Deploying CMAF In 2019
INTRODUCTION

@johngainfort
OUTLINE

- What is CMAF?
- What are the key features?
- Advanced considerations?
- Future of CMAF?
- Conclusion
WHAT IS CMAF?
WHAT IS CMAF?
Overview

What is CMAF?

- Specifies the container, NOT the manifest (manifest agnostic)
- It is an ISOBMFF, fMP4 container, specifically ISO/IEC 14496-12:201
- Common Encryption (CENC) - ISO/IEC 23001-7: 2016
- Baseline supports HEVC, ACV, and AAC with interoperability (VP9, etc.)
- Captioning/Subtitling - WebVTT, IMSC-1 & CEA 608/708
- **REQUIRES** non-muxed audio and video segments
WHAT IS CMAF?

Overview

Translation?

- Specifies the container, NOT the manifest (manifest agnostic)
  - USE HLS OR DASH
- It is an ISOBMFF, fMP4 container, specifically ISO/IEC 14496-12:201
  - SAME SEGMENTS WORK IN EITHER - THEY ARE FRAGMENTED MP4 (WHAT DASH USES)
- Common Encryption (CENC) - ISO/IEC 23001-7: 2016
  - DRM IS...COMPLICATED - SORRY
- Baseline supports HEVC, ACV, and AAC with interoperability (VP9, etc.)
  - CODEC DOESN'T MATTER
- Captioning/Subtitling - WebVTT, IMSC-1 & CEA 608/708
  - SAME OLD CC/SUBTITLE OPTIONS
- REQUIRES non-muxed audio and video segments
  - OHH....CRAP (MAYBE)
WHAT IS CMAF?

Overview

Performance

- Can perform much better on some devices with CMAF vs HLS-TS

- Direct pipe to MSE vs client side transmux
  - Browsers
  - Web views on devices
  - Limited hardware devices like smart TV’s
  - Anywhere HLS isn’t native but supports HLS or MSE has potential gain
WHAT ARE THE KEY FEATURES?

- Unified segments across different manifests
- DRM - Common Encryption
UNIFIED SEGMENTS
KEY FEATURES OF CMAF

Unified Segments

### BEFORE

- HLS .m3u8
- DASH .mpd

<table>
<thead>
<tr>
<th>TS</th>
<th>TS</th>
<th>TS</th>
<th>...</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASH/MP4</td>
<td>DASH/MP4</td>
<td>DASH/MP4</td>
<td>...</td>
<td>DASH/MP4</td>
</tr>
</tbody>
</table>

- Protocol selection was a long-term commitment to manifests and media containers.
- Maintaining two complete workflows was expensive

### AFTER

- HLS .m3u8
- DASH .mpd

| CMAF | CMAF | CMAF | ... | CMAF |

- Protocol selection is now a MANIFEST preference because media containers are common
1. Lower production costs associated with encoding, packaging, and storage.

2. Significantly improve CDN cache efficiency, especially for Live.

3. Foundation to define additional conformance points for content and devices, regardless of manifest protocol. (i.e., CTA WAVE)

4. Most importantly, provides common ground for industry cooperation
KEY FEATURES OF CMAF

Unified Segments

IMPORTANT NOTE REGARDING HLS

- Full support as of HLS v6

```text
#EXT-X-VERSION: 6
```

KEY FEATURES OF CMAF

DRM - Common Encryption

BENEFITS

● Cross Platform and Media Format Compatibility
  ○ Supported out of the box by AVPlayer, ExoPlayer, and Safari

● Single Encoding and Encryption
  ○ Because of Common Encryption (CENC)

● Increased Performance
  ○ Does not require transmuxing MPEG-2 TS into ISOBMFF for MSE.
  ○ Single packaging workflow (mostly)
  ○ Single publishing workflow (mostly)
KEY FEATURES OF CMAF

DRM - Common Encryption

TERMINOLOGY

- **CENC AES-CTR** or **cenc**: CENC Protection Scheme using AES 128-bit keys in Counter Mode (AES-128 CTR)

- **CENC AES-CBC** or **cbc1**: CENC Protection Scheme using AES 128-bit keys in Cipher-block chaining mode (AES-128 CBC)

- **CENC AES-CTR Pattern** or **cens**: CENC Protection Scheme using AES 128-bit keys in Counter Mode (AES-128 CTR) using pattern of unencrypted/encrypted bytes

- **CENC AES-CBC Pattern** or **cbcs**: CENC Protection Scheme using AES 128-bit keys in Cipher-block chaining mode (AES-128 CBC) using pattern of unencrypted/encrypted bytes,
KEY FEATURES OF CMAF
DRM - Common Encryption

CHALLENGES

● CBCS Compatibility
  ○ Widevine CBCS is not currently supported in Firefox (although they claim it works)
  ○ Playready CBCS is not currently supported in Edge
  ○ Likely to change by end of year or next year.

● CMAF Does Not Allow Muxed Audio
  ○ Both audio and video renditions need to be decrypted
  ○ Possibility of different decryption keys

● Open Source Player Availability is Limited
  ○ Shaka Player does not support Playready and Fairplay for HLS [2]
  ○ Hls.js does not support ANY DRM
  ○ Both platforms have ongoing efforts for support [3,4,5,6]
KEY FEATURES OF CMAF

DRM - Common Encryption

OPTIONS

- Two Manifests, Two Fragment Encryptions for Widevine
  - CBCS to support Apple, Android (7.0+), Roku (9.0+), Chrome, and Safari
  - CTR to support Firefox and Edge (Likely to change next year)
    - Older Versions of Android (4.4 -> 7.0)

- Roll your own DRM workflow for your player using EME
  - EME is designed to enable the same app and encrypted files to be used in any browser, regardless of the underlying protection system [7]

- Contribute Back to the Community!
## Key Features of CMAF

**DRM - Common Encryption**

### Support

<table>
<thead>
<tr>
<th></th>
<th>Apple</th>
<th>Android</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Edge</th>
<th>Safari</th>
<th>Roku</th>
<th>Chrome cast</th>
<th>XBox One</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBCS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>/</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTR/CENC</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
KEY FEATURES OF CMAF
DRM - Common Encryption

REFERENCES

1. https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-cenc-with-multidrm-access-control
2. https://github.com/google/shaka-player#hls-features
5. https://github.com/video-dev/hls.js/pull/2194

Great Resource: https://www.linkedin.com/pulse/its-cbcsing-time-phil-harrison/
ADVANCED CONSIDERATIONS

- Mixed Content & DAI
MIXED CONTENT & DYNAMIC AD INSERTION
CONSIDERATIONS WITH DYNAMIC AD INSERTION

- REMEMBER -> CMAF requires non-muxed audio

- If stitching ads be sure ads also have non-muxed audio in ads
  - Still can be TS or CMAF (FMP4) if HLS

- Applies to SSAI & CSAI

- If switching between TS and CMAF in HLS best to use a discontinuity
FUTURE OF CMAF?

- Low Latency
LOW LATENCY
CLARITY - WHAT IS LOW LATENCY?
Latency Achievable at Scale Via Mainstream CDNs Streaming Technologies

### BUSINESS USE CASES
- **BROADCAST LATENCY – DTT, DTH, CABLE DISTRIBUTION**
- **LIVE SPORTS AND e-SPORTS**
- **VOICE**
- **LIVE AUCTIONS**
- **GAMBLING**
- **BETTING**
- **SOCIAL MEDIA – SMS/IP**

### DISTRIBUTION TECHNOLOGY
- **OTT STREAMING WITH HLS & DASH**
- **CHUNKED CMAF segments 1s-6s**
- **10s segments**
- **6s segments**
- **2s segments**
- **1s segments**

### Latency Ranges
- **LEGACY LATENCY RANGE**
  - 45+ seconds
- **LOW LATENCY RANGE**
  - 30 seconds
  - 10 seconds
  - 8 seconds
- **ULTRA LOW LATENCY RANGE**
  - 6 seconds
  - 4 seconds
  - 2 seconds
  - 1 second
- **SUB-SECOND**
  - 200 ms

- Typical Broadcast Latency
FUTURE OF CMAF
Low Latency

- Content generation & notification delay

2 sec KFI

Segment1 → Segment2

6 seconds

[Upload to Origin] → [Update Manifest] → [Propagate To Edges] → [Deliver To Client] → [Buffer 3 Segments] → [Play]
FUTURE OF CMAF
Low Latency

BOTH IMPORTANT - NOT THE SAME THING

- **Chunked Encoding - [How it is made]**
  - How the encoder breaks down, saves and sends data

- **Chunked Transfer Encoding - [How it is delivered]**
  - How the data is sent as it is made available without knowing final size. The transmission ends once a zero-length chunk is sent.
FUTURE OF CMAF

Low Latency

1. **Encode**: CMAF segment content is chunked-encoded.
2. **Manifest Update**: Manifest/Playlist signals the usage of chunked encoding and pre-announces availability of the data chunks.
3. **Push**: Content is pushed to the origin via HTTP 1.1 chunked encoding transfer.
4. **Distribute**: Content is propagated through the CDN tiers to the client using HTTP chunked encoding transfer at each step through the distribution chain.
5. **Player**: Client must ‘stream’ the content as it receives
   a. Must not wait for end of segment
   b. Use Fetch() versus XHR API to read/stream the response body while still being downloaded
FUTURE OF CMAF

Low Latency

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Protocol</th>
<th>Ty</th>
<th>Initiator</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunk-stream_t_0-10575.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.73 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10576.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.79 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10577.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.77 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10578.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.86 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10579.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.80 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10580.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>low-latency-play...</td>
<td>1...</td>
<td>5.82 s</td>
</tr>
</tbody>
</table>

Player at live edge receiving chunk-transferred segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Protocol</th>
<th>Ty</th>
<th>Initiator</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunk-stream_t_0-10575.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>1.15 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10576.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>862 ms</td>
</tr>
<tr>
<td>chunk-stream_t_0-10577.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>1.46 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10578.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>2.24 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10579.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>495 ms</td>
</tr>
<tr>
<td>chunk-stream_t_0-10580.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>1.15 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10581.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>1.73 s</td>
</tr>
<tr>
<td>chunk-stream_t_0-10582.m4s</td>
<td>200</td>
<td>http/1.1</td>
<td>fet</td>
<td>service_worker/js...</td>
<td>1...</td>
<td>3.44 s</td>
</tr>
</tbody>
</table>

Player 18s behind live receiving cached segments
CDN SCALING

- Chunked encoding transfer HTTP 1.1 is based on 21-year old technology
- Widely supported across almost all modern CDNs
- Able to leverage their existing cache systems to achieve scale.
- Proven to support millions of viewers regardless of low-latency stream
CONCLUSION
CONCLUSION

- Unifies Segments
- Unifies DRM ... almost
- Performance Improvement
- Lowers Costs / CDN Cache Efficiency
- Sets you up for the future - Low Latency!
CONCLUSION

PRO’s

● Unified content is REALLY good: Cacheability
● Better performance for MSE implementations
● Scalable low latency capable, utilizing industry standards & readily available
● DRM Unification to CBCS is in the realistic future 9-36 months (a solid guess)

CON’s

● External (non-muxed) audio may cause challenges if not using already
● Some legacy players/platforms may not support
  ○ Should be less than 5% on average
CONCLUSION

ROADMAP TO DELIVER

- Basic CMAF Delivery
  - Encoder - Output basic CMAF segments and manifests
- Multi-DRM Delivery (CBCS & CENC-CTR)

- Low Latency Encoder Update (if necessary)
- Validate CDN Chunked Transfer Delivery
- Low Latency Player Updates
RESOURCES

- **General**
  - [https://bitmovin.com/what-is-cmaf-threat-opportunity/](https://bitmovin.com/what-is-cmaf-threat-opportunity/)

- **Low Latency**
  - [https://www.wowza.com/blog/low-latency-cmaf-chunked-transfer-encoding](https://www.wowza.com/blog/low-latency-cmaf-chunked-transfer-encoding)

- **DRM**
  - [https://gpac.wp.imt.fr/mp4box/encryption/common-encryption/](https://gpac.wp.imt.fr/mp4box/encryption/common-encryption/)