Unifying Global Video Strategies

MP4 File Fragmentation For Broadcast, Mobile and Web Delivery

A *Transitions in Technology* White Paper

Timothy Siglin
Co-Founder
Transitions, Inc.
16 November 2011

Unifying Global Video Strategies

MP4 File Fragmentation For Broadcast, Mobile and Web Delivery

Consistent multi-platform audio and video content delivery presents an ongoing challenge for broadcasters. Explosive smartphone and tablet growth on varying operating systems —Android™, Apple® iOS, or Windows® Phone—threatens to create a user-experience divide between users on mobile devices, at the desktop or in the living room.

Broadcasters must address multi-platform consumption demands without compromising content security or network efficiencies. Many broadcasters are assessing efficiency of transport protocols used for content delivery, to see how they stack up for web and mobile delivery. Some legacy solutions, such as MPEG-2 Transport Stream (M2TS), lack basic web delivery functions.

What key information do broadcasters and network operators need to know as they look for more efficient approaches to the media delivery? This white paper explores fragmented MP4 files (fMP4) and considers whether the fMP4 format can replace legacy file formats.

Along the way, we'll explore four key areas that impact both broadcasters and network operators:

- Format benefits of fMP4
- Movement toward fMP4 standardization
- Network benefits of fMP4
- Platforms supporting fMP4

The Challenge

MPEG-2 Transport Streams (M2TS), in use for almost two decades, have been deployed for widely varied broadcast and physical media delivery solutions (e.g. ATSC and Blu-Ray). However, little has changed in that timeframe for basic M2TS transport stream capabilities. For instance M2TS still lacks an integrated solution for digital rights management (DRM). The M2TS derivative used by the proprietary Apple® HTTP Live Streaming (HLS) protocol also lacks timed-text or closed-captioning features such as CEA 708 for ATSC TV or SMPTE Timed Text for fMP4 Common File Format.

MPEG-4 versus MP4

MPEG-4 is often confused with MP4 (a particular MPEG-4 file format). The MPEG-4 standard's 28 parts detail audio, video and file structure specifications. Part 3 is for AAC audio (ISO/IEC 14496-3) and Part 10 covers AVC video (ITU-T H.264 or ISO/IEC 14496-10). Part 12 covers ISO Base Media File Format (ISOBMFF) using *boxes* (or *atoms* in the QuickTime® file format, the basis for Part 12). Part 14 defines MP4 file format with multiplexed audio/video samples. Fragmented MP4 (or fMP4) refers to fragmenting Part 12 ISOBMFF using non-multiplexed audio/video.

Appendix A shows how key boxes ('mdat', 'moof', 'moov') fit together.

Benefits of the Fragmented MP4 Approach

Is fragmenting MP4 an alternative solution that addresses the shortcomings of M2TS? It appears the answer is "yes" as fragmented MP4 files have six key benefits:

- Content/Metadata Separation
- Independent Track Storage
- Trick-Play Modes
- Backwards Compatibility
- Seamless stream splicing
- Integrated DRM

Separating Content and Metadata. With MP4 files, metadata can be stored separately from audio and video content (or "media data"). Conversely, M2TS multiplexes (combines) multiple elementary streams along with header/metadata information.

In other words, the MP4 approach allows flexible placement of metadata, independent of where media data is stored. This separation of content and metadata allows specification of a format using movie fragments optimized specifically for adaptive switching between different media data, whether that media data is varying video qualities, multiple camera angles, different language tracks or even different captioning or time-texted data.

Independent Track Storage. In fragmented MP4 (fMP4), elementary streams are not multiplexed and are independently stored separate from metadata, as noted above. As such, fMP4 file storage is location agnostic and can store audio- or video-only elementary streams in separate locations that are addressable from XML-based manifest files. One benefit of separating metadata and media data is the ease with which a stream combination can be aggregated by the playback device. An elementary audio file (m4a) stored at one location can be combined with an elementary stream video file (m4v) stored at another. The manifest file combines both together, by downloading and synchronizing their movie fragments, without requiring a multiplex of several elementary streams beforehand on the server.

The fragment-addressable nature of fMP4 stores all of an audio or video track's fragments in the same file, unlike the proprietary HLS approach which uses thousands of small M2TS multiplexed files. Audio and video segments in fMP4 can be addressed separately as single fragments of bytes, with each fragment containing the necessary metadata and media data to make it decryptable and decodable. A manifest file for fMP4 needs only a template to generate addresses for audio or video movie fragments, rather than the HLS approach of a manifest containing a list of every file name for the thousands of short file chunks needed to construct every multiplexed audio/video combination.

Trick-play modes. Trick-play modes—such as fast-forward or reverse, slow motion, and random access to chapter points—can be accomplished via intelligent use of metadata. Super fast forward and reverse are also supported by requesting only the start of each movie fragment containing the 'moof' box and the start of the 'mdat', which contains a random access picture (AVC IDR picture).

Transport stream chunks may not start with entry points, so skipping from one keyframe to another is difficult without parsing through the entire set of TS packets. In addition, fMP4 supports multiple

playback modes with scalable codecs like SVC that can encode different frame rates in the same media data so a player can request and play different frame rates from the fMP4 container.

The Unseen Danger of Combinatorial Complexity

The one-to-one distribution model of web video delivery works against M2TS delivery. Consider the following scenario:

A typical DVD movie in Europe or Asia requires several discs with different combinations of audio/language, subtitles, description tracks for the hearing or visually impaired, stereo, multichannel, various codecs, commentaries, camera angles and the like. All told, more than 40 different audio, video, and subtitle tracks may be distributed in a region.

Each video track needs to be encoded at several bitrates for adaptive streaming delivery at multiple resolutions for displays ranging from cell phones to HD TVs. The number of tracks can easily exceed those on a typical single-region DVD. Streaming is usually bandwidth limited, so only the tracks a user selects for playback should be delivered to a particular user, since unused audio and subtitle tracks directly reduce video quality in proportion to the bandwidth wasted. For multiplexed video formats, such as M2TS, that means a different file must be encoded for each combination of audio, video, and subtitles to be delivered.

For example, one combination could be a single video track with English audio and French subtitles, or another with French audio and English subtitles. Each combination needs to be encoded at several bitrates.

This plethora of combinations can quickly become unwieldy, with upwards of 4000 possible transport streams combinations—a multiple far above the 40 audio, video and subtitle tracks required for the DVD example. Netflix® estimates using an M2TS / HLS approach for their library could result in several billion assets, due to combinatorial complexity—a situation they avoid by using an fMP4 elementary stream approach, by selecting and combining tracks on the player.

Keeping track of these possible combinations, if encoded and multiplexed ahead of time, becomes a content management nightmare, especially since some transport stream solutions consist of thousands of small files for each transport stream combination. Even if a byte-range solution were available, however, the asset management of up to 4000 different transport streams becomes an unnecessary and cumbersome approach to single-stream delivery.

Tracks for broadcast purposes are typically limited to a primary audio, secondary audio for another language, commentary audio and subtitles in one or more alternate languages. This limited grouping of tracks, when applied to streaming delivery, still requires a large number of combinations. Take three different resolutions for a variety of screen sizes, add eighteen video tracks at different bitrates for desktop, mobile and tablet resolutions, then top it off with a number of audio and subtitle tracks for language and accessibility purposes. Based on this simple example, it is easy to see how even basic DVD video and audio delivery on the web can quickly generate many multiplexed files due to combinatorial complexity, and why the fMP4 approach greatly simplifies delivery

Backwards compatibility to MPEG-2 Transport Streams. Separating timing information from elementary video and audio streams allows conversion from several fragmented MP4 elementary streams into a multiplexed M2TS. For Adobe® and Microsoft®, time-alignment properties inherent to fMP4 means their respective server solutions can easily convert to the proprietary Apple HLS protocol for iPhone® and iPad® devices, which uses MPEG-2 Transport Stream segments.

“There are many media formats in the world,” says Kilroy Hughes, a Microsoft senior digital media architect, “so media interoperability between services and devices is a huge problem for Internet

video. We feel the best solution is to migrate to a Common File Format for Internet video delivery, analogous to standardizing on DVD in packaged media. When an HLS device requests a stream, we create an M2TS on the fly from the requested combination of independent tracks stored in Common File Format, which uses the fragmented ISO Media file format. MP4 elementary streams are time-aligned, so it's easy to create MPEG-2 multiplex chunks corresponding to each movie fragment from the MPEG-4 stored video and audio elementary streams."

One of the reasons this conversion from MP4 elementary streams to HLS (M2TS) is possible is the way that MPEG-4 stores streams. MPEG-4 houses raw elementary streams with integrated DRM using the Common Encryption format, while M2TS stores in the older multiplexed Annex B format that limits playback compatibility. Even if M2TS were used to create separate audio- and video-only transport stream, it is still difficult to recombine them using unrelated encoding timestamps stored in an M2TS stream.

Seamless stream splicing. Fragmented MP4 audio or video elementary streams can be requested separately because each movie fragment only contains one media type. By contrast, given M2TS's multiplexed nature, a chunk normally contains a mix of media types that can't be requested separately. This means that, even if an M2TS could be accessed at a byte-addressable level, the requested byte-range will be a mash-up of metadata, audio and video content, and header information. Any attempt to download multiple multiplexed files and select tracks on the player requires twice the download bandwidth, buffering, and demultiplex processing on the player, and has additional synchronization problems aligning tracks from different transport streams, and splicing them during adaptive switching.

"From the early days of adaptive streaming," said Hughes, "attempts to do bitstream splicing using M2TS relied on strict encoding to allow concatenating one chunk with another chunk to make a decodable stream. Concatenating different audio or packet ordering from two different files results in undecodable streams. When M2TS chunks aren't time aligned or don't start with IDR frames, duplicate chunks must be downloaded in the same time interval and approximate splices made at different audio and video locations—at significant network efficiency and device complexity costs."

Fragmented MP4 sync points are matched by bitrate and media type: audio sync points should all match each other as should video sync points, but sync points for different media types don't need to match each other. Accurate bitstream splices require identical timing for proper audio-video sync between two streams. Time-aligning audio and video elementary streams separately means sync points can be perfectly aligned to Groups of Pictures (GOPs) in video or sync frames in audio.

Using independent sync points for media types along with simple concatenation splicing yields shortened segment lengths: M2TS segment lengths are typically ten seconds, limiting frequency of switching and require extra bandwidth and processing. Segment lengths for fMP4 can be as low as 1 or 2 seconds as they don't consume more bandwidth or require extra processing. Short

segments make quicker adjustments in bitrate to avoid dramatic video quality changes or rebuffer pauses, and more time-accurate splice points for ad insertion.

One challenge for DVD-like Internet delivery is alternate camera angles. Time aligning alternate angles, though, allows adaptive streaming to duplicate the seamless switching of a DVD.

Overlap Splicing: Device Limitations

Overlap splicing is required in adaptive streaming formats such as HLS, when alternate files do not have time aligned switch points that allow requested file chunks from alternate bitrate files to be simply concatenated and run through a single demultiplexer and decoders for audio, video, and subtitles. Overlap splicing requires multiple overlapping file chunks from new and old bitrates to be downloaded and processed in parallel until an entry point in the new video stream is reached in the decode and display sequence and the remaining chunk of the previous bitrate can be discarded. This requires additional buffering, processing, and analysis of the timestamps of two transport streams to conform the contained packets to a common timeline.

The additional memory and processing required for overlap splicing can negatively affect audio/video quality and battery life in a device, but its most serious cost may be network performance. Downloading two file chunks in the same time period and discarding the first portion of the new bitrate chunk and the last portion of the old bitrate chunk roughly cuts network efficiency in half when adaptation is taking place.

This is a huge performance problem in the typical case when switching is required to prevent input buffer underflow and a “Paused for rebuffering” error message. In order to switch to a lower bitrate to prevent underflow, a device must download a file chunk at the existing bitrate as well as a file chunk at the new bitrate in order to maintain playback of the old bitrate while processing the new file chunk to find audio and video splice points. Since the network is unable to keep the buffer full downloading one file chunk per time interval, it increases the risk of a buffer underflow error to require two file chunks to be downloaded in the same timer interval in order to change bitrates.

A format such as fMP4 reduces network bandwidth immediately when switching by requesting the next video fragment at a lower bitrate instead of (not in addition to) the higher bitrate fragment next in sequence. Fragments are concatenated and decoded without any additional processing requirement. Separation of audio, video, and subtitle tracks means each track can be chunked exactly at their access points, and concatenated without disturbing the tracks timeline or decoding.

Integrated Digital Rights Management (DRM). Inherent to the MPEG-4 specification, digital rights and—if necessary—encryption, can both be applied at the packet level. In addition, the emerging MPEG Common Encryption (ISO/IEC 23001-7 CENC) scheme can be combined with MPEG-4 Part 12 to enable DRM and elementary stream encryption.

AVC elementary stream encryption is applied to the NAL (Network Abstraction Layer) unit. Streams can be edited without decryption, enhancing compatibility with different decoders and allowing re-containment (moving encrypted elementary streams from several fMP4 files to a single M2TS file).

With a number of available DRM schemes, however, it is not practical to create a different version of each file for each DRM scheme. MPEG Common Encryption (CENC) scheme shares a multi-DRM scheme approved for UltraViolet (UVVU) that uses five DRM solutions—one each from Adobe, Google, Marlin, Microsoft and the Open Mobile Alliance. Work has been done to make CENC an MPEG standard as well, for interoperability across a number of devices and DRMs. Both Adobe Flash® Access and Microsoft Protected Interoperable File Format (PIFF) reference CENC.

MPEG-DASH players share CENC for elementary streams, although pure M2TS lacks common encryption due to multiplexing and a lack of indexed metadata separate from elementary streams.

Combining CENC to the Common File Format (CFF) championed by the Digital Entertainment Content Ecosystem (DECE) may yield some interesting results. When CENC is ratified shortly as MPEG Part 12 Amendment 3, proponents say CFF and CENC will represent two important steps toward large-scale online video distribution via adaptive delivery of fragmented elementary streams.

Since CFF can also be used outside of the bounds of UltraViolet, significant interoperability may also exist between UltraViolet disc-based playback and online video platforms, in much the same way that the DVD Forum's published specifications for DVD playback guaranteed interoperability between DVD players. It's not out of bounds to think of CFF as the DVD standard for the web.

Network Effects of fMP4 versus M2TS

Beyond the basic functional differences of fragmented MP4 and MPEG-2 Transport Streams, it is worth exploring the effects of each, from the perspective of bandwidth and cache efficiency.

Caching efficiencies. When considering comparative complexity, noted in the sidebar above, it's also worth considering the effect that all the possible combinations has on cache efficiency. Each track and bitrate combination required for M2TS delivery means a higher likelihood that an edge cache will not have the particular combination, resulting in a request back to the origin server.

With fMP4, cache-hit ratios are improved almost by a factor of 100. Since fMP4 manifest files use the same fMP4 fragments from a single elementary stream combined on the device, constant use of the same fragments across multiple combination requests works to increase the cache-hit ratio. The likelihood is higher, then, that a fragment will reside on a nearby edge cache (e.g. same audio fragment used for all video bitrates, resolutions, and camera angles). In our DVD example, the 40 fMP4 elementary streams could populate a nearby edge cache, but many of the 4000 M2TS combinations would reside on the origin server until a particular combination request is made.

Bandwidth efficiencies in segment alignment. A key factor when generating multiple files at different bitrates is segment alignment. HTTP adaptive streams via fMP4 relies on movie fragments each being aligned to exactly the same keyframe, whether for on-demand or live streaming (live

requires simultaneous encoding of all bitrates to guarantee time alignment). A packaging tool that is fragment-alignment aware maintains elementary stream alignment for previously recorded content.

Time alignment and media synchronization to keyframes is equally critical for two-pass encoding. As a best practice, the first-pass output should be used for all second-pass and multibitrate encodings to assure keyframe alignment. But what if time-alignment information is arbitrary?

Arbitrary time alignment creates both device and network efficiency issues. If a device encounters bandwidth deficiency and requests a segment at a different bitrate, arbitrary time alignment means the device will be sent two segments (twice as much data) when a device requests less data. Demultiplexing and splicing disparate multiplexed transport streams into a desired combination of audio-video-text content also impacts device processing power and battery life (sidebar, page 6).

Proprietary solutions like Apple HLS allow arbitrary I-frame alignment but often require downloading two segments at a time, which is detrimental to network efficiency. When bandwidth fluctuates, HLS requires almost twice the amount of bandwidth in order to switch between the current and next-available bitrate available in the manifest.

Let's assume a user has 1 mbps of available bandwidth to view content. If no bandwidth variation occurs, a user would view the entire content piece at 1 mbps. When bandwidth fluctuates, though, a user's player then needs to request a different bitrate to watch the remaining content. A switch needs to occur between the current 1 mbps stream and the next-closest bitrate in the manifest.

In order to accomplish overlap switching, two segments of considerable length—Apple HLS typically uses10-second segments—are simultaneously downloaded to begin overlap splicing. More information about overlap splicing can be found in the sidebar, but the point about network efficiencies should not be lost in the debate between concatenation and overlap approaches: short of the necessity to download two segments for overlap splicing, there is no reason to tax the network in this way at the very moment at which bandwidth has dropped considerably. Yet this is precisely what occurs in the overlap splicing model.

How does an fMP4 approach stack up? Latency goes down and ongoing visual quality goes up.

With shorter segment lengths come lower latencies, a boon for live streaming and one Microsoft exploited in streaming the last few Olympic Games. According to numerous sources, a good balance between efficiency and switching latency is to insert keyframes every 2-4 seconds.

Second, the perception of ongoing quality directly correlates to the length of any given fragment. Seamless switching allows for use of 2-second segments, where the complexity of overlap splicing means an M2TS approach has a typical 10-second segment length. A 2-second increment has a much less visible image quality shift between segments than a 10-second increment. The end result of shorter segments is a lower perceived quality variation between A-B visual comparisons.

The Move Toward Standardization of fMP4

A protocol is only as good as its adoption rate, both in terms of interoperability and critical mass. Many standards, including mobile, MPEG-DASH and UltraViolet (UVVU) standards, are using fMP4.

MPEG-DASH. MPEG, the standards body responsible for MPEG-2 and MPEG-4, is addressing dynamic adaptive streaming over HTTP (MPEG-DASH) through the use of four key profiles—two around CFF for fMP4 and two for MPEG-2 TS. For fMP4 both the live (ISO LIVE) and on-demand (ISO ON DEMAND) profiles reference Common Encryption (CENC or ISO 23001-7). ISO LIVE, a superset of live and on-demand profiles, is close to Microsoft Smooth Streaming, meaning Microsoft could easily update its server and client viewers to meet DASH compliance.

For MPEG-2 TS, there are two profiles: Simple and Main. Simple uses constrained encoding and decoding parameters to comply with M2TS chips in web-connected DVD players or set-top boxes.

The Transport Stream Main profile allows for unconstrained encoders and arbitrary segment lengths that may not begin with an I-frame, similar to the Pantos specification. The "Pantos spec" is Apple's submission to the Internet Engineering Task Force (IETF) for an HTTP streaming format. However, according to an MPEG-DASH panelist at the 2011 Streaming Media West show, the Pantos spec introduced to IETF is merely an informational draft, not a specific standards request.

While DASH-compliant players expect a DASH-compatible manifest—meaning that it's not possible to use an Apple .m3u8 playlist—it's easy to see that a DASH player could handle both standards-compliant M2TS and fMP4 decoding in a single player. Transport Stream Main Profile allows an XML-based manifest (an MPD) to serve M2TS fragments prepared for initial use in HLS. The TS Main profile in DASH offers Apple a path toward standards compliance and interoperability.

Many companies are considering implementation of the Common File Format profiles. One that's been in the news quite a bit recently, Netflix, stated its reasons for supporting MPEG-DASH:

> *The proposed DASH standard defines a way to advertise a range of different streams to a player together with the information it needs to pick which ones to stream. It also defines media file formats suitable for adaptive streaming. The file formats enable efficient and seamless switching between streams, enabling a player to adapt to changing network conditions without pausing playback for re-buffering. The standard considers the differing needs of both on-demand services such as ours, and live services. And it's all based on the use of industry standard HTTP servers.*

DASH's "range of different streams" is also known an Adaptation Set. Representations are listed in an MPD (Media Presentation Description), each with a specified bitrate. A device picks the most likely bitrate for its network, then adjusts bitrates by changing the Representation from which it selects Segments as its buffer either gets too full or to empty based on actual network throughput.

An Adaptation Set may contain the same bitrates for cell and Wi-Fi networks (e.g. 300, 450 kbps).

As with other adaptive bitrate considerations, audio and aspect ratio are key: audio streams should be exactly the same, and while video can be different dimensions and bitrates, the video aspect ratio and frame rates should remain constant between video streams.

Some solutions use a command-line authoring tool to create a variant playlist, based on previously chosen media segment parameters, while others integrate the variant playlist creation directly into the media segmentation workflow. It should also be noted that, while variant or multiple playlists are supported by a number of servers, the client end must initiate a call for the correct playlist. In much the same way that CSS3 is used for browser queries to determine whether the page to be served up is on a desktop, mobile or tablet browser, the server relies on the client player to request the proper variant playlist, depending on whether the player senses a cellular or Wi-Fi connection.

DASH also allows identification of tracks that are entirely random access pictures so they can be scanned backwards and forwards to find the desired spot to resume playback.

UltraViolet. This digital-rights management and delivery standard—using key DRM (digital rights management) schemes from Adobe, Google, Marlin, Microsoft and the Open Mobile Alliance—allows content owners to deliver in a common file format with the Common Encryption standard.

The 70-member Digital Entertainment Content Ecosystem (DECE) and six studios back UltraViolet, which uses ISO Base File Format to ensure codec, DRM, timed text and media format consistency.

This common file format (CFF) in UltraViolet is itself based on Microsoft Protected Interoperable File Format (PIFF) and PIFF 1.3 is compatible at the container layer with CFF. A common file format allows interoperable playback and file copying across a number of consumer electronics (CE) devices and computers as the CFF end goal. CFF files can be played on software- or hardware-based UltraViolet players, using any one of the five CENC common DRM systems via an AES cipher and key identifiers embedded within the fMP4 fragments ('moof').

All three profiles—portable definition (PD), standard definition (SD), and high definition (HD)—use the fMP4 container format (ISO/IEC 14496-12). This ISO container uses H.264/AVC video (ISO/IEC 14496-10), allowing multiple resolutions, aspect ratios, and frame rates for progressive-scan video.

UltraViolet leverages fMP4 to allow for SMPTE Timed Text (SMPTE TT), incorporating Unicode text and PNG graphics for captions/subtitles for the hearing-impaired user (SDH). Sign language, dialog translation, and commentaries can also be displayed through the use of subtitles and subpictures.

Fragments can reside at a CDN or in "the cloud" for easy access by consumers. UltraViolet uses a "digital locker" containing digital rights information for content, but not content itself. UltraViolet's "coordinator" in the cloud stores rights for content purchased or rented by any member of a family but does not store content in the cloud. Retailers who sell UltraViolet content make CFF-compliant UltraViolet video files available for download, and they may be freely copied by consumers because playback is controlled by DRM licenses. UltraViolet purchase rights also entitle users to stream the titles they've purchased, and we can anticipate DASH streaming will be applied to CFF files using the DASH Live and On Demand Profiles designed for that type of fMP4 file format.

Learnings on the Path to fMP4

Major industry players have embraced the fragmented MPEG4 protocol for delivery. Some keep the ISO file formats unaltered, others have chosen proprietary extensions and maintain parallel proprietary functionality.

Two companies that contributed to this white paper—Adobe and Microsoft—are worth noting, not just for their decades-long legacy in creating compelling workflow solutions for their proprietary formats, but also for their journey towards fMP4. Their recent experience with adaptive streaming of fragmented MP4 serves both their users and the standards committees well. For insight into Adobe and Microsoft's learnings on the path to fragmented MP4, see Appendix B.

This doesn't mean either company will necessarily abandon proprietary solutions, as it is easier to enhance delivery platforms that one owns versus one that is standards-based, but both companies are committed to continuing to contribute to emerging industry standards.

Conclusion

M2TS has served its purpose, one-to-many broadcasting, for a number of years. The one-to-one nature of streaming delivery to a variety of platforms—mobile, set-top box and web—calls for a new approach to the functional and network limitations inherent to M2TS.

Fragmented MP4 (fMP4) addresses delivery variability on current mobile or fixed-line networks, providing more than just another HTTP adaptive delivery approach. With fMP4, broadcasters get real benefits: asset management of premium content, protection of assets and consistency of user experience across the growing variety of Internet connected video devices.

Seamless switching between bandwidths, without the need to encode, store, and cache segments for thousands of possible combinations, means that fMP4 can independently select and combine segments of different audio, video, and text tracks from a manifest file in order to target specific device classes or network conditions.

In addition, the re-use of elementary stream segments across multiple track combinations means caching efficiencies for HTTP delivery, leveraging better cache-hit ratios on edge servers while decreasing reliance on origin servers to deliver a consistent flow of segments to end users.

To best utilize an fMP4 workflow, a strong need for standardization exists, to move from proprietary to common file and encryption approaches. Support for the Common File Format (CFF) and Common Encryption (CENC) ensure initial interoperability across multiple online and offline use cases, while the MPEG-DASH profiles and manifests add an additional level of interoperability.

We expect to see companies like Adobe, Microsoft, Netflix and others advocate for interoperable common approaches to mutli-device delivery for consumer/professional content delivery platforms.

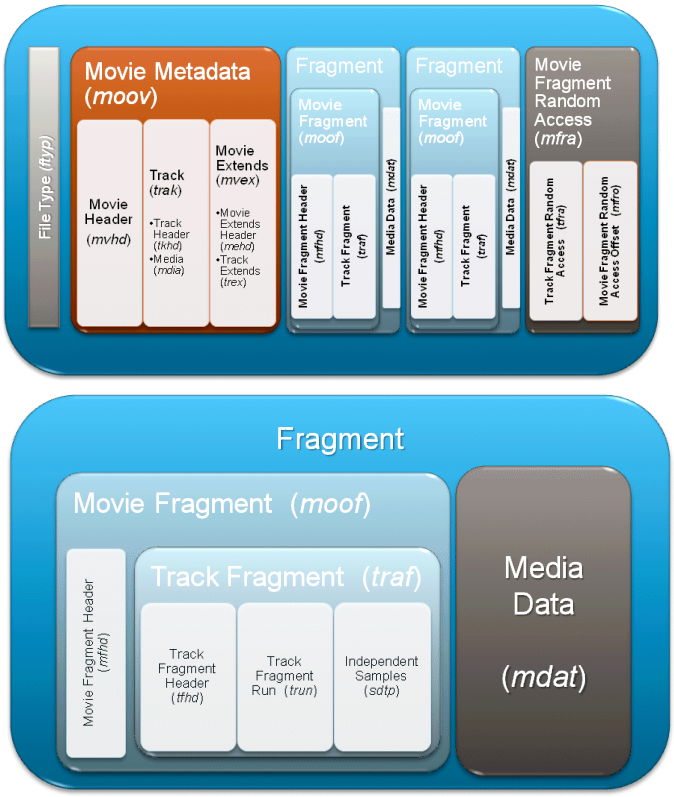
Appendix A: Fragmented MP4 Boxes

A fragmented MP4 file contains a number of discrete units called boxes (also known as atoms). Three key boxes in the fragmented MP4 approach are the 'moov', 'moof' and 'mdat' boxes:

• 'moov' (movie metadata box) contains a virtual movie timeline with display characteristics, duration and timescale elements, as well as sub-boxes providing information for each movie track.

• 'mdat' (media data box) contains media data, raw audio and video information and timed-text elements that are decoded based on a movie box's information.

• 'moof' (movie fragment box) contains short audio, video or text portions of an elementary stream. Movie fragments typically house a closed GOP (Group of Pictures) or AVC Coded Video Sequence.



MPEG-4 file sources did not originally support MP4 file streaming but MPEG is now exploring DASH (Dynamic Adaptive Streaming over HTTP) using MP4 elementary stream fragments as network packets for HTTP streaming delivery.

Coupling an 'mdat' and a 'moof' creates an fMP4 movie fragment that includes one track fragment (either audio or video) with sufficient metadata to make it independently decodable. Using movie fragments to create packets optimized for adaptive streaming is similar in concept to RTP packets of audio/video, but fMP4 movie fragments are optimized for bitrate switching and HTTP delivery over unreliable networks.

The two images above—based on initial Microsoft Smooth Streaming fragmentation—showcase fMP4 integration of 'moof', 'moov' and 'mdat' boxes. The images come from Alex Zambelli's blog post which offers more details.

Appendix B: Adobe & Microsoft Thoughts on Adoption of fMP4

Two companies that contributed to this white paper—Adobe and Microsoft—shared thoughts on their journey towards fMP4.

Adobe. Ask members of the Adobe Flash Media Server team what they do, and you'll likely get this response: *Adobe is in the business of delivering video experiences. Period.*

For more than a decade, that business centered on the Real-Time Media Protocol (RTMP) that Adobe launched back in 2001 with Flash Player 6. RTMP stood at the heart of Adobe Flash Media Server (FMS) and many de facto workflows sprung up around the protocol, although it was extended over the years to address encryption (RTMP-E) and peer-to-peer delivery (RTMFP).

One such enhancement to FMS was a precursor to Adobe's foray into HTTP delivery. Back in December 2009, with the release of FMS 3.5.3, Adobe essentially decoupled the buffer from the connection, allowing semi-stateless connection between player and server and empowering Flash Player to continue play back even if the connection dropped. Developers can use ActionScript to reconnect to Flash Player to FMS and, if reconnection occurs before the buffer empties, there is no perceived disruption in client playback, critical for mobile video delivery on intermittent networks.

Adobe saw opportunity to provide similar workflows for devices that did not use its Flash Player plug-in. In 2010, Adobe announced Adobe HTTP Dynamic Streaming (HDS), an HTTP-based adaptive streaming approach that built on fMP4 as its base protocol.

"This is a very different approach than our RTMP delivery with Adobe FMS, which uses normal FLV or MP4 (F4V) file formats," said Kevin Towes, Adobe's senior product manager for delivery. "The MP4-Fragment format [Adobe calls it F4F] adheres to the industry standard which is important for re-usability. The format also enabled a major requirement for media streaming – protection—allowing you to deliver your live or pre-recorded content, leverage the caching devices, and still maintain control over your video assets."

As part of the new Flash Media Server, announced in September 2011, Adobe enhanced support for both manifest (F4M) and fragment (F4F) file formats. Adobe uses an XML-based solution it calls F4M version 2, similar in concept to a variant playlist in what Adobe calls "set level manifests".

"Workflows in RTMP have more than a decade of robustness behind them," said Towes, "but the same workflows haven't been as easy in HTTP. Our investment into the HTTP workflows—bringing them on par with RTMP workflows—allows established FMS customers to maintain their sizable desktop/laptop viewer base while also pursuing new iPad and other iOS device revenue streams."

Modifications to F4F allow real-time packaging, protection and features such as variant playlists to act on par with RTMP-based workflows. For encryption, FMS encrypts F4F fragments within FMS, using either a built-in Flash Access encryption module or a stand-alone Flash Access server.

Adobe emphasizes the point that fMP4 natively allows for network optimization and content encryption, but it also notes that different devices require different optimizing tweaks for delivery. Leveraging FMS for both desktop and mobile devices, including creation of HLS-based transport streams, Adobe feels it can supply content to every device at optimal resolutions and bandwidths.

Seeking is one such area: given the ability to identify and deliver a byterange via fMP4, Adobe can bring its “seek” enhancement from RTMP directly into an HTTP delivery scenario. When enabled, seeking occurs first within the buffer, significantly reducing the load on the server compared to traditional back-and-forth seeks between client and server. The duration of the buffer can be dynamically changed, something not currently possibly in MPEG-2 TS solutions, to allow for pseudo-DVR functions such as instant replay or simple time-shifting options within the player. This also allows playback at different speeds and frame-accurate stepping from scene to scene.

“We try to help broadcasters realize new revenue streams,” said Adobe’s Kevin Towes, “regardless of platform being delivering to. Where Flash Player is available, we'll use RTMP or HTTP Dynamic Streaming to deliver world-class quality of experience, encryption and rights management. With other platforms, such as iOS-based devices, we'll do an equally good job. Adobe also sees the benefit of standards-based approaches like the Common File Format, HTML5 and MPEG-DASH.”

Microsoft. Microsoft had a significant installed Windows Media® user base for well over ten years. Yet the company announced in late 2008 it was moving toward fragmented MP4 with the launch of Smooth Streaming, a first-to-market adoption of fMP4. Smooth Streaming combines Windows Server® and Silverlight® (or other Smooth Streaming clients) with XML and SMIL manifests.

Manifests have an *.ismv extension. Manifests indicate to an adaptive streaming client what bitrates are available in a file set, where fragments reside, and header parameters from each video file.

Microsoft stores like-bitrate "chunks" of video as movie fragments in a single file, using timecode information stored in each movie fragment (moof) as an index for particular segments to download and play back. Smooth Streaming devices download movie fragments containing either a video closed Group of Pictures (GOP) or an integral number of audio sync frames, via a single HTTP request. The player than select segments (typically 2-4 seconds long) from multiple streams encoded with different bitrates, resolutions, languages, etc.

What were Microsoft’s reasons for choosing fMP4 over ASF? The company notes five:

- MP4 is a lightweight container format with less overhead than ASF
- MP4 is easier to parse in managed code (.NET) than ASF
- MP4 is itself a widely used standard, making 3rd party adoption / support more straightforward
- MP4 was architected with H.264 video codec support in mind
- MP4 was designed to natively support payload fragmentation within the file

"The MPEG-4 fragmented format was able to do everything we needed," wrote Ben Waggoner, principle program manager at Microsoft, "so it was simplest to just use that rather than make up something new. We're not trying to make up a new file format here; just take advantage of existing technologies in a novel way."

"ASF was a great format for bit-pumping," wrote Waggoner in 2009, "with a MIPS-per-Mbps ratio better than other formats of its era. However, it simply wasn't well set up to allow a byterange to encapsulate a fragment that could potentially be a single closed GOP. Since server hardware is so much more powerful these days and proxy caching so dramatically reduces the load on the origin server, going for a somewhat more complex server-side parsing methodology made good sense."

Microsoft chose to go "all in" around the standard, including use of *.ismv and *.mp4 extensions.

What have they done with Smooth Streaming since its announcement in 2008?

Quite a lot, actually. Smooth Streaming has seen significant traction with Silverlight, but doesn't require Silverlight to deliver adaptive bitrate HTTP streaming, as evidenced by Smooth Streaming clients such as the Comcast® XFINITY® TV App for iOS devices and Linux-based Netgem set-top boxes. Such non-Silverlight clients can be created using a Smooth Streaming client source-porting kit from Microsoft that enables playback on any device.

In addition, publication of a Smooth Streaming specification, called PIFF (protected interoperable file format) has been adopted—along with Microsoft PlayReady® DRM technology—for use in new Netflix ready devices and applications, including the Apple iPad.

Microsoft has also enabled several compelling scenarios—including advertising and alternate language tracks—on top of fMP4. In addition, the Microsoft ecosystem surrounding the Smooth Streaming format includes two key tools: Expression® Encoder for live- and on-demand Smooth Streaming encoding and IIS Media Services. The latter rides on top of the Microsoft Internet Information Services (IIS) web server that ships in every version of Windows Server 2008.

Another very interesting part of the workflow, that I've covered for several trade publications as well as the Workflowed blog, is IIS Transform Manager, a server-side transcoding and transforming tool. Transform Manager can convert a variety of formats, including segmented .mp4 files, to Smooth Streaming presentations. It can also convert Smooth Streaming content for use on Apple iOS devices, converting from fMP4 to HLS (MPEG-2 TS) as part of its standard workflow. Transform Manager scales out on multi-core HPC clusters, plus allows audits of all file copies/movement to improve monitoring/management control.

Microsoft is committed to supporting the standards-based approach, including the Common File Format (CFF), Common Encryption (CENC) and the MPEG-DASH adaptive delivery over HTTP.

Acknowledgements

Special thanks to Microsoft Corporation and Adobe Systems for providing financial resources and access to subject-matter experts. The following subject-matter experts spent time expanding on key concepts and the ever-changing nature of fragmented MP4 as well as the MPEG-DASH ratification process and use cases:

- Adobe Systems. Srinivas Manapragada, Kevin Streeter, Vishy Swaminathan, Kevin Towes
- Microsoft. John Deutscher, Kilroy Hughes, Chris Knowlton, David Sayed, John Simmons, Sudheer Sirivara
- Netflix. Mark Watson

About the Author

Tim Siglin, co-founder of Transitions, Inc., holds an MBA with emphasis in entrepreneurship. Involved with visual communications for over fifteen years, Siglin's key focus is digital media entrepreneurship, including market analysis, implementation and product launches. He has served in executive roles in marketing, strategy and technology for a number of startups. He also serves as a contributing editor for several tech publications.

About the Company

Transitions, Inc., is a technology and business development firm with extensive experience in technology design and go-to-market strategy consulting. Transitions specializing in assistance to businesses seeking to identify "transition points" that hinder growth or a return to profitability. We also offer research and consulting services to several Big 5 consulting firms and internal skunk work project coordination for Fortune 500 clients.

Repeat customers account for more than 80% of all on-going business, but Transitions also takes on select project challenges assisting startups, distressed and expanding small businesses. Based in Tennessee, Transitions' business strategy and marketing consulting clients include companies in Silicon Valley, Boston, London, Milan, Mumbai, New York and Switzerland. Ongoing Transitions' projects include established businesses and startups in the digital media, financial services and global marketing industries.