fact, the Library of Congress has many files encoded in DNxHD as part of certain collections[[18]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=LC%20experience%20or%20existing%20holdings,version%20or%20profile%20or%20MXF). MXF is the most common container for DNxHD/DNxHR (particularly OP1a MXF in broadcast), though QuickTime .mov is also used (Avid systems often can import/export DNx in mov)[[80]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=DNxHD%20is%20a%20video%20codec,in%20a%20%2059%20container). DNxHR, being newer, is increasingly used for 4K mezzanines and is similarly MXF-friendly. In summary, DNxHD/DNxHR (VC-3) codecs are **preservation-friendly intermediate formats**: open documentation, no playback royalties, and designed for multi-generation use with minimal degradation. Archives receiving Avid projects or network masters will often store the DNxHD files as-is, confident that the format will be supported long-term and that it meets a high quality bar. The only downside is large file size (compared to inter-frame codecs) – the trade-off for easier editing and decoding[[79]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=Avid%20DNxHD%20%28,3%20standard.%5B%201)[[80]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=DNxHD%20is%20a%20video%20codec,in%20a%20%2059%20container).

## Container Formats

### QuickTime File Format (.mov)

**QuickTime File Format (QTFF)** is a general-purpose multimedia container developed by Apple, widely used on both macOS and Windows. A QuickTime **“.mov”** file can wrap video, audio, text (subtitles), and other tracks in a structured, hierarchical format of atoms (boxes)[[82]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=File%20format%20that%20wraps%20video%2C,versions%207%20and%208)[[83]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Specialists%20describe%20QuickTime%20atoms%20as,type%2C%20and%20a%20data%20payload). Structural characteristics: QuickTime’s atom-based design is highly extensible – each track or piece of data is contained in an *atom* with a type and length, and atoms can nest (for example, a movie atom contains track atoms, which contain media atoms, etc.)[[83]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Specialists%20describe%20QuickTime%20atoms%20as,type%2C%20and%20a%20data%20payload). This allows QuickTime to support a wide range of media types (video codecs from uncompressed to ProRes to H.264; audio from PCM to AAC; even still images or VR panoramas) within the same format. It also has robust support for **metadata and timecode**. QuickTime files include a “moov” atom (header) which can hold significant metadata: e.g., it supports **user data or annotations** for things like title, comment, copyright, creation date, and more[[84]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=MJPEG%20%2C%20%2036%2C%20AAC,documentation). It also can embed timing information like edit lists and synchronization across tracks. Common applications: QuickTime became the default for **professional video software (Final Cut Pro, Adobe Premiere, Avid, etc.)** for many years, and many cameras (especially early DSLRs and phones) produced .mov files. It’s used in editing and **production workflows** (e.g., ProRes or DNxHD in .mov for exchange) as well as in **end-user delivery** for some platforms (though MP4 has largely taken that role now). Workflows often treat .mov as a “master” wrapper for high quality content. In archiving, Apple’s ProRes guidelines and widespread support have made .mov a preferred wrapper for production-quality files (and indeed LoC’s Recommended Formats lists ProRes 422/HQ in QuickTime as a preferred file format for video)[[85]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=LC%20experience%20or%20existing%20holdings,Digital%20And%20Physical%20Media). Preservation pros and cons: QuickTime is **proprietary in origin but fully documented** (Apple published specifications, and the format is the basis of the ISO Base Media File Format used by MP4)[[86]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=As%20of%20February%202012%2C%20two,the%20file%20wrapper%20per%20se)[[87]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Disclosure%20Fully%20documented,See%20information%20on%20the). Therefore, there’s strong confidence in its longevity; many parsing libraries exist. It’s a binary format but relatively transparent in structure. One benefit is the rich metadata support – QuickTime files can carry embedded descriptive metadata (title, director, etc.) in their headers, which is useful for self-documentation[[88]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Technical%20metadata%20is%20in%20the,Writer). Also, QuickTime can handle timecode tracks, which is crucial for preserving original timing and reel information. As for capabilities: it supports **multiple tracks** of any type (e.g., multiple audio languages, video angles, subtitles) though in practice it’s less used for subtitles compared to Matroska or MP4. Transparency/complexity: QuickTime is slightly more complex than say AVI, but tools to handle it are plentiful. One minor preservation concern was that some aspects (like certain older proprietary codecs in .mov) require specific software, but the container itself is stable. QuickTime has also been extended over time (e.g., to support newer color information or codecs), and those extensions are usually forward-compatible or at least safely ignored by readers. In summary, QuickTime is a **versatile, well-established container**. It has been called the “Swiss army knife” of containers in production. For archiving, a QuickTime file with a common codec (like ProRes, DV, Motion JPEG, etc.) is generally considered a safe and well-understood choice[[87]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Disclosure%20Fully%20documented,See%20information%20on%20the)[[89]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Transparency%20The%20file%20format%20is,documentation). Its influence is seen in MP4 (which is basically a stricter form of QuickTime), so knowledge and tools often apply to both.

### AVI (Audio Video Interleaved)

**AVI** is a long-standing container format introduced by Microsoft in 1992 as part of the Video for Windows platform. It is based on the **RIFF** (Resource Interchange File Format) structure and is designed to interleave audio and video data for synchronous playback[[90]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=File%20format%20for%20moving%20image,the%20data%20chunks%20within%20the). Structural characteristics: An AVI file contains a RIFF header identifying it as AVI, a header chunk (hdrl) with metadata like frame rate, width, height, codec FourCCs, etc., and a data chunk (movi) that contains audio and video frames interleaved in sequence[[91]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=File%20format%20for%20moving%20image,A%20third%20optional). Optionally, an index chunk (idx1) at the end provides quick seek access to frame positions[[92]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=subchunk%20is%20tagged%20movi%20and,data%20chunks%20within%20the%20file). AVI is relatively simple: it doesn’t inherently support features like subtitles or chapters (those would be separate streams possibly, but rarely used in AVI), and originally it had a 2 GB file size limit. **OpenDML (AVI 2.0)** extensions in 1996 addressed the size limit and allowed improvements like better support for non-PCM audio[[93]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=AVI%20was%20first%20specified%20in,described%20in%20the%20Notes%2C%20which). Capabilities: AVI can technically wrap any video codec for which a FourCC code is defined and any audio codec (identified by a format tag). In practice, it was commonly used with **DivX/Xvid (MPEG-4 ASP)**, various DV codecs, Motion JPEG, Indeo, Cinepak, uncompressed, etc. It handles **synchronous audio-video playback** but doesn’t inherently ensure constant sync if variable bitrate audio is used (that historically caused issues, although hacks exist to compensate). Common applications: AVI was the **standard PC video container in the 1990s and early 2000s**. For example, **digital camera videos**, early NLE exports, and a vast number of internet-distributed videos (like fan videos, early “scene” releases of TV episodes, etc.) were in AVI format. Even today, some scientific or standalone systems use AVI for output because it’s straightforward. Preservation considerations: AVI is **fully documented and not encumbered by patents** (it’s essentially just a file structure)[[94]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Disclosure)[[95]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Adoption). This makes it transparent – any competent developer can write a parser for AVI. Tools like FFmpeg handle AVI very well. One issue is that AVI is an older format with limitations: it has trouble with **modern features like very long duration (size), variable frame rate, or subtitles**, and certain newer codecs don’t fit well (e.g., H.264 in AVI is possible but not ideal due to how AVI handles B-frame reordering – leading to so-called “packed bitstream” hacks). Also, metadata in AVI is minimal – it has an “INFO” chunk where a few tags (Title, Artist, etc.) can be stored, but beyond that, not much standardized descriptive metadata is supported[[96]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Self). In preservation, one would generally not choose AVI as a container for new encodings (better to use MKV or MOV or MP4 which handle modern needs), but one will encounter many legacy files in AVI. Those should be preserved or migrated carefully. Migration is usually straightforward (transmuxing AVI to MKV without re-encoding, for instance, is possible for many codecs and can help bypass AVI’s size limits or allow adding metadata). Nevertheless, an AVI kept in its original form will be readable for the foreseeable future, given the simplicity and wide adoption of the format[[95]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Adoption). In summary, AVI is a **legacy container** that is simple, widely supported, but limited. It’s an integral part of digital video history (the “.avi file” is virtually synonymous with video on a PC for a whole generation), so archives will have plenty of them. Ensuring those files remain playable may involve re-wrapping or transcoding if they used any quirky features, but overall AVI’s longevity outlook is solid[[97]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Widely%20adopted%20for%20video%20production,permitted%20codecs%20and%20other%20features)[[96]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Self).

### MPEG-2 Program Stream (PS)

**MPEG-2 Program Stream** is a container format defined in the MPEG-2 standard for **multiplexing one or more video and audio streams into a single stream** for reliable playback from storage media[[98]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=According%20to%20ISO%2FIEC%2013818,packets%2C%20expecting%20the%20channel%20coder)[[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous). It’s designed for relatively error-free environments (like discs or hard drives, as opposed to broadcast). Structural characteristics: Program Streams consist of variable-length **PES (Packetized Elementary Stream) packets** that contain video or audio frames, combined sequentially. They include metadata like time stamps (PTS/DTS) to synchronize audio and video[[100]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=transport%20stream%20and%20program%20stream,program%20stream%20is%20generally%20more). Unlike Transport Streams, Program Streams typically carry a single program (e.g., one video with its audio tracks) and do not include extensive forward error correction or multiple programs. Capabilities: Program Streams can embed limited metadata – for example, on DVD-Video (which uses a slightly restricted Program Stream known as VOB), navigation packs contain timing and chapter info. Subpictures (DVD subtitles) and extra audio tracks can be included as additional elementary streams. Common applications: The most famous usage is **DVD-Video’s VOB files**, which are MPEG-2 Program Streams containing MPEG-2 video, one or more audio tracks (PCM, AC-3, or DTS), and subtitle substreams. Program Streams were also used in early **digital video files** – files with extension .mpg or .mpeg often are PS (commonly containing MPEG-1 or MPEG-2 video with MP2 audio). Some camcorders that recorded MPEG-2 (early DVD-based camcorders) produced program stream files as well. Preservation considerations: As part of the MPEG-2 family, Program Streams are **openly documented** and broadly supported[[32]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Adoption)[[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous). Any software that handles MPEG video can usually parse .mpg PS files. The format is fairly simple (a linear interleaving of audio/video), so it’s transparent in that respect – though one downside is that it’s not as indexed as modern containers, so seeking can be slower unless an index is built (players often build one on the fly). For archiving, if one has DVD-Video content, the VOB (program stream) is the source and should be preserved or extracted without loss. The program stream, however, is not very **extensible**: it’s effectively limited to MPEG coded bitstreams (e.g., you wouldn’t put H.264 in a program stream; that’s what Transport Stream is for via later standards). Additionally, Program Streams are not optimal for streaming or error-prone scenarios (corruption of a packet could throw off a large portion of the stream). But in an archive on stable storage, that’s not a big concern. Compatible codecs: MPEG-2 PS can carry **MPEG-2 video** (and MPEG-1 video, and even MPEG-1/2 Audio or AC-3 audio). It’s worth noting that **MPEG-1’s System Stream** is conceptually similar to a Program Stream (and often .mpg files with MPEG-1 video are called program streams too). Modern usage: Program Streams have been largely supplanted by Transport Streams and MP4/MKV for most uses. However, **they remain important for legacy content** (DVDs, legacy broadcasts stored on DVD, etc.). The Library of Congress and other institutions actually list MPEG-2 Program Streams as an acceptable format for video[**[101]**](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=LC%20experience%20or%20existing%20holdings,See%20also%20%2018), likely because so much important content (like DVD-based collections) is encoded that way and it is a standardized, well-understood format. In summary, MPEG-2 Program Streams are a **simple multiplexed container** best suited for contained file playback (not streaming), exemplified by DVD media; they are easy to preserve (decoders will always be around due to DVDs), but not used for new content outside those contexts.

### MPEG-2 Transport Stream (TS)

**MPEG-2 Transport Stream (TS)** is a container format optimized for **transmission and broadcast** of video/audio, defined in MPEG-2 Part 1[[98]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=According%20to%20ISO%2FIEC%2013818,packets%2C%20expecting%20the%20channel%20coder)[[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous). Unlike Program Streams, Transport Streams break data into fixed-size packets (188 bytes each) and include robust features for synchronization and error resilience. Structural characteristics: TS packets are 188 bytes, each beginning with a sync byte (0x47)[[102]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=Digital%20Video%20and%20HD%20Algorithms,in%20the%20decoders%20do%20not)[[103]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=packetized%20into%20packets%20of%20188,and%20visual%20data%20and%20time). Each packet has a Packet Identifier (PID) that tells what stream it belongs to (video, audio, metadata, etc.), enabling multiplexing of multiple streams. TS can carry multiple programs (e.g., multiple TV channels in one stream) by assigning different PID sets. It also provides PSI (Program Specific Information) tables – like PAT (Program Association Table) and PMT (Program Map Table) – that describe what streams (PIDs) make up each program and timing information. Capabilities: Transport Streams are designed for **lossy or noisy environments** (over-the-air broadcast, etc.)[[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous). They have continuity counters to detect packet loss, and receivers can lock onto the stream at any point (important for live streaming). They also typically pad the bitrate to a constant rate by inserting null packets when necessary, especially in broadcast where a constant RF data rate is needed[[104]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=Transport%20streams%20may%20be%20either,Clock%20Unlike%20video%20encoding%2C%20when)[[105]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=CBR%20is%20referred%20to%20in,fluctuates%2C%20the%20overall%20transport%20stream). TS supports *real-time clocks*: PCR (Program Clock Reference) packets to sync decoder timing. Common applications: MPEG-2 TS is used for **digital television broadcasts** worldwide (ATSC in North America, DVB in Europe, etc.), for streaming media over networks (UDP/RTP streaming of TS, and HLS segments in an older mode), and on **Blu-ray Discs and AVCHD camcorders** (where .m2ts files are slightly extended transport streams)[[106]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=video%20and%20audio%20compressed%20streams,a%20player%20connects%20to%20a)[[107]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=both%20video%20and%20audio%20data,2%20Program%20Stream). For instance, an HDV videotape records MPEG-2 TS to miniDV cassettes, and Blu-ray multiplexes audio/video in 192-byte packets (188 + 4 extra bytes) known as M2TS. Preservation considerations: TS is a bit more complex than program stream and is *not* as straightforward as a file format, but it’s very standard in broadcast archives. Its strength is carrying synchronized multi-program content through systems where data might drop. In an archive context, you typically encounter Transport Streams in two scenarios: **off-air broadcast captures** (e.g., a recorded .ts file of a TV multiplex) or **camera recordings** (many cameras output .MTS files). The format’s documentation is open (ISO/IEC 13818-1 and ITU-T H.222) and very widely supported[[98]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=According%20to%20ISO%2FIEC%2013818,packets%2C%20expecting%20the%20channel%20coder)[[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous). Tools like FFmpeg, tsrecover, etc., can parse TS to retrieve program streams or elementary streams. One challenge is that raw transport streams can be large and contain multiple channels – archives might extract just the streams they need. Also, without an index, seeking inside a TS file can be slow (though format doesn’t really do indexing – it’s continuous). For long-term storage of a single video program, TS is somewhat storage-inefficient (the fixed packet overhead and alignment to meet constant bitrate can waste space, and the structure is more complex than needed for error-free storage)[[108]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=likely.,%28p.%20167). Thus, archives may choose to rewrap content from TS into a simpler container like MP4 or MKV for preservation, unless maintaining original format is important (like preserving an exact broadcast capture). Nonetheless, the Library of Congress lists **MPEG-2 TS as a preferred format for video** in their recommendations, likely because it ensures broad compatibility and often is the native format of born-digital broadcast content[[101]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=LC%20experience%20or%20existing%20holdings,See%20also%20%2018). In summary, MPEG-2 Transport Stream is the **standard container for broadcast video**, ensuring that video/audio can be transmitted or recorded in real-time reliably. Its continued use in broadcasting and media delivery (even new standards like ATSC 3.0 still optionally use TS within IP) means it will be decodable for a long time. Archives dealing with TV and radio collections will have to be comfortable with TS, extracting or trimming them as needed while keeping the essential structure for playback.

### MP4 (MPEG-4 Part 14)

**MP4** is a **container format standardized as ISO/IEC 14496-14**, derived from the QuickTime format. Often referred to as **MPEG-4 file format**, it’s one of the most common multimedia containers for video, audio, and subtitles. Structural characteristics: MP4 is built on the **ISO Base Media File Format (ISOBMFF)**, which uses a box/atom structure similar to QuickTime’s atoms[[86]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=As%20of%20February%202012%2C%20two,the%20file%20wrapper%20per%20se). An MP4 file usually contains a ftyp box (identifying the file type/compatibility), a moov box (with metadata: track definitions, timing, codec info), and one or more mdat boxes (media data). It can also include boxes for extending functionality (e.g., udta for user data, XMP metadata boxes, etc.). MP4 supports **streamable layout** (the moov can be placed at the beginning for progressive download, or fragmented MP4 features allow streaming). Capabilities: MP4 is very flexible but within a controlled scope – it’s designed to carry MPEG codecs and some others. Common tracks in MP4: **Video** (H.264, H.265, MPEG-4 Visual, AV1, etc.), **Audio** (AAC, MP3, AC-3, etc.), **Subtitles** (Timed Text, or images for subtitles), **Chapter and Metadata** tracks (e.g., Nero chapters or Apple’s metadata). It also supports **MPEG-4 Systems** like object descriptors, though those are rarely used now. An important aspect is MP4’s support for **metadata**: it can embed basic info like creation time, modification time, track names, and more. It’s not as extensive as Matroska’s tagging system, but it covers common needs and can embed cover art or rights information (and can use private metadata or XMP if needed). Common applications: MP4 became the dominant format for **online video distribution and storage**. Virtually all phones, cameras, and editing software can output MP4 files. It’s the standard for platforms like YouTube (which deliver video in MP4 format for certain browsers/devices), for streaming (HTTP Live Streaming segments can be fragmented MP4), and for **consumer video** (recordings, screen captures, etc. often end up as MP4). Its success comes from standardization and broad hardware support (all modern devices have some MP4 playback capability). Preservation considerations: MP4 is an **open standard (ISO)** and is essentially an offspring of QuickTime, so it’s well-documented and understood[[86]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=As%20of%20February%202012%2C%20two,the%20file%20wrapper%20per%20se). This makes it a safe choice compatibility-wise. For long-term preservation, a potential limitation is that MP4’s design expects certain codec types; while it can technically store non-MPEG codecs (using private codes), it’s usually used with standard codecs. Archives aiming to store, say, FFV1 video or other niche codecs might prefer Matroska or MOV, as MP4 doesn’t officially support FFV1, etc. Additionally, MP4 doesn’t natively support multiple subtitle formats as flexibly (it has Timed Text and can use image-based subs in MP4, but not SRT without conversion). Another consideration: MP4 is highly geared toward final delivery. It doesn’t have built-in redundancy or recovery data – it’s meant for reliable storage or streaming where the protocol ensures data integrity. If corrupted, an MP4 might be harder to recover than a simpler format or one with redundancy (like MPEG-2 TS, which has sync bytes often). However, for files on stable storage, this is minor. Many institutions use MP4 for **access copies** or proxy files (e.g., making H.264/AAC MP4 files for user access, while keeping an MOV/MKV master). As a master format, MP4 could also wrap high-quality video (it can hold ProRes if one wanted, or uncompressed, although not common; it’s more typical to stick to MP4’s intended codecs). One advantage is that MP4 supports **fragmentation and partial streaming**, meaning one can have very large recordings by segmenting them or stream them easily. In summary, MP4 is a **widely adopted, standard container** that strikes a good balance of flexibility and interoperability. It’s well-suited for distribution and is acceptable for preservation when containing high-quality codecs (the Library of Congress, for example, lists .mp4 with specific codecs as an acceptable video format[[109]](file://file-4szQrGLGTxcpr8hoTuhgTS#:~:text=3.%20MPEG)). Its broad ecosystem support ensures that an MP4 file created today (with common codecs) will likely be playable for decades to come.

### Matroska (MKV)

**Matroska** is an open, binary **container format** known for its flexibility and extensive feature set. An MKV file can contain video, audio, subtitles, chapters, attachments (like fonts or images), and metadata in one package. Structural characteristics: Matroska is built on **EBML (Extensible Binary Meta Language)**, which is like a binary XML – it defines a self-describing file structure with IDs and lengths[[110]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Matroska%20is%20a%20open%2C%20non,expected%20in%20files%20with%20version)[[111]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=As%20a%20purpose%20built%20multimedia,complex%27%2C%20logo%2C%20buttons%2C%20and%20control). The format defines a top-level Segment containing sub-elements: **Info** (general metadata like title, timestamp, segment duration), **Tracks** (defining each stream’s codec and properties), **Chapters**, **Tags** (for metadata tagging), **Cluster** (which holds the actual media frames in time-sliced “clusters”), **Cues** (an index for quick seeking), and **Attachments**[[112]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=EBML%20%27Root%20Element%27%20in%20the,for%20video%20tracks%20and%20SamplingFrequency)[[113]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=are%20available%20in%20the%20Codec,in%20that%20it%20supports%20that). This rich structure means Matroska can **precisely encapsulate complex relationships** – e.g., multiple audio tracks labeled with languages, or editions of a film via ordered chapters. Capabilities: Matroska supports virtually any codec – its design allows **Codec ID** strings to identify formats, and it’s extensible (new codecs can be added to the registry)[[114]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=audio%20and%20video%20payloads%20are,Subtitles%20and%20captions). It handles **advanced subtitle formats** (text subtitles like SRT/ASS or image-based subtitles), chapters (with editioning and nested chapters), and can even carry menus or interactive content (though those are rarely used). Attachments feature allows including fonts (for styled subtitles) or cover art or even executables. Importantly, Matroska is capable of **error resilience** in local playback: each Cluster is independently readable, so even if part of a file is damaged, later clusters can often still be accessed. Also, being open, it’s designed to be **evolvable** – the current version supports things like 3D video and is getting improvements (like better timecode support for multiple frame rates)[[115]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Improved%20support%20for%20timecode%2C%20as,planar%20and%203D%20videos). Common applications: Matroska is widely used in **digital video distribution in enthusiast communities** – e.g., high-quality encodes of videos often use MKV with H.264/H.265 video and various audio formats (because MKV imposes almost no restrictions). It’s also used by **WebM**, the web media format by Google (WebM is essentially a subset of Matroska for VP8/VP9/AV1 video and Vorbis/Opus audio)[[116]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=the%20Matroska%20file%20format,modified%20version%20WebM%2C%20WebM). Increasingly, Matroska is making inroads in professional and archival circles as well – for instance, many archives prefer **FFV1 in MKV** as a preservation master format[[117]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=long,constraint%20about%20captions%20and%20timecodes)[[118]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=factors%20because%20the%20Library%20of,now%20FFV1%2FMKV%20meets%20both%20requirements). The format is now officially being standardized under IETF (the Matroska specification is a work-in-progress Internet Draft, as of 2024)[[119]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Matroska%20is%20a%20open%2C%20non,Matroska%201%20to). Preservation pros: Matroska is **non-proprietary and well-supported in open-source**. It has no licensing issues. It can store *everything* relevant to an audiovisual item in one file (video, multiple audio tracks, subtitles, chapters, metadata, attachments), which is very attractive for preservation – no sidecar files needed for, say, subtitles or cover art. It’s also highly customizable; archivists can use Matroska tags to store metadata in a structured way. In practice, institutions like the University of Cologne and others (through the CELLAR working group of IETF and communities like PREFORMA) have championed Matroska for preservation. The Library of Congress in late 2023 added FFV1/MKV as a **Preferred Format** for video preservation[[120]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Statement%20%28RFS%29%20www,have%20allowed%20the%20upgrade%20from)[[121]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=confidence,now%20FFV1%2FMKV%20meets%20both%20requirements). A small drawback: not all consumer software supports MKV equally (particularly historically, some players on Windows or devices refused MKV in favor of MP4). But this is less of an issue now with VLC, FFmpeg, and even many TVs supporting MKV. Another consideration: Matroska’s flexibility means **due diligence is needed** – e.g., specifying what codecs and metadata conventions are allowed – so that it remains interoperable. In preservation workflows, using known combinations (like MKV with PCM audio and FFV1 video, or MKV with MPEG-2 video from DVDs) is recommended, rather than something highly exotic. Summing up, Matroska stands out as a **powerful, future-friendly container** ideal for archival use. It is sometimes called the “digital container that can hold anything,” comparable to how TAR works for files. With active development and community support, Matroska is likely to remain accessible long-term and avoids the pitfalls of proprietary formats[[119]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Matroska%20is%20a%20open%2C%20non,Matroska%201%20to)[[122]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=As%20a%20purpose%20built%20multimedia,complex%27%2C%20logo%2C%20buttons%2C%20and%20control).

### Material Exchange Format (MXF)

**MXF (Material Exchange Format)** is a professional audiovisual container/wrapper format standardized by SMPTE, designed for the **exchange of video, audio, and metadata** in production and archiving environments[[123]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Object,aware). MXF is often described as a “digital equivalent of a videotape” because it tries to package essence (media data) and metadata in a robust, though simpler-than-project-files, way for interoperability[[124]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=metadata). Structural characteristics: MXF files are object-oriented and use a binary key-length-value (KLV) coding. They contain **Header Metadata** (describing the file’s content and structure) followed by **essence containers** which hold the actual media bitstreams (video frames, audio samples)[[125]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Essences%20are%20wrapped%20in%20containers%2C,types%20of%20containers%3F%20%206)[[126]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Subtype%20of%20AAF_1_1%2C%20Advanced%20Authoring,1). MXF is codec-agnostic but typically wraps **broadcast-oriented codecs** (MPEG-2, DV, AVC-Intra, JPEG2000, etc.) in standardized wrappers called **Generic Containers**[[125]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Essences%20are%20wrapped%20in%20containers%2C,types%20of%20containers%3F%20%206). It supports multiple tracks (video, audio, data) and **synchronous timestamping** for each frame. Importantly, MXF files can be **indexed** internally, allowing for frame-accurate seeking and random access which is crucial in editing or playback of long files. Capabilities: MXF is *metadata-rich*. It can embed **descriptive metadata** (like title, program ID, etc.) and technical metadata (like aspect ratio, color spec) in its header. It often uses standards like **Dublin Core or TV Anytime** or custom schemas for metadata, and can also carry **timecode tracks** and segmentation data (as used in broadcast content delivery)[[123]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Object,aware)[[127]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=essences%20and%20what%20amounts%20to,aware). MXF supports **Operational Patterns (OPs)**, which are profiles defining how essences are stored – for example, **OP-1a (Single Item, Interleaved)** is one of the most common (all essence interleaved into one file, one complete sequence)[[128]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=The%20standard%20specifies%20a%20number,selected%20Application%20Specifications%20during%202012), whereas OP-Atom stores each essence (video or each audio channel) in separate files (often used in camera recording). There are also Application Specifications (AS) built on MXF to target specific use cases (e.g., AS-11 for UK broadcast delivery, AS-07 for archiving)[[126]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Subtype%20of%20AAF_1_1%2C%20Advanced%20Authoring,1). Common applications: MXF is widely used in **broadcast and video production**. Many tapeless cameras (Sony XDCAM, Panasonic P2) record MXF files. Broadcasters exchange MXF files as standardized delivery formats (often containing MPEG-2 or AVC-Intra video). **Digital cinema** packaging also uses MXF – a DCP will contain JPEG2000 MXF files for picture and sound. Archives use MXF for projects like videotape preservation – for instance, wrapping lossless JPEG 2000 or uncompressed v210 video in MXF OP-1a as an archival master[[18]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=LC%20experience%20or%20existing%20holdings,version%20or%20profile%20or%20MXF). Preservation considerations: MXF is an **open standard (SMPTE)**, and documentation is freely available (though it’s complex)[[129]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Disclosure%20Open%20standard,Adoption). It’s built specifically with long-term stability in mind: binary KLV is a very standardized method from the broadcast world (originating in digital videotape formats). The structure is somewhat heavy, but it is self-contained – the file’s metadata can describe all components and even an edit decision list for how to playback segments in order[[127]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=essences%20and%20what%20amounts%20to,aware). One advantage is that MXF’s rich metadata and **partitioning** can help ensure no loss of context – e.g., original timecodes from tape can be preserved, and multiple audio channels each labeled. MXF is widely adopted by archives focusing on broadcast content (e.g., national libraries archiving television often store in MXF). A notable example: the Library of Congress uses MXF with JPEG 2000 (per AS-07 spec) for its videotape reformatting masters[[18]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=LC%20experience%20or%20existing%20holdings,version%20or%20profile%20or%20MXF), indicating MXF’s acceptance as a preservation wrapper. Potential downsides: MXF’s flexibility means not all MXF files are equal – one must follow specific profiles for interoperability. Tools working with MXF must handle KLV parsing and various mappings, which can be complicated (but many broadcast-oriented software and FFmpeg do support common types). The complexity also extends to **multiple “flavors”** – an archive might need to commit to a particular Application Specification to avoid confusion. Nonetheless, because MXF is used in active broadcast, there is strong support and likely longevity. The existence of published **“Archive and Preservation” profiles (SMPTE RDD 48)** shows the format is intended to serve archives[[130]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Has%20subtype%20AMWA%20Application%20Specification,Specifications%2C%20not%20described%20here%20at). In short, MXF is a **robust, metadata-rich container** well-suited to complex media (multiple tracks, timecodes, etc.) and professional workflows. It’s effectively the standard for interchange in broadcast and a growing choice in preservation of video, ensuring that essence and metadata stay together in one file much like they did on videotape[[131]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Although%20the%20specification%20allows%20for,for%20continued%20production%20or%20archiving).

### ASF (Advanced Systems Format / WMV)

**ASF (Advanced Systems Format)** is a proprietary container format created by Microsoft for its Windows Media framework[[132]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Full%20name%20Advanced%20Systems%20Format,Relationship%20to%20other%20formats). It’s the wrapper used for files with **.wmv and .wma extensions** (Windows Media Video/Audio), and it can also carry other data streams (like script commands, images). Structural characteristics: ASF is an object-based file format using **GUIDs to identify each structure**[[133]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes). The file starts with a header containing a File Properties object (overall length, bitrate, etc.), Stream Properties objects (describing each stream’s codec and type), and possibly metadata objects for things like title/author (the Content Description and Extended Content Description objects)[[134][135]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes). The data is stored in Data Packets that interleave the streams for playback. ASF was designed with streaming in mind, meaning it can be delivered over networks in real time. Capabilities: ASF can encapsulate **audio, video, and even arbitrary binary streams** (with appropriate GUIDs)[[132]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Full%20name%20Advanced%20Systems%20Format,Relationship%20to%20other%20formats). Typically, it’s used with Windows Media codecs: **WMV1/2/3 (Windows Media Video versions, with WMV3 = VC-1) for video** and **WMA for audio**, but it also supports others (there were instances of ASF carrying MPEG-4 video or MP3 audio, though not common). It has provisions for **script command streams** (for example, events that trigger URLs or captions), for digital rights management info (DRM object in header), and for embedding **metadata** tags (Author, Title, Copyright, etc., plus extended tags can carry more)[[134][135]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes). ASF supports both **streaming and download**: a file can be played while downloading because the header up front provides indexes and the packets can be parsed on the fly. It also allows indexed seeking if an index object is present. Common applications: In the early 2000s, ASF was widely used in the form of **.wmv files for video** and **.wma files for audio**. For example, a lot of official movie trailers, corporate videos, or internet streaming clips were distributed as .wmv, and music as .wma (in competition with MP3/RealAudio). It was also the format behind **streaming via MMS protocol** and used in applications like Microsoft Silverlight. Some camcorders (especially some early HD pocket cams) recorded in WMV/ASF too. Preservation considerations: ASF is fully documented by Microsoft (specs released in 2004)[[136]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Disclosure%20Fully%20documented,per%20use%20or%20per%20unit), but it is proprietary in the sense that it was tied to the Windows Media ecosystem. However, being documented and reverse-engineered, tools like FFmpeg can demux ASF, and players like VLC can play it. In terms of long-term viability, **the ASF container itself is less popular now**, but support will linger due to the huge number of .wmv/.wma files created. From a metadata standpoint, ASF actually has decent support for self-documentation: the header can carry not only technical metadata but also bibliographic fields (title, author, etc.)[[134][135]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes). It also can include **integrity data**: e.g., ASF files can have a simple index for error correction if streaming, but there’s no inherent error correction like in transport streams. One advantage for streaming it had: it could be chopped into pieces for server distribution. For preservation, one challenge might be the **reliance on specific codecs**: many .wmv files use VC-1 or older WMV codecs, which are not as common today. But FFmpeg covers decoding of those (and Microsoft provides codec compatibility in Windows). Another challenge is if files were DRM-protected (some .wma or .wmv have DRM that requires contacting a license server). If an archive encounters such files, it needs to attempt to break or legally bypass the DRM now, as future license servers may be gone. As a container, ASF can be repackaged: it is possible to transmux the streams to another container (e.g., VC-1 video and WMA audio from ASF into MKV), which archives sometimes do to integrate those into modern systems. But it might be easier to just keep the ASF since players will likely support it for a long time (due to simplicity and documentation). In summary, ASF is the **container behind the Windows Media era**: not as flexible as Matroska or MP4, but it supported streaming and metadata and became quite widespread[[137]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Adoption%20WMA%20%20appears%20to,17%2C%20and%20WMV9_PRO). It’s a closed ecosystem format that has been opened up by documentation; while new content isn’t really produced in ASF, archives will have to handle existing ASF files (especially from government or corporate video in the 2000s). It’s generally advisable to migrate content out of ASF unless it’s too time-consuming, but simply playing and transcoding ASF is feasible with today’s tools.

### Flash Video (FLV)

**Flash Video (FLV)** is a container format used to deliver video over Adobe Flash Player. During the 2000s, it was one of the most prevalent formats for streaming and embedded web videos (e.g., YouTube’s early years used FLV). Structural characteristics: An FLV file starts with a short header (FLV signature and version) and then is composed of a sequence of **tags**[[138]](https://docs.fileformat.com/video/flv/#:~:text=FLV%20File%20Format%20FLV%20,audio%2C%20video%2C%20scripts%2C%20encryption). Each tag is of a certain type: **audio tag**, **video tag**, or **script data tag**. The tags carry encoded audio frames, video frames, or ActionScript Metadata (usually containing an *onMetaData* event with key metadata like duration, width/height, framerate, etc.)[[139]](https://stackoverflow.com/questions/42368566/if-metadata-in-flv-container-must-be-recorded-at-the-beginning-of-the-file#:~:text=,and%20on%20the%20same). The file is essentially a continuous stream of these tags, each preceded by its size and timestamp. The format is fairly simple: it doesn’t provide an index or complex structure, since it was primarily meant for streaming where the server pushes data and the client plays sequentially. Capabilities: FLV originally supported **Sorenson Spark (H.263 baseline)** video and MP3 audio (that was Flash Player 7 and earlier). Later, Flash Player 8 added support for **On2 VP6 video** (which gave better quality at low bitrates) and ADPCM or Nellymoser audio codecs for higher fidelity. By Flash Player 9 Update 3, **H.264 (AVC)** video and AAC audio support was introduced – though that typically was in the context of the **F4V** format (an MP4-like container). Standard FLV (version 1) could technically carry H.264/AAC, but Adobe encouraged F4V for that. Still, many H.264 streams were served as .flv by just embedding H.264/AAC in FLV tags. FLV supports **alpha channel** with VP6 codec (there was a VP6 with alpha used for transparent videos). It also allows some basic interactive data: e.g., synchronized event triggers via script tags (like cue points). **Metadata**: The onMetaData script tag in FLV typically includes info like duration, video codec, framerate, frame dimensions, audio codec, sample rate, and optionally custom metadata inserted by an encoder (some tools include creation time or encoder name). This is usually at the start of the file for progressive download, or can be provided in the first few packets on a stream[[140]](https://en.wikipedia.org/wiki/Flash_Video#:~:text=Flash%20Video%20,bytes%20and%20timecode%20of). Common applications: FLV was the default for **Flash Player video streaming** – sites from YouTube and Vimeo to news sites and Web advertisements all used FLV in the mid-2000s. Content creators would often output a .flv to put on websites because Flash Player had a consistent decoder. It was also used with **RTMP** (Real Time Messaging Protocol) for live streaming and quick seek on servers. Preservation issues: Flash Video as a format is now effectively obsolete – Adobe Flash reached end-of-life in 2020 and modern browsers do not support the plugin. However, the content in FLV files is still accessible via tools like FFmpeg, which can easily decode and even transcode FLV files. So the container itself isn’t difficult to handle if one has the file. One challenge is that many FLV files might be low quality (designed for low bandwidth) or use proprietary codecs (Sorenson Spark and VP6 are not used elsewhere). FFmpeg does support those codecs, but they’re not as optimized as mainstream ones. Also, if any FLV relied on a *streaming server context* (for example, the absence of metadata or split across multiple files), you might need to reconstruct it. But generally, standalone FLV files contain everything needed (except maybe an external player to interpret script events, which is rarely critical for content). Another note: **F4V** files (with .f4v extension) were introduced by Adobe for H.264/AAC content – but they are actually ISO BMFF (like MP4). The .flv files usually imply Spark, VP6, or similar older codecs. Archives will encounter .flv in web archive collections or user-generated content collections from the 2000s. The prudent approach is often to transcode them to a modern format for access (since native playback requires either Flash or a converter). But one should keep the original FLV as well because transcoding from a lossy source will lose quality – maybe a better converter in the future could eke out slightly better decoding or recover something from artifacts. The FLV container itself is lightweight and doesn’t add significant overhead or risk aside from the need to parse the tags. Summarily, FLV is a **legacy streaming container** that was once everywhere on the web. It’s relatively simple (and lower-overhead than something like MKV or MP4 for its task) but limited – e.g., cannot have multiple audio tracks or advanced subtitles, and it assumes progressive streaming. For its era, it served its purpose of getting video online. In an archival context, FLV files are important as they represent the first wave of user-shared online videos and news streams. Converting them to something like MKV/MP4 with more preservation-friendly codecs (or at least documenting how to play them via FFmpeg) will ensure they remain accessible as Flash technology fades.

### Ogg (Ogg Vorbis/Theora Container)

**Ogg** is a free and open container format created by the Xiph.Org Foundation, commonly used for **Vorbis audio, Theora video, and other codecs like Opus or FLAC**. It was designed for streaming and file storage of multimedia, emphasizing open standards. Structural characteristics: Ogg is fundamentally a stream of “pages” (typically ~4-8KB each) that contain segments of the various data streams[[141]](https://wiki.xiph.org/Theora#:~:text=Theora%20is%20a%20video%20codec%2C,future%20scope%20for%20encoder). It uses a framing scheme where each page has a sync pattern and headers that include granule position (a codec-specific time marker) and flags for page sequencing. This makes Ogg good for **continuous streaming** – pages can be parsed on the fly, and if some are lost, you can resynchronize at the next page boundary. Capabilities: Ogg can multiplex multiple streams (e.g., one video + one audio, or multiple audio tracks) – though in practice it’s most often one video with one audio, or just audio. It supports **basic metadata** via the “VorbisComment” system (which is actually used by various codecs in Ogg, not just Vorbis). These comments can include Title, Artist, etc., similar to ID3 tags. However, Ogg’s metadata system is not as elaborate as MKV’s; it’s more akin to an array of key–value pairs. Ogg has no inherent concept of chapters or subtitle tracks (subtitles have been done by muxing Kate streams or similar, but that’s not common). It does support **theora-specific or vorbis-specific headers** that describe video frame rates or audio sample rates. A key aspect of Ogg is that it’s a **bitstream format rather than a file format** strictly – an Ogg stream can be within a file (.ogv, .ogg) or broadcast live. Common applications: The most popular use is **Ogg Vorbis audio (.ogg)**, which became a common format for free music distribution (Vorbis being an alternative to MP3/AAC). For video, **Ogg Theora (.ogv)** was widely used by Wikipedia and other open-content projects around 2007-2013, because it was the only widely viable royalty-free video then. Many Linux applications and games also adopted Ogg for soundtracks or cutscenes. With Opus (an audio codec), Ogg is the default container (often with extension .opus or .ogg). Preservation considerations: Ogg is completely free of patents and well-documented. It was designed with long-term in mind (Xiph’s philosophy for open media). Decoders and libraries (like libogg, libvorbis, libtheora) are open-source and widely available, so support for Ogg is strong in open-source software (VLC, FFmpeg, etc., all handle Ogg). However, some mainstream/legacy systems (like Windows Media Player) historically did not support Ogg without adding codecs, which is a minor point if someone in future tries to click an .ogg file in old software. For archiving, Ogg’s streaming nature means it doesn’t have a built-in index; this isn’t an issue if playing linearly, but random access can be slower (the player may have to scan through pages to find a certain timestamp, though there are tools to build an index). In terms of usage in archives: Ogg might appear in collections of early web videos (especially because Theora was often suggested to archives wanting an open format back when H.264 was patent-encumbered). There are also potentially some radio broadcasts archived in Ogg (with Vorbis or Opus) by communities favoring open formats. By today’s standards, Theora is an outdated codec (worse compression than H.264) and Vorbis is being supplanted by Opus, but the content already encoded in them is still perfectly playable. One tricky scenario: Ogg can embed FLAC (lossless audio) or even Dirac (another open video codec) – these are less common, but possible in archives (e.g., some archives might have chosen Ogg/FLAC for audio preservation at one point, though now they’d more likely use WAV or FLAC in native container). If so, those should still be fine; just that not every player might know an .ogg file can contain FLAC. Overall, Ogg is a **simple, effective container** for one video+audio or audio-only. It’s not as feature-rich as other containers in terms of multiple tracks or chapters, but it’s very suited to continuous streaming and has low overhead. Because it’s open, it’s considered a good format for long-term accessibility: no legal barriers and fully published specs. If an archive has materials in Ogg, there’s no strong need to transcode or rewrap them (unless for standardization purposes), since the format itself is healthy. The main drawback might be that outside certain circles, .ogg isn’t as ubiquitous as .mp4 or .mkv, but support is easily attainable via open-source tools. In summary, Ogg has been an **important container for open media** and remains relevant especially for audio (Opus/Vorbis), while for video it’s largely of historical interest after being overtaken by MP4/WebM in usage. It exemplifies an archival-friendly approach (no patents, documentation available) and thus content stored in Ogg from the 2000s can likely be decoded far into the future with minimal hassle.

[[1]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=A%20digital%2C%20color,or%20Y%27UV) [[2]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Uncompressed%204%3A2%3A2%20video%20picture%20streams,provides%20codes%20for%2030%20packed) [[3]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml#:~:text=Relationship%20to%20other%20formats%20Subtype,Web%20site%20at%20this%20time) Uncompressed YCbCr Video Picture Stream (4:2:2)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000352.shtml>

[[4]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Lossless%20Compression%20Codecs%3AFFV1%3A%20A%20key,like%20MDPI%20for%20format%20selection) [[5]](file://file-2x1C3u3a8h8V9tBz9n5RFQ#:~:text=Considerations%20for%20Lossless%20Codecs%3A%20They,possibly%20greater%20computer%20processing%20capabilities) Doc\_Notes.docx

<file://file-2x1C3u3a8h8V9tBz9n5RFQ>

[[6]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Format%20Overview) [[7]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Open%20Source%20and%20Affordable) [[117]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=long,constraint%20about%20captions%20and%20timecodes) [[118]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=factors%20because%20the%20Library%20of,now%20FFV1%2FMKV%20meets%20both%20requirements) [[120]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=Statement%20%28RFS%29%20www,have%20allowed%20the%20upgrade%20from) [[121]](https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/#:~:text=confidence,now%20FFV1%2FMKV%20meets%20both%20requirements) Embracing FFV1 in Matroska Container as a “Preferred Format” in the RFS | The Signal

<https://blogs.loc.gov/thesignal/2023/12/embracing-ffv1-matroska-container-preferred/>

[[8]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%27s%20algorithm%20is%20roughly%20the,The%20predictor%20functions%20are) [[9]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=,and%20decoded%20on%20its%20own) [[10]](https://wiki.multimedia.cx/index.php/HuffYUV#:~:text=HuffYUV%20is%20a%20lossless%20video,so%20its%20main%20features%20are) HuffYUV - MultimediaWiki

<https://wiki.multimedia.cx/index.php/HuffYUV>

[[11]](https://en.wikipedia.org/wiki/Lagarith#:~:text=Lagarith%20is%20an%20open%20source,a%20few%20aims%20in%20mind) [[12]](https://en.wikipedia.org/wiki/Lagarith#:~:text=While%20not%20as%20fast%20as,63%2C%20and%20RGBA%20colorspaces) [[13]](https://en.wikipedia.org/wiki/Lagarith#:~:text=be%20slower,each%20frame%20can%20be%20separately) [[14]](https://en.wikipedia.org/wiki/Lagarith#:~:text=colorspaces.%20Keyframes%20Disallowing%20inter,joining%20and%20seeking%20much%20easier) Lagarith - Wikipedia

<https://en.wikipedia.org/wiki/Lagarith>

[[15]](https://en.wikipedia.org/wiki/Motion_JPEG#:~:text=Prior%20to%20the%20rise%20in,demand%20%20143) [[16]](https://en.wikipedia.org/wiki/Motion_JPEG#:~:text=M,121%20and%20Microsoft%20Edge) Motion JPEG - Wikipedia

<https://en.wikipedia.org/wiki/Motion_JPEG>

[[17]](https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf#:~:text=Archiving%20lhncbc,a%20truly%20lossless%20compression) [PDF] An Evaluation of Motion JPEG 2000 for Video Archiving

<https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9302.pdf>

[[18]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=LC%20experience%20or%20existing%20holdings,version%20or%20profile%20or%20MXF) [[123]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Object,aware) [[124]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=metadata) [[125]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Essences%20are%20wrapped%20in%20containers%2C,types%20of%20containers%3F%20%206) [[126]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Subtype%20of%20AAF_1_1%2C%20Advanced%20Authoring,1) [[127]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=essences%20and%20what%20amounts%20to,aware) [[128]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=The%20standard%20specifies%20a%20number,selected%20Application%20Specifications%20during%202012) [[129]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Disclosure%20Open%20standard,Adoption) [[130]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Has%20subtype%20AMWA%20Application%20Specification,Specifications%2C%20not%20described%20here%20at) [[131]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml#:~:text=Although%20the%20specification%20allows%20for,for%20continued%20production%20or%20archiving) Material Exchange Format (MXF)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000013.shtml>

[[19]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=independent%20files,place%20in%20professional%20television%20production) [[20]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=Clarity%20) [[21]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=,data%20rate%20of%20100%20MB%2Fs) [[22]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=Open%20standards,SMPTE) [[23]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=incorporated%20in%20files%20is%20also,Licensing%20and%20patents%20None%20identified) [[24]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml#:~:text=DV%20signals%20are%20formatted%20for,as%20AVI%2C%20QuickTime%2C%20and%20MXF) Digital Video Encoding (DV, DVCAM, DVCPRO)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000183.shtml>

[[25]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Moving%20Image%20Normal%20rendering%20Good,off%20to%20make%20sure%20things) [[26]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Description%20Compression%20encoding%20for%20video,by%20%2010%2C%20QuickTime%20MPEG) [[27]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Clarity%20%28high%20image%20resolution%29%20Moderate,compression%2C%20and%20the%20encoder%20used) [[28]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Telecommunication%20Standardization%20Sector%29%20recommendation%20H,2%20Layer%20II%20Audio%20Encoding) [[29]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Adoption) [[30]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml#:~:text=Tag%20Value%20Note%20Filename%20extension,2046%20Internet%20Media%20Type%20video%2Fmpv) MPEG-1 Video Coding (H.261)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000035.shtml>

[[31]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Description%20This%20description%20pertains%20to,Each%20PES%20packet%20includes%20a) [[32]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Adoption) [[33]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=MPEG,at%20this%20time%3A%20SNR%20Scalable) [[34]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Widely%20adopted%20for%20filemaking%2C%20DVD,exist%20for%20encoding%20and%20decoding) [[35]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Disclosure%20Open%20standard,Adoption) [[38]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml#:~:text=Spatially%20Scalable%2C%20and%20High,and%20Layer%20II%20Audio%20Encoding) MPEG-2 Video Encoding (H.262)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000028.shtml>

[[36]](file://file-AH8UoofHcWxUPBGCWjywFQ#:~:text=video%2C%20typically%2050%20Mbps%20for,supported%20by%20the%20vendor%20community) [[37]](file://file-AH8UoofHcWxUPBGCWjywFQ#:~:text=MPEG,2) FADGI\_VideoReFormatCompare\_pt5\_20141202.pdf

<file://file-AH8UoofHcWxUPBGCWjywFQ>

[[39]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Compression%20encoding%20for%20video%20identical,264%20video) [[40]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Beginning%20in%20about%202005%2C%20there,See%20also%20%2022) [[41]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Adoption) [[42]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=MPEG,at%20a%20lower%20data%20rate) [[43]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml#:~:text=Compression%20encoding%20for%20video%20identical,264%20video) MPEG-4, Visual Coding (Part 2) (H.263)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000080.shtml>

[[44]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=Adoption%20The%20underlying%20encoding%2C%20sometimes,in%202005%3B%20adoption%20not%20determined) [[45]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=and%20an%20encoding%20variously%20listed,4_AVC_BP) [[46]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml#:~:text=Relationship%20to%20other%20formats%20Used,ISO%2FIEC%20TS%2030135%3A2014) MPEG-4, Advanced Video Coding (Part 10) (H.264)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml>

[[47]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=One%20of%20the%20main%20advantages,for%20hardware%3B%20capable%20of%20consistent) [[52]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=The%20formation%20of%20AOMedia%20and,HEVC%20was) [[54]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=,than%20or%20equal%20to%202) [[55]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=Relationship%20to%20other%20formats%20Modification,Not%20documented%20on%20this%20resource) [[56]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig#:~:text=Nonetheless%2C%20the%20HEVC%20patent%20situation,See%20HEVC%20for%20more%20details) AV1 Video Encoding (AOMedia Video 1)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000541.shtml?loclr=blogsig>

[[48]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=and%20intra%20,HEVC%20in%20a%202017%20video) [[49]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=HEVC%20is%20a%20important%20advance,varied%20test%20set%20of%20video) [[50]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=of%20the%20reference%20software%20for,improvement) [[51]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=MPEG,a%20desired%20image%20or%20video) [[53]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml#:~:text=The%20underlying%20HEVC%20encoding%20is,Joins) High Efficiency Video Coding (HEVC) Family, H.265, MPEG-H Part 2

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000530.shtml>

[[57]](http://www.linux-magazine.com/Online/News/Google-funds-OGG-Theora#:~:text=The%20OGG%20Theora%20codec%20is,2%2C) Google funds OGG Theora - » Linux Magazine

<http://www.linux-magazine.com/Online/News/Google-funds-OGG-Theora>

[[58]](https://wiki.xiph.org/Theora#:~:text=Theora%20is%20a%20video%20codec%2C,future%20scope%20for%20encoder) [[141]](https://wiki.xiph.org/Theora#:~:text=Theora%20is%20a%20video%20codec%2C,future%20scope%20for%20encoder) Theora - XiphWiki

<https://wiki.xiph.org/Theora>

[[59]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=alternatives.%20For%20example%2C%20MPEG,and%20RealAudio%2C%20and%20for%20related) [[67]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Normal%20rendering%20Good%20support,and%20the%20encoding%20option%20selected) [[68]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Adoption%20Widely%20adopted%20for%20streaming,A%20new%20option%20is) [[69]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Disclosure%20Proprietary%20format%20with%20little,Although%20those%20developing%20HTML5) [[70]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Licensing%20and%20patents%20Licensing%20for,source%20code%20for%20the%20codecs) [[71]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml#:~:text=Full%20name%20RealVideo%2C%20Version%2010,from%20version%2010%20only%20in) RealVideo, Version 10

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000050.shtml>

[[60]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Wikipedia%20VC,broadcast%20and%20video%20industry%20professionals) [[61]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=According%20to%20the%20Wikipedia%20VC,free%20implementation%20of%20this%20codec) [[62]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=According%20to%20the%20Wikipedia%20VC,free%20implementation%20of%20this%20codec) [[63]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Disclosure%20Decoding%20specification%20is%20an,Adoption) [[64]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=H.261%2C%20H.263%2C%20MPEG,broadcast%20and%20video%20industry%20professionals) [[65]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Adoption) [[66]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml#:~:text=Licensing%20and%20patents%20According%20to,exclusive%20licensing%20body) Windows Media 9 Video Codec; SMPTE VC-1

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000095.shtml>

[[72]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=Apple%20ProRes%20is%20a%20family,Apple%20ProRes%204444%20Codec%20Family) [[73]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20was%20designed%20to%20be,the%20majority%20of%20FCP%20users) [[74]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=The%20key%20character%20traits%20that,422%20family%20are%20support%20for) [[75]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=While%20detailed%20technical%20information%20on,bitstream%20syntax%2C%20the%20bitstream%20element) [[76]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=While%20detailed%20technical%20information%20on,available%20for%20a%20fee) [[77]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=ProRes%20codecs%20are%20usually%20contained,See%20Notes%20for%20more%20information) [[78]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml#:~:text=but%20rather%20,which%20details%20the%20frame_size%20and) Apple ProRes 422 Codec Family

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000389.shtml>

[[79]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=Avid%20DNxHD%20%28,3%20standard.%5B%201) [[80]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=DNxHD%20is%20a%20video%20codec,in%20a%20%2059%20container) [[81]](https://en.wikipedia.org/wiki/Avid_DNxHD#:~:text=On%20February%2013%2C%202008%2C%20Avid,3%20standard.%5B%202) Avid DNxHD - Wikipedia

<https://en.wikipedia.org/wiki/Avid_DNxHD>

[[82]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=File%20format%20that%20wraps%20video%2C,versions%207%20and%208) [[83]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Specialists%20describe%20QuickTime%20atoms%20as,type%2C%20and%20a%20data%20payload) [[84]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=MJPEG%20%2C%20%2036%2C%20AAC,documentation) [[85]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=LC%20experience%20or%20existing%20holdings,Digital%20And%20Physical%20Media) [[86]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=As%20of%20February%202012%2C%20two,the%20file%20wrapper%20per%20se) [[87]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Disclosure%20Fully%20documented,See%20information%20on%20the) [[88]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Technical%20metadata%20is%20in%20the,Writer) [[89]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml#:~:text=Transparency%20The%20file%20format%20is,documentation) QuickTime File Format

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000052.shtml>

[[90]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=File%20format%20for%20moving%20image,the%20data%20chunks%20within%20the) [[91]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=File%20format%20for%20moving%20image,A%20third%20optional) [[92]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=subchunk%20is%20tagged%20movi%20and,data%20chunks%20within%20the%20file) [[93]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=AVI%20was%20first%20specified%20in,described%20in%20the%20Notes%2C%20which) [[94]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Disclosure) [[95]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Adoption) [[96]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Self) [[97]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml#:~:text=Widely%20adopted%20for%20video%20production,permitted%20codecs%20and%20other%20features) AVI (Audio Video Interleaved) File Format

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000059.shtml>

[[98]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=According%20to%20ISO%2FIEC%2013818,packets%2C%20expecting%20the%20channel%20coder) [[99]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=of%20the%20specification%20published%20as,where%20a%20stable%20and%20continuous) [[100]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=transport%20stream%20and%20program%20stream,program%20stream%20is%20generally%20more) [[101]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=LC%20experience%20or%20existing%20holdings,See%20also%20%2018) [[102]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=Digital%20Video%20and%20HD%20Algorithms,in%20the%20decoders%20do%20not) [[103]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=packetized%20into%20packets%20of%20188,and%20visual%20data%20and%20time) [[104]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=Transport%20streams%20may%20be%20either,Clock%20Unlike%20video%20encoding%2C%20when) [[105]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=CBR%20is%20referred%20to%20in,fluctuates%2C%20the%20overall%20transport%20stream) [[106]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=video%20and%20audio%20compressed%20streams,a%20player%20connects%20to%20a) [[107]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=both%20video%20and%20audio%20data,2%20Program%20Stream) [[108]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml#:~:text=likely.,%28p.%20167) MPEG-2 Transport Stream

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000635.shtml>

[[109]](file://file-4szQrGLGTxcpr8hoTuhgTS#:~:text=3.%20MPEG) Recommended Formats Statement – Moving Image Works \_ Resources (Preservation, Library of Congress).pdf

<file://file-4szQrGLGTxcpr8hoTuhgTS>

[[110]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Matroska%20is%20a%20open%2C%20non,expected%20in%20files%20with%20version) [[111]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=As%20a%20purpose%20built%20multimedia,complex%27%2C%20logo%2C%20buttons%2C%20and%20control) [[112]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=EBML%20%27Root%20Element%27%20in%20the,for%20video%20tracks%20and%20SamplingFrequency) [[113]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=are%20available%20in%20the%20Codec,in%20that%20it%20supports%20that) [[114]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=audio%20and%20video%20payloads%20are,Subtitles%20and%20captions) [[115]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Improved%20support%20for%20timecode%2C%20as,planar%20and%203D%20videos) [[116]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=the%20Matroska%20file%20format,modified%20version%20WebM%2C%20WebM) [[119]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=Matroska%20is%20a%20open%2C%20non,Matroska%201%20to) [[122]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml#:~:text=As%20a%20purpose%20built%20multimedia,complex%27%2C%20logo%2C%20buttons%2C%20and%20control) Matroska Multimedia Container

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000342.shtml>

[[132]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Full%20name%20Advanced%20Systems%20Format,Relationship%20to%20other%20formats) [[133]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes) [[134]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes) [[135]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Metadata%20is%20contained%20in%20Header,rights%20management%20and%20other%20purposes) [[136]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Disclosure%20Fully%20documented,per%20use%20or%20per%20unit) [[137]](https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml#:~:text=Adoption%20WMA%20%20appears%20to,17%2C%20and%20WMV9_PRO) ASF (Advanced Systems Format)

<https://www.loc.gov/preservation/digital/formats/fdd/fdd000067.shtml>

[[138]](https://docs.fileformat.com/video/flv/#:~:text=FLV%20File%20Format%20FLV%20,audio%2C%20video%2C%20scripts%2C%20encryption) FLV File Format

<https://docs.fileformat.com/video/flv/>

[[139]](https://stackoverflow.com/questions/42368566/if-metadata-in-flv-container-must-be-recorded-at-the-beginning-of-the-file#:~:text=,and%20on%20the%20same) If metadata in flv container must be recorded at the beginning of the ...

<https://stackoverflow.com/questions/42368566/if-metadata-in-flv-container-must-be-recorded-at-the-beginning-of-the-file>

[[140]](https://en.wikipedia.org/wiki/Flash_Video#:~:text=Flash%20Video%20,bytes%20and%20timecode%20of) Flash Video - Wikipedia

<https://en.wikipedia.org/wiki/Flash_Video>