Standards in the UHD Streaming Backend

HbbTV Webinar 26th of April

Dr.ir. Rufael Mekuria Lead Research Engineer Unified Streaming

Outline of Presentation

Unified Streaming
Platform

Introduction

Backend according to DASH-IF

DASH-IF Live Media Ingest Protocol

Content Protection Interchange Format (CPIX)

Supporting custom DASH features on-the-fly

Upcoming standards in this area

Short Introduction

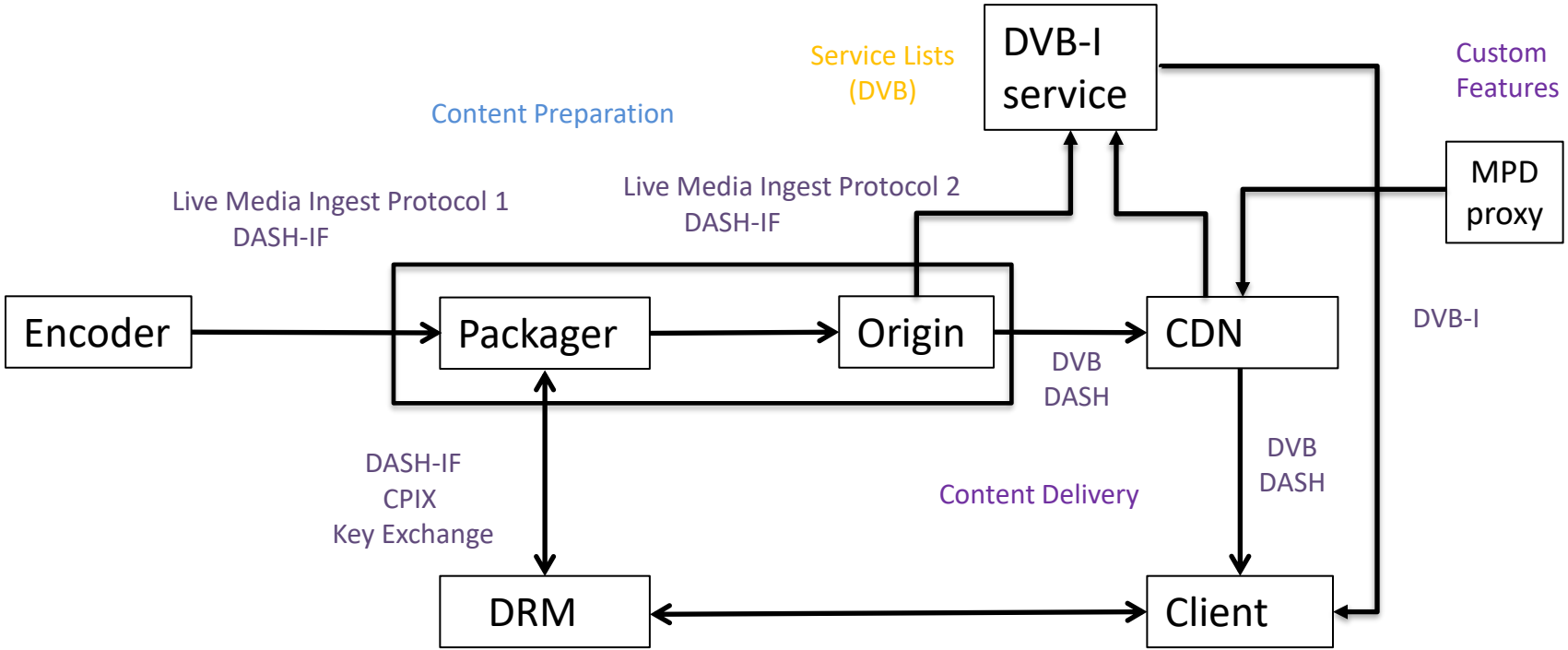
Unified Streaming is a company that makes software, mainly packager, origin, content stitching, software that enables you to create solutions for streaming VoD, Live NPVR etc.

Unified Streaming products are used around the globe and extensively documented docs.unified-streaming.com

I joined Unified Streaming in 2016, as research and standards engineer working on H2020 5G Superfluidity and standards projects (DASH-IF, MPEG, DVB, CTA). I am currently editor and lead of DVB streaming group (TM-STREAM).

Before I did my PhD at CWI and started a project in MPEG on volumetric video coding and point cloud coding. This project was joined by all major mobile device manufecturers (Apple, Samsung, Huawei and Sony) at the end of my PhD.

Generic Backend [DASH-IF]



The heavy Job: The ABR Encoder

Unified Streaming Platform

HEVC and AVC Video profiles for DVB DASH
https://www.etsi.org/deliver/etsi_ts/101100_101199/101154/02.04.01_60/ts_101154v020401p.pdf

Video Codec (AVC and HEVC, main and high profile)
Colorimetry usage (BT.709, BT.2020 BT.2100)
HDR support HLG10, PQ10, dynamic metadata
Audio codecs (MPEG/Dolby/DTS)

How to best output content from the encoder ? ? ?

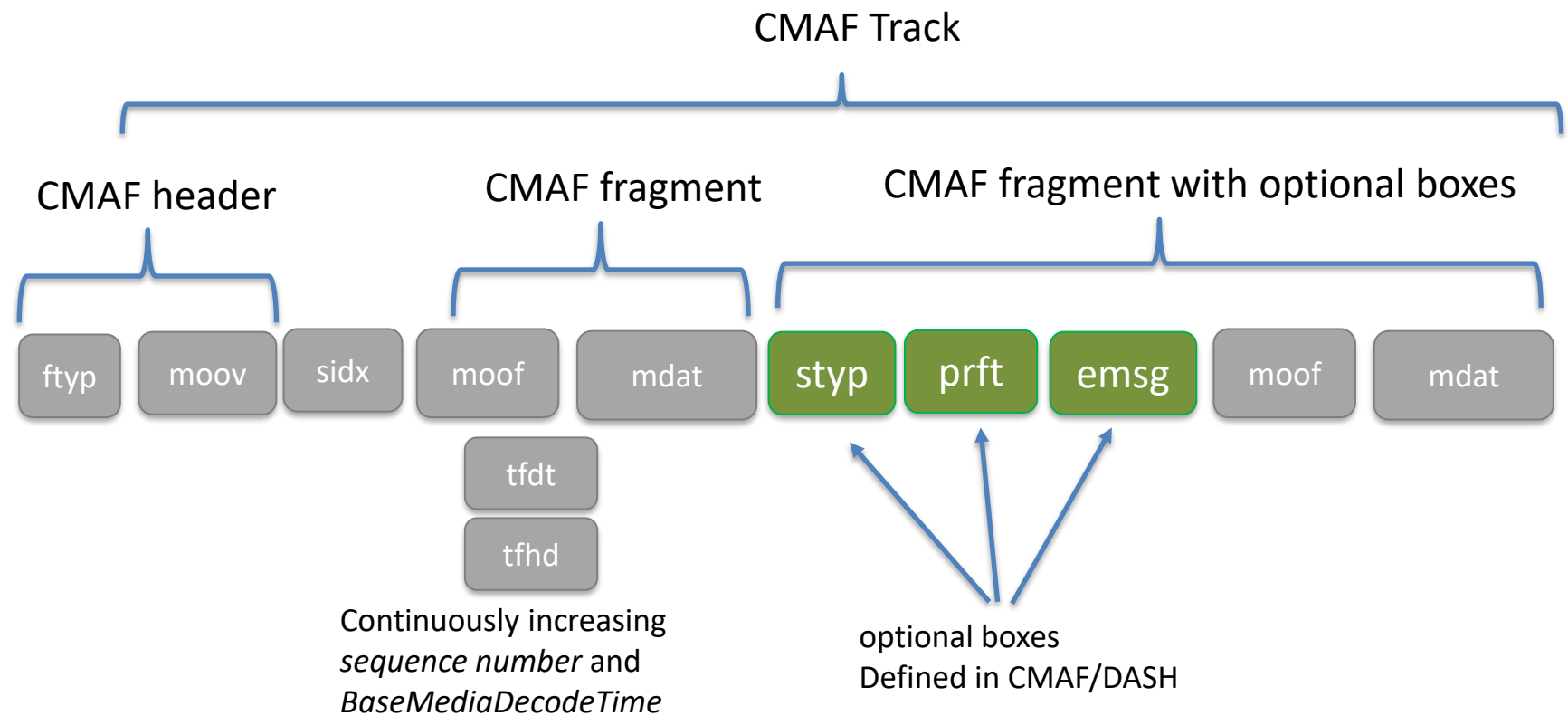
Encoder output

What format does the ABR encoder need to output ?

CMAF specification (2018):
https://www.iso.org/standard/71975.html

Fragmented MP4 based on ISOBMFF
Guidelines for colometry and aspect ratio for ***seamless switching***
Optional **Track Role** signaling and **Accessibility** signaling in *kind* box
CMAF Tracks can carry video, audio, subtitles, timed metadata
Most of the signalling (not all) in DASH/HLS also exists or can be derived from CMAF
Active work is done on creating profiles for codecs and technologies, therefore,
CMAF is future proof as bindings for new technologies are actively developed

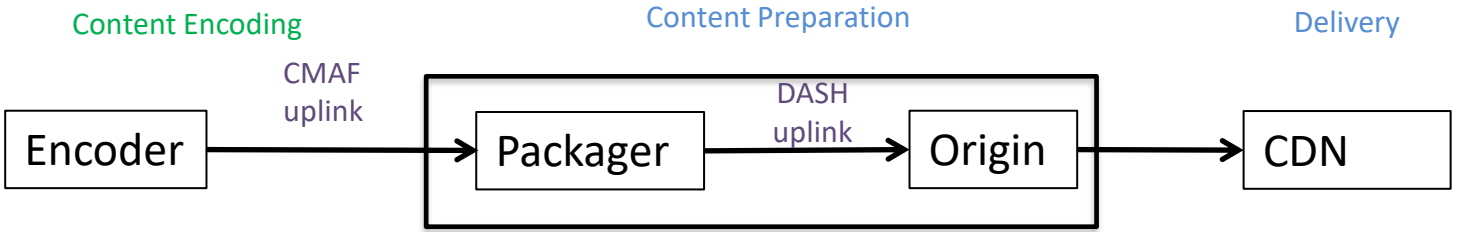
CMAF Track Example



Live Media Ingest

HTTP POST for pushing fragments downstream

- DASH-IF technical specification:
- https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html
- Uses CMAF and optional DASH manifest describing grouping and naming
- Generate final delivery manifest downstream



Benefits of live media ingest

Use CMAF and **HTTP POST** for pushing fragments downstream:

- Easy to work with new codecs and profiles
- Easy to enable redundant and fault tolerant workflows, including setups with multiple encoders/packagers and origins (more later).
- Distributed encoding: e.g higher bit-rate ladder can be encoded on different machines
- Retransmission of segments in case of errors and packet loss errors are avoided
- All media, timed text, subs and metadata are supported
- Implementation available in FFMpeg, encoder vendors
- Offload encoder, avoid single point of failure
- FFMpeg demo with multiple encoders: https://github.com/unifiedstreaming/live-demo-cmaf

Content Protection Interchange

For UHD Multi-key is often desired, to acquire and exchange complex key information between network distributed entities CPIX is recommended to be used.

Content Protection Interchange Format (CPIX) can be used to exchange Key information between DRM provider, packager and encoder

CPIX defines an XML schema for carrying content key and encryption information, the **protocol/API** for information exchange is not standardized

DASH-IF and pending **ETSI standard**:
https://dashif-documents.azurewebsites.net/CPIX/master/Cpix.html

Information in a CPIX Document

- **ContentKey**

  A CPIX doc Must have a Key ID (KID) used to identify the content and associate it with a (secret) Content Encryption Key. It May have a Content Encryption Key (CEK), which is used to encrypt the content.
- **DRMSystem,** a cpix document, must have a System ID which represents a specific DRM system such as Microsoft PlayReady. DASH-IF defines and documents different DRM system IDs.
- Must have a Key ID which must refer to an existing Content Key's KID.
- Optionally, it has a Protection System Specific Header (PSSH) element. Depending on the DRM system, contains protection information such as such as licenses, rights, and license acquisition information.

- Optionally has **ContentProtectionData** used for signaling DRM in the MPEG-DASH playout manifest.
- Optionally has **HLSSignalingData** used for signaling DRM in the Apple HLS Manifest.
- Optionally has **SmoothStreamingProtectionHeaderData** used for signaling DRM in the Microsoft Smooth Streaming playout manifest.
- Optionally has **HDSSignalingData** used for signaling DRM in the HTTP Dynamic Streaming playout manifest.

Minimal CPIX Example

Minimal CPIX example

The following example shows a minimal CPIX document:

```
<?xml version='1.0' encoding='UTF-8'?>
<CPIX xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:xsi="http://www.w3.org/2001/XMLSchem
  <ContentKeyList>
    <ContentKey kid="e82f184c-3aaa-57b4-ace8-606b5e3febad">
      <Data>
        <pskc:Secret>
          <pskc:PlainValue>wvr2bihSzExKdR8KKpQf2w==</pskc:PlainValue>
        </pskc:Secret>
      </Data>
    </ContentKey>
  </ContentKeyList>
  <DRMSystemList>
    <!-- Widevine -->
    <DRMSystem kid="e82f184c-3aaa-57b4-ace8-606b5e3febad" systemId="edef8ba9-79d6-4ace-a3c8-27dcd
      <PSSH>AAAAMnBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABIiCnVzcHd2dGVzdDNI49yVmwY=</PSSH>
      <ContentProtectionData />
      <HLSSignalingData />
    </DRMSystem>
  </DRMSystemList>
</CPIX>
```

CPIX Example with multiple keys

```
<?xml version='1.0' encoding='UTF-8'?>
<CPIX xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:dashif:org:cpix" xsi:schemaLocation="urn:dashif:org:cpix
cpix.xsd"> <ContentKeyList>
   <ContentKey kid="e82f184c-3aaa-57b4-ace8-606b5e3febad">
    <Data> <pskc:Secret> <pskc:PlainValue>...</pskc:PlainValue></pskc:Secret> </Data>
   </ContentKey>
   <ContentKey kid="087bcfc6-f7a5-5716-b840-6aa6eba3369e">
    <Data><pskc:Secret><pskc:PlainValue>...</pskc:PlainValue> </pskc:Secret> </Data>
   </ContentKey>
   <ContentKey kid="0d6b4023-8da1-5e75-af68-75c514c59b63"><Data> <pskc:Secret>
       <pskc:PlainValue>...</pskc:PlainValue></pskc:Secret> </Data>
   </ContentKey>
  </ContentKeyList>
  <DRMSystemList>
<!– ommited -->
</DRMSystemList>
  <ContentKeyUsageRuleList>
   <ContentKeyUsageRule kid="e82f184c-3aaa-57b4-ace8-606b5e3febad">
    <VideoFilter maxPixels="589824"/>
   </ContentKeyUsageRule>
   <ContentKeyUsageRule kid="087bcfc6-f7a5-5716-b840-6aa6eba3369e">
    <VideoFilter minPixels="589825" maxPixels="2073600"/>
   </ContentKeyUsageRule>
   <ContentKeyUsageRule kid="0d6b4023-8da1-5e75-af68-75c514c59b63">
    <AudioFilter/>
   </ContentKeyUsageRule>
  </ContentKeyUsageRuleList>
</CPIX>
```

Python CPIX tool

pycpix - a Python library for working with CPIX 2.2 documents: https://github.com/unifiedstreaming/pycpix
cpix-gen - a CPIX generator tool - a fully installed, dockerised version of pycpix. https://github.com/unifiedstreaming/cpix-gen/

Example
To create a simple CPIX document with a single key:

```
import cpix

full_cpix = cpix.CPIX(
   content_keys=cpix.ContentKeyList(
      cpix.ContentKey(
         kid="0DC3EC4F-7683-548B-81E7-3C64E582E136",
         cek="WADwG2qCqkq5TVml+U5PXw=="
      )
   ),
   drm_systems=cpix.DRMSystemList(
      cpix.DRMSystem(
         kid="0DC3EC4F-7683-548B-81E7-3C64E582E136",
         system_id="EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED",
         pssh=("AAAAxnBzc2gBAAAA7e+LqXnWSs6jyCfc1R0h7QAAAAINw+xPdoNUi4HnPGT"
               "lguE2FEe37S9mVyu9EwbOfPNhDQAAAIISEBRHt+0vZlcrvRMGznzzYQ0SEF"
               "rGoR6qL17Vv2aMQByBNMoSEG7hNRbI51h7rp9+zT6Zom4SEPnsEqYaJl1Hj"
               "4MzTjp40scSEA3D7E92g1SLgec8ZOWC4TYaDXdpZGV2aW5lX3Rlc3QiEXVu"
               "aWZpZWQtc3RyZWFtaW5nSOPclZsG")
      )
   )
)
```

Custom Features Using MPD Proxy

- DVB-DASH and other profiles of DASH use specific features/properties
- Using an MPD Proxy or MPD manipulator can be attractive to customize manifest/playlist
  - Adding a descriptor
  - Adding Service description element
  - Re-order the adaptation sets in the mpd
  - Introducing adaptation set switching property to switch between adaptationsets
  - Adding or removing labels
  - Remove sidecar or webvtt subtitles adaptation sets
  - Change track roles
  - .....
  - https://docs.unified-streaming.com/documentation/manifest-edit/index.html

Upcoming standards

- DVB-DASH TA signalling (DVB-TA part 3)
  - Signalling of SCTE-35 based ad breaks in DVB-DASH
  - Reporting and content/asset identification for ads
  - For client and server side ad insertion
  - Joint work with SCTE DVS WG5 and WG7

- MPEG encoder Synchronization
  - Proposal for encoder synchronization using DASH-IF live media ingest and epoch counting
  - Work in progress
  - Presentation at MHV 2022 https://dl.acm.org/doi/abs/10.1145/3510450.3517313
  - Demo stream on our webpage: demo.unified-streaming.com/k8s/live

Conclusion

Standardized interfaces can help to implement UHD live streaming at scale

- **DASH-IF Live Media ingest**
  - Future proof and support for new codecs
  - Distributed setups and encoder synchronization
  - Offload the encoder
- **DASH-IF CPIX**
  - Distribute Keys in interoperable manner
  - Tools and examples available
  - Supported by USP, AWS and most DRM vendors
- **MPD Proxy**
  - Many DASH profiles (DVB, DASH-IF,SCTE) use custom options
  - Manipulate the manifest downstream with an MPD proxy