ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ

(MSc)

στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Μετάδοση Οπτικοακουστικού Υλικού με Τεχνολογίες IP"

Βυζικίδης Στέφανος

ΜΜ4130006

ΑΘΗΝΑ, ΙΟΥΝΙΟΣ 2016

Στην οικογένεια και τους φίλους μου

για τη στήριξη τους

Contents

Abstract

Distribution of video over the Internet is becoming mainstream and global IP video is expected to dominate Internet traffic by 2018. Consumers expect high quality video streaming to be available on every Internet-connected device, over any available network connection. More importantly, they expect high-availability and demand a seamless user experience. Delivery of video over IP networks exists since mid-90s but traditional methods like RTP/RTSP or more recently Progressive Download fail to deliver the experience expected today. The ecosystem has evolved mainly due to the introduction of smartphones and other internet-connected devices, but also due to the development in speed and reliability of mobile and broadband networks. Introduction of HTTP Adaptive Streaming technologies from market leaders like Apple, Microsoft and Adobe brought solutions but introduced also inherent weaknesses. Technologies were not compatible with each other, could only address specific devices, providing a degraded user experience and giving headaches to engineers and content providers. Given these facts, MPEG decided to issue a specification for a new standard that will solve the weaknesses and provide an open and standardized HTTP Adaptive Streaming Solution. MPEG DASH standardized delivery, video and audio codecs, DRM and metadata structure and with the support of commercial and non-profit organizations is trying to replace proprietary technologies in the market. It is still unclear whether DASH will fulfill its destiny, but technologies that it supports like Common Encryption providing interoperable DRM make already a deep impact in the market. Moreover, with the wide adoption of HbbTV in Europe, the future of DASH looks very promising.

Περίληψη

Η μετάδοση οπτικοακουστικού υλικού μέσω του Internet έχει γίνει συνήθης ενώ η παγκόσμια κίνηση στα IP δίκτυα που αφορά οπτικουακουστικό υλικό αναμένεται να κυριαρχήσει μέχρι το 2018. Οι καταναλωτές αναμένουν το οπτικοακουστικό υλικό που λαμβάνουν στις συνδεδεμένες στο Internet συσκευές τους να είναι υψηλής ποιότητας ανεξαρτήτως δικτύου στο οποίο είναι συνδεδεμένοι. Επιπλέον, αναμένουν υψηλής διαθεσιμότητας υπηρεσίες και απρόσκοπτη εμπειρία χρήσης. Η μετάδοση οπτικοακουστικού υλικού μέσω δικτύων IP ξεκίνησε από τα μέσα της δεκαετίας του ’90 μέσω παραδοσιακών τεχνικών μετάδοσης όπως RTP/RTSP και πιο πρόσφατα το Progressive Download, οι οποίες όμως αποτυγχάνουν να προσφέρουν το επιθυμητό αποτέλεσμα με τα σημερινά δεδομένα. Το οικοσύστημα έχει εξελιχθεί κυρίως λόγω της εισαγωγής στην αγορά των smartphones και συσκευών με άμεση σύνδεση στο Internet αλλά και λόγω της αύξησης της ταχύτητας και της αξιοπιστίας των κινητών και ευρυζωνικών συνδέσεων. Η εισαγωγή των τεχνολογιών HTTP Adaptive Streaming από εταιρίες-ηγέτες της αγοράς όπως η Apple, η Microsoft και η Adobe έφερε λύσεις, αλλά ταυτόχρονα δημιούργησε και προβλήματα λόγω εγγενών αδυναμιών. Οι τεχνολογίες δεν είναι συμβατές μεταξύ τους και απαιτείται η χρήση συγκεκριμένης τεχνολογίας για συγκεκριμένες συσκευές, με αποτέλεσμα την παροχή υποβαθμισμένων υπηρεσιών προς τους καταναλωτές και τη δυσκολία των παρόχων υπηρεσιών να προσφέρουν μια ολοκληρωμένη λύση. Με αυτά τα δεδομένα το MPEG αποφάσισε την έκδοση προδιαγραφών για την δημιουργία ενός προτύπου που θα λύσει τις υπάρχουσες αδυναμίες και θα παρέχει μια ανοιχτή και τυποποιημένη λύση HTTP Adaptive Streaming. Το πρότυπο αυτό ονομάστηκε MPEG DASH και τυποποίησε την παράδοση του οπτικοακουστικού περιεχομένου, τη χρήση κωδικοποίησης ήχου και εικόνας, το σύστημα DRM και τα μεταδεδομένα. Με την υποστήριξη εμπορικών και μη κερδοσκοπικών οργανώσεων προσπάθησε να αντικαταστήσει τις κλειστές και μη τυποποιημένες τεχνολογίες. Η αποδοχή του DASH δεν βρίσκεται σήμερα στα αναμενόμενα επίπεδα, αλλά τεχνολογίες που υποστηρίζονται από αυτό όπως το Common Encryption ήδη έχουν κερδίσει μεγάλο μέρος της αγοράς. Επιπλέον, με την ευρεία υιοθέτηση του προτύπου HbbTV στην Ευρώπη το μέλλον του DASH δείχνει ιδιαίτερα υποσχόμενο.

1. Introduction

1.1. Preface

This dissertation, published for the MSc in Information Systems of Athens University of Economics and Business, examines the ecosystem around the distribution of video content over IP networks. It mainly elaborates on the streaming technologies currently in use, the emerging ones and their advantages and disadvantages, trying to enlighten readers in the details of these technologies and give guidance on the use of them.

1.2. Goals

The goal of this dissertation is to provide details on the structure and the implementation of the streaming technologies. Based on that and the features provided by these technologies, it suggests the use of open, non-proprietary standards for video delivery. These standards can push media industry in a path that facilitates both consumers and content providers.

1.3. Structure

In the *Introduction* chapter, a general introduction of the dissertation and the goals are given.

The *Overview-Current State* chapter elaborates on the digital video over IP network current state, the key factors that drove the growth of it and the technical challenges that need to be addressed.

In chapter *Content Delivery Methods* all current streaming technologies are described and we emphasize on the features, their advantages and disadvantages. Traditional IP Streaming, Progressive Download and HTTP Adaptive Streaming technologies are examined and given their weaknesses we examine the open standard MPEG-DASH and the standards supported by it like CENC and EME.

Finally, in chapter *HbbTV*, a new initiative that harmonizes the broadcast and broadband environment and uses DASH is examined.

In chapter *Conclusion* we summarize the details given in the previous chapters and propose the use of open standards.

2. Overview – Current State

Consumption of digital video over IP networks has become the global standard for video viewing and evolved to a mainstream activity for consumers of all ages and geographies. Overall, 90 percent of consumers globally watch video content over the Internet. Video-on-demand services will account for 77% of daily consumption by the year 2020 [1], while BusinessInsider.com reports that, over the last two years, the U.S. mobile video audience increased by 77% [2]. Taking into account the above, delivery and availability of any media, on any device, over any available network connection at the same high quality expected from traditional television service, are now essential in order to reach and engage audiences. Two key reasons driving the increasing adoption of watching video content over the Internet is Multi-Screen Delivery and TV Anywhere model.

2.1. Key factors that drove Multi-Screen Delivery and TV Anywhere Model

- Explosion in penetration of Internet-connected smart devices (Smartphones, Tablets, Smart TVs, Game Consoles, Internet-enabled set-top boxes). The majority of households now own High-Definition Television sets, Internet-connected computers and smartphones, giving consumers more choices on how and when they can access video content. BusinessInsider research, shown in Figure 1, indicates the explosion in device penetration during the last decade.
- Advancements in core and last mile network bandwidth (ADSL, VDSL, FTH) as well as mobile networks (4G/LTE), during the last decade made HD video delivery everywhere possible.
- Changes in consumer habits and specifically in the way and time of accessing video content. Users want to access live or on-demand content during commuting to/from work or have instant access to a TV series without having to stick to TV stations schedule. Linear TV is outdated.
- Mobile Application development has given content providers the ability to easily monetize their content and offer better user experience to consumers.

2.2. Challenges

- Fragmentation of Internet-Connected Devices and Operating Systems. Compatibility with all the devices, operating systems and even different versions of operating systems is difficult to be obtained.
- Mobile and Home Broadband bandwidth coverage. May regions still suffer from low speed, unreliable and congested networks.
- Provide seamless user experience on mobile devices, tablets and Smart TVs.
- Bandwidth needs grow as High-Definition content becomes mainstream, with Ultra High-Definition already in the corner. As an example, the Netflix Streaming service accounts for a whopping 35% of all downstream traffic during peak periods on North America [3].

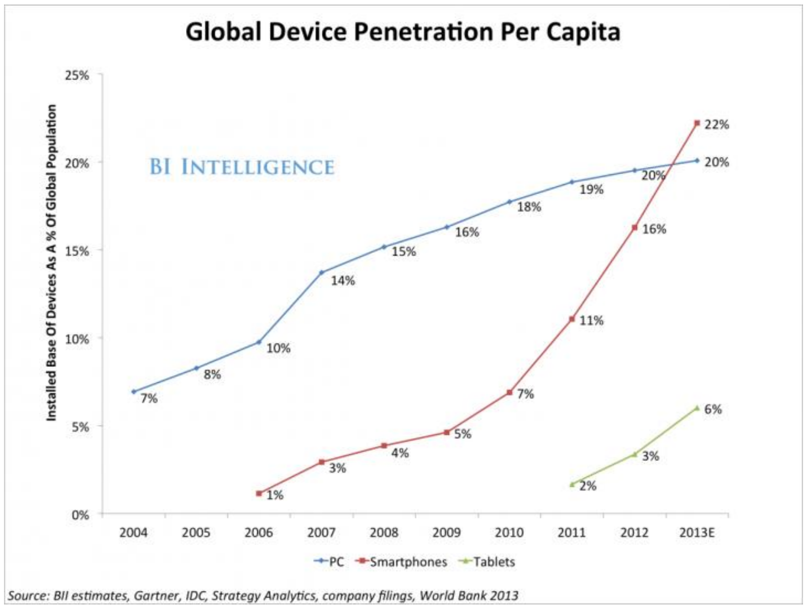


Figure 1. Global device penetration per capita

3. Content Delivery Methods

3.1. Traditional IP Streaming

In traditional IP streaming methods, a video server is delivering video to a client at a fixed bandwidth, typically over UDP, a connectionless protocol. The client and server must have synchronized states and agree on when video is stopped, playing or paused. Traditional streaming can adapt to changes in network bandwidth, using complex synchronization schemes between the server and the client.

RTSP (Real-Time Streaming Protocol) is an example of a traditional streaming protocol still in use today in many applications. It was developed by Real Networks, Netscape and Columbia University, was standardized by MMUSIC WG of the IETF and published as RFC 2326 in 1998. It is a stateful, application layer protocol, allowing the server to keep track of the client's state. The client is issuing control sequence commands, as defined in RTSP specification, that can control multimedia playback such as *PLAY*, *PAUSE* or TEARDOWN to terminate the streaming session [4].

The actual transmission of the streaming data is not a task of the RTSP protocol. RTP (Real-time Transport Protocol) and Real-time Control Protocol (RTCP) are used for media stream delivery. As the bandwidth of the video delivery is fixed, the duration of the video in seconds that is carried on each RTP packet of size of 1452 bytes is dependent on the bitrate. For example, in a video stream encoded at 1Mbps, each packet carries approximately 11 milliseconds of video. RTSP supports either UDP or TCP transports.

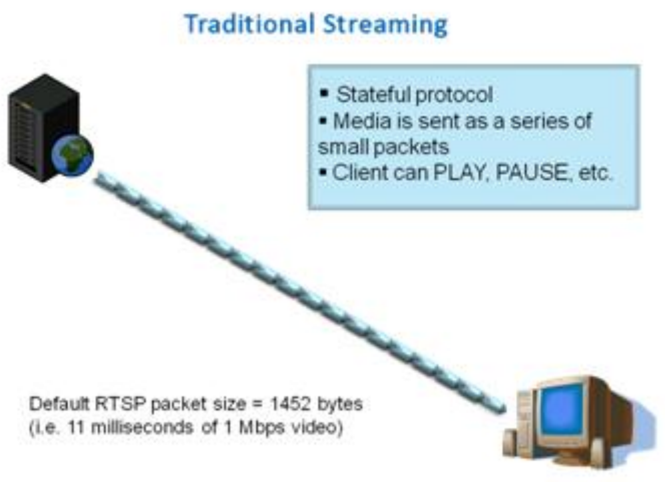

Traditional Streaming
▪ Stateful protocol
▪ Media is sent as a series of small packets
▪ Client can PLAY, PAUSE, etc.
Default RTSP packet size = 1452 bytes
(i.e. 11 milliseconds of 1 Mbps video)

Figure 2. Traditional Streaming

Two key points about traditional streaming methods are:

- The server sends the video stream to the client in real-time rate, specifically the bitrate at which the media is encoded.
- The server only sends ahead enough data packets to fill the client buffer, which is typically between 1 and 10 seconds. In case the client push the pause button in the player only some seconds of content, typically half of the buffer, is downloaded on the user side.

3.1.1. Advantages of RTSP-RTP Streaming

- Fast start of streaming session. Reliable and fast internet connection typically means good user experience.
- User can move to any part in a video, making it well suited for interactive applications.
- Saves bandwidth.

3.1.2. Disadvantages of RTSP-RTP Streaming

- Very sensitive on the internet connection. Requires speed and reliability.
- RTP, typically using the UDP protocol, has issues routing through firewalls, routers and NAT.
- Not suitable for mobile delivery.

- Lacks native support in modern web browsers.
- Requires persistent connection between client and server which leads to increased implementation cost and limited scalability.

3.2. Progressive Download

Progressive Download is a content delivery method which actually involves a file download from an HTTP Web Server. The key difference between streaming media and progressive download is how the digital media is received and stored by the end user while the end user experience remains the same.

In progressive download delivery method the video file is downloaded and stored in the end user's device, typically in a web browser cache. In that case even if the user hit the pause button at the beginning of the playback, the entire video will be downloaded, allowing for a better and smoother video playback. On the negative side, if the user doesn't like the video the whole file will be downloaded in its device consuming bandwidth.

Progressive download have been used in the past on popular video sharing web sites like Youtube, Vimeo and MySpace as it allowed for fast start of the video which was crucial for the end users.

3.2.1. Advantages of Progressive Download Streaming

- Video content is available to play immediately after downloading starts. No need to wait for video to be completely buffered in the client.
- Video can be saved in users' computers giving them the ability to have offline access and replay multiple times without bandwidth waste.
- TCP protocol used by Progressive Download Streaming is highly reliable.

3.2.2. Disadvantages of Progressive Download Streaming

- User cannot seek video until the specific part gets downloaded.
- As video is downloaded to user's computer there is no security. Video can be copied and redistributed without permission of content provider.
- Live Streaming is not possible.

3.3. HTTP Adaptive Streaming

Both content delivery methods described previously inherit structural problems mostly due to the protocols used in their implementation and complex structures used to control and maintain streaming sessions. On top of that, networks of high bandwidth, continuous availability and reliability, Internet-connected devices and High/Ultra-Definition content require new solutions adapted to their needs. HTTP Adaptive Streaming technologies enable the optimum streaming video viewing experience for a diverse range of devices over a broad set of connection speeds.

3.3.1. Implementation

In adaptive HTTP streaming the input source video, file or live stream, is encoded into file segments – also referred as 'chunks' or 'segments' – using predefined encoding formats typically consisting of streaming container, video codec, audio codec, encryption protocol and other parameters. Chunks are typically two to ten seconds longs depending of the implementation of encoder/transcoder and content publisher needs. The stream is broken into segments at video GOP (Group of Pictures) boundaries that begin with an IDR frame (a frame that can be independently decoded without dependencies on other frames). This allows each segment to be decoded independently from other segments.

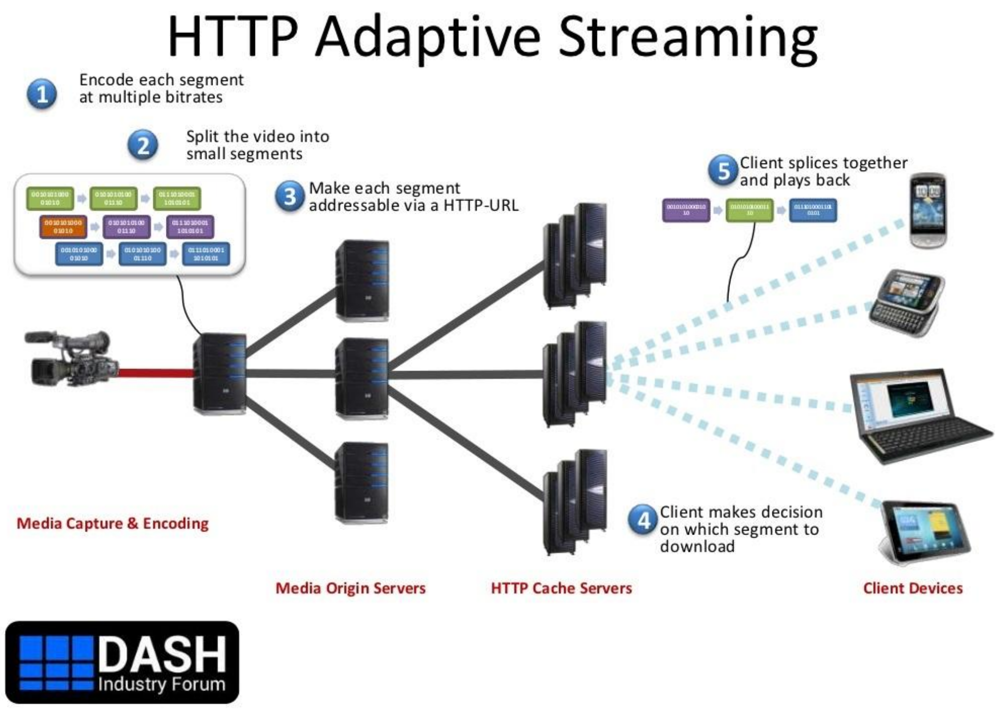

HTTP Adaptive Streaming
1 Encode each segment at multiple bitrates
2 Split the video into small segments
3 Make each segment addressable via a HTTP-URL
4 Client makes decision on which segment to download
5 Client splices together and plays back
Media Capture & Encoding
Media Origin Servers
HTTP Cache Servers
Client Devices
DASH Industry Forum

Figure 3. HTTP Adaptive Streaming

The segments produced are hosted on regular HTTP servers. Each sequence of segments is called a profile. Profiles typically differ in bitrate, resolution or audio/video codec used. A client plays the stream by requesting segments of a profile from a web server and downloading them via regular HTTP. As the segments are downloaded, the client plays back the segments in the linear order requested. Since video segments are sliced along GOP boundaries with no gaps between them, video playback is seamless – even if in reality it is just a collection of independent file downloads via a sequence of HTTP GET requests.

Adaptive delivery of content is supported by encoding the source video at multiple bitrates and resolutions. Adaptive delivery enables a client to 'adapt' to network conditions by automatically downloading video segments from different encoding profiles. The client during the initialization of the connection receives a file containing the list of available profiles called a manifest or a playlist which is typically an XML file with predefined structure. Delivery of segments starts from the lower profile and depending on several criteria the client decides which profile will be requested next. By computing the available network bandwidth, the client determines if it must continue with the download of a lower bitrate/resolution profile or of a higher one. Bandwidth calculation on the client side is repeated at every chunk download so the client can adapt to network conditions every few seconds. Other conditions that may affect the client’s choice of profile are CPU utilization, dropped frames or client’s ability to play back a specific codec or resolution e.g. Apple iPhone 5 resolution is 1136x640 while iPhone’s 6 is 1920x1080 so if the target device is an Apple iPhone 5 there is no point in downloading segments from a profile supporting 1920x1080 resolution.

The delivery method described above works for both live input and file-based sources. The manifest file downloaded from the client, lists the bitrates associated with the available profiles and the client uses the appropriate URL to download the chunks for the specific profiles. In the case of file-based workflows the manifest contains information for every chunk in the content while in the case of the live streaming a ‘rolling window’ manifest file containing references to the latest available chunks is delivered. That means that the client is downloading an updated manifest file repeatedly containing information for new chunks available.

3.3.2. Topology

The components used in an HTTP Adaptive scheme consist of an encoder/transcoder, a packager, the Content Delivery Network and the client application.

3.3.2.1. Encoder/Transcoder

The encoder/transcoder is typically integrated into a single software or hardware solution. Commercial and open-source solutions can be used to provide this functionality. The transcoder is used to re-encode video files in the desired profiles while the encoder is used to encode live video/audio input.

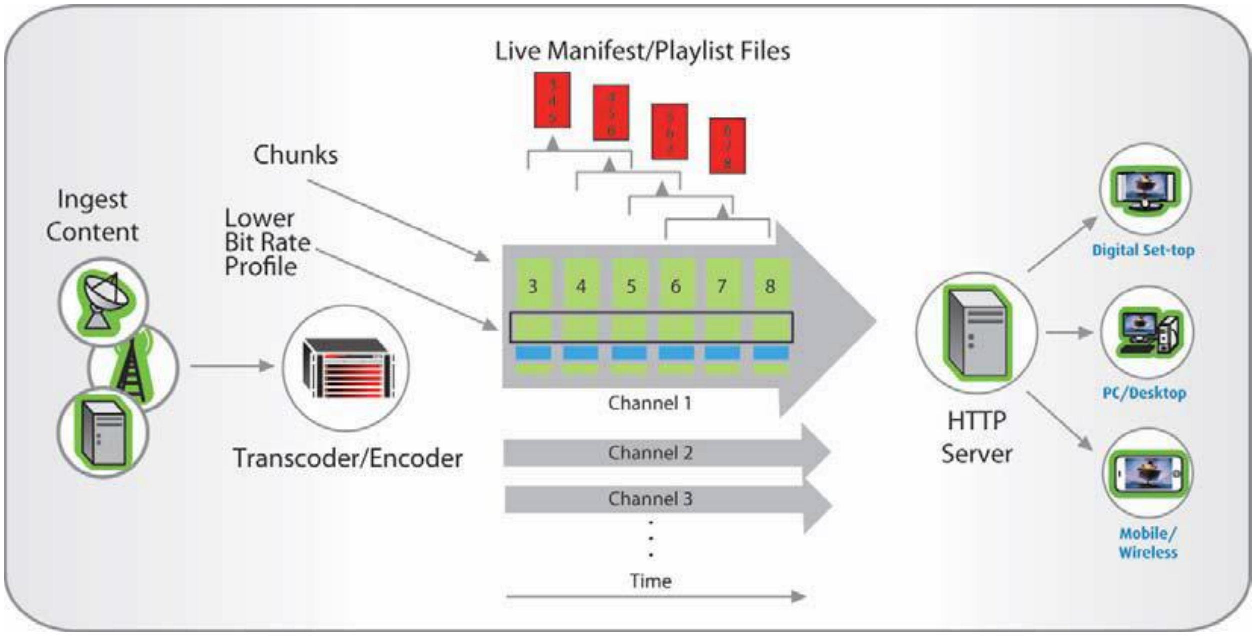


Figure 4. Content Delivery Chain for Live Adaptive Streaming

The encoder/transcoder component provides the following functionality:

- Scale the output video to the resolutions configured on the profiles e.g. from 720x576 to 320x180, 640x360 or even upscale to 960x540.
- Encode/transcode video in the bitrates configured on the profiles e.g. from 300Kbit/s to 4Mbit/s in case of High Definition content.
- De-interlance input video to produce progressive formatted outputs.
- Encode/transcode audio. All commercial implementations of HTTP Adaptive Streaming use AAC or AAC+ as the standard audio codec.
- All output profiles must be aligned based on the IDR and GOP structure to ensure that the playback of the chunks during profile change remains continuous and smooth.

One of the key characteristics of the ‘adaptive’ nature of HTTP streaming is the number of different profiles created by the transcoder/encoder. The processing power of the encoder should be such that it can sustain a large number of different output profiles, to support more end user devices and deliver better user experience. Stacking more than one encoder to produce the desired number of profiles is not the optimal solution as in that case the alignment of the different profiles can be a headache and finally result in a non smooth playback. The number of profiles created depends on the following criteria:

- Resolution of the input source video. Typically video content produced in HD is expected to be delivered in HD.
- Target devices. With the launch of smartphones and tablets with HD or 2K resolution screens users expect delivery of video content of the best possible resolution and bitrate.
- Network quality. Typically unreliable networks require more encoding profiles to fill the gaps and deliver a smoother end user experience.
- CDN costs. High bitrate encoding profiles have the downside of high CDN costs charged per bandwidth used.

Based on the above criteria we present an example of the profiles used in the HTTP Adaptive Streaming Solution of a Greek Broadcaster. The source video resolution is 1920x1080 but in this case the content provider decided to make a compromise between high CDN costs and quality. Furthermore, in the demo streams they operate, the majority of their clients' devices are still not equipped with HD screens or the hardware is still not capable to properly decode high bitrate HD H.264 streams.

Resolution	Video Bitrate	Audio Bitrate
320x180	300Kbps	48Kbps
640x360	800Kbps	48Kbps
848x480	1200Kbps	48Kbps
960x540	2000Kbps	48Kbps
1280x720	3000Kbps	48Kbps

Table 1. Profiles used in a Greek Broadcaster HTTP Adaptive Streaming Solution

3.3.2.2. Packager

The packager is the component that takes as input the output of the encoder and packages the video for a specific delivery protocol. The packager is either built into a single appliance along with encoder/transcoder or installed as a standalone service on the edge of the network. In the latter case, bandwidth is saved as the encoded profiles are first sent to the network edge and then packetized to the desired delivery protocol for the end user device. Lately, packaging services are also offered by CDNs. The packager typically has the following features:

- Support for all current delivery protocols such as Apple HLS, Microsoft Smooth Streaming, Adobe HDS and MPEG-DASH.

- Integration with many CDNs.
- Encryption capabilities.
- Integration with third party key management solutions.
- Ability to packetize live streams or files.
- Support multiple ways of delivering the chunks – typically HTTP PULL or HTTP POST.

3.3.2.3. Content Delivery Network

Contrary to the Traditional IP Streaming solutions, the CDN needed for HTTP Streaming is not specialized, as the packets transferred are based on the HTTP protocol. The CDN should have the following features:

- Easy to grow in capacity and scale to meet the challenges of larger audiences.
- Provide monitoring mechanisms by keeping track of the traffic, visitors and devices used, giving insights to content providers.
- As the number of chunks can be very large, the CDN must be able to properly deliver and sustain large number of files.
- A mechanism to time out the chunks no longer valid for live content, should be built in.

Currently OTT (Over-the-top) dominates as the model used for streaming services, but there are still operators using closed, managed networks to deliver video content mainly based in IPTV business models.

3.3.2.4. Client

Typically clients are software either natively supported in an OS or delivered as an installation package. Major categories of clients are:

- Native in the web browser. The Apple HLS implementation is supported natively in Safari Web Browser in desktop, tablets and mobiles. MPEG-DASH supports the Media Source Extensions of W3C, allowing JavaScript to generate media streams for playback supported by the majority of web browsers.
- In web browsers using plug-ins. Adobe HDS and Microsoft Smooth Streaming implementations require installation of plug-ins, Flash and Silverlight respectively, to playback video in the web browser.

- Mobile applications typically developed from Broadcasters or Content Providers deliver a more consistent user experience and offer Live, Catch-up and Video-on-Demand capabilities. An example of such application is BBC iPlayer offered for Android, iOS and Windows desktop and mobile platforms. This category of client has the added value of providing seamless user experience across different devices.
- Desktop Applications for Windows and OS X. Using technologies like Silverlight or Adobe Air.
- Applications for Smart TVs supported by companies like Samsung, LG and Panasonic.

The challenge for Broadcasters and software engineers, involved in developing video streaming applications, is to deliver the same user experience for their content in as many as possible of the clients listed above. Initiatives for specifying guidelines have been started with help of commercial and non-profit organizations like MPEG and SMPTE.

3.3.3. Advantages of HTTP Adaptive Streaming

HTTP Adaptive Streaming is already the industry standard for online video delivery. Some advantages of the method and its implementations are:

- Lower costs for content providers since HTTP Adaptive Streaming is handled and offered by CDNs as web download services and not as special media streaming services.
- Lower infrastructure costs as generic HTTP caches/proxies can be used for delivery.
- Client adapts to the content dynamically and automatically without requiring special mechanisms inside the CDN network or on the server.
- Media protocols, typically based on UDP, have issues routing through firewalls, routers and NAT. HTTP Adaptive Streaming, based on TCP, doesn’t.
- Fast start-up and seek times as playback can be initiated on the lowest bitrate and move up to higher bitrates.
- Smooth playback experience as long initial buffer time, disconnects and playback start/stop are eliminated.
- Easier to scale and maintain infrastructure.

3.3.4. Disadvantages of HTTP Adaptive Streaming

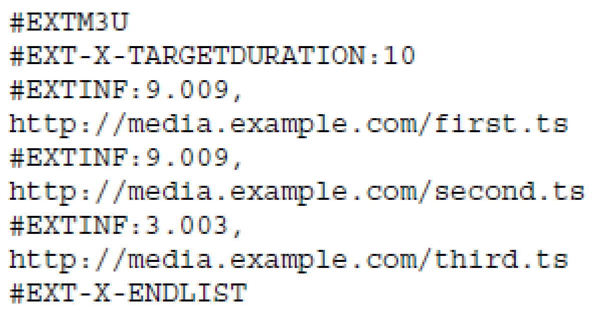
HTTP Adaptive Streaming Method also has some disadvantages:

- The clients must buffer a few chunks to make sure they don’t starve their input buffers, increasing the end-to-end latency of live streams.
- HTTP is based on TCP, providing low packet loss and easy recovery. However, when packet loss rises, TCP can fail completely. Thus, clients will typically have good quality playback or no playback.
- Most implementations start with low bitrate profiles and then, if possible, move to a higher bitrate one. This means that the user experiences a lower than expected video quality at the start of the stream.

3.3.5. Apple HTTP Live Streaming (HLS)

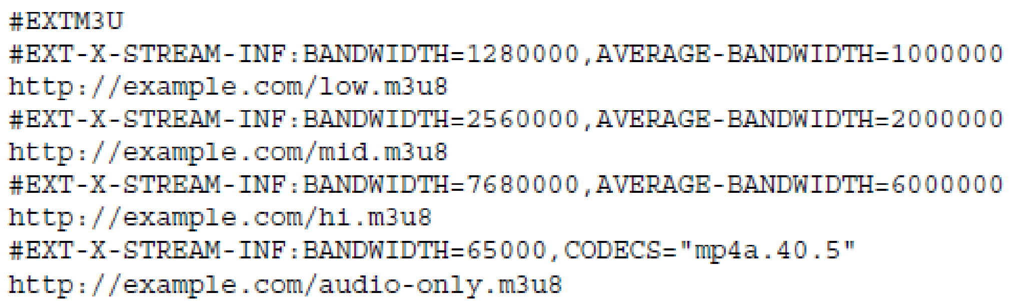
Apple introduced HTTP Live Streaming (HLS) with iOS 3.0 in 2009, as a built-in feature of Safari and later implemented it in QuickTime, iOS and OS X to provide a reliable, cost-effective means of delivering continuous video over the Internet. It works by segmenting the input stream into a sequence of small files that can be downloaded by end user device and produce a smooth video stream playback. It also provides a framework for basic media encryption and the ability to offer multiple versions of the same content that can support schemes such as multiple audio translations.

Segments are pushed on HTTP servers along with the playlist files. HLS defines two types of playlist files: Media Playlist and Master Playlist. The Media Playlist file is a list of URLs that point to chunks that should be played sequentially. The Master Playlist file points to a set of different Media Playlist files, one for each output profile, called Variant Streams. A Master Playlist can also specify a set of alternate versions of the content, typically audio in different languages or different camera angles. Figure 5 and Figure 6 show sample HLS Media and Master Playlists.



```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXTINF:9.009,
http://media.example.com/first.ts
#EXTINF:9.009,
http://media.example.com/second.ts
#EXTINF:3.003,
http://media.example.com/third.ts
#EXT-X-ENDLIST
```

Figure 5. Media Playlist



```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=1280000,AVERAGE-BANDWIDTH=1000000
http://example.com/low.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2560000,AVERAGE-BANDWIDTH=2000000
http://example.com/mid.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=7680000,AVERAGE-BANDWIDTH=6000000
http://example.com/hi.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5"
http://example.com/audio-only.m3u8
```

Figure 6. Master Playlist

A Media Playlist contains a series of Media Segments, each specified by a URI. A client can download a segment by requesting the segment by the specific URI. The HLS specification [5] indicates that each Media Segment must be formatted in MPEG-2 TS and each TS segment must contain a single MPEG-2 Program. A Media Segment should have at least one key frame, not necessarily at the beginning of the GOP, and maintain the continuity of the provided stream.

Playlist Files must be encoded in UTF-8. Lines beginning with #EXT are tags which specify parameters and metadata of the playlist or information about the Media Segments or Media Playlists. In the following we will explain the usage of common tags in playlists.

- *EXTM3U* – This tag must be the first line of every playlist, indicating that the file is an Extended M3U Playlist File. Format is #EXTM3U
- *EXTINF* – Specifies the duration of a Media Segment. It is required for each Media Segment and applies only to the next Media Segment line appearing in the playlist. Its format is #EXTINF: <duration>,<title> where duration is a decimal-integer or decimal-

floating-point that specifies the duration in seconds. This tag can only appear in a Media Playlist.

- *EXT-X-KEY* – HLS supports encryption of Media Segments and this tag specifies how to decrypt them. Decryption specified is valid until a new EXT-X-KEY tag appears in the playlist file. Its format is #EXT-X-KEY:<attribute-list>. The most important attribute is METHOD whose value can be one of NONE, AES-128 and SAMPLE-AES. NONE means that Media Segments are not encrypted, AES-128 means that Media Segments are encrypted using the Advanced Encryption Standard with 128-bit key, Cipher Block Chaining and PKCS7 padding. SAMPLE-AES means that the Media Segments contain media samples that are encrypted using AES-128. Other attributes like URI, IV, KEYFORMAT and KEYFORMATVERSIONS facilitate the encryption process.
- *EXT*-X-TARGETDURATION – This tag specifies the maximum Media Segment duration. Duration specified in EXTINF tags in the playlist file must be less than or equal to the target duration as tag applies to the entire Playlist file. Its format is: #EXT-X-TARGETDURATION:<s>, where s is a decimal-integer and duration is counted in seconds. This tag can only appear in a Media Playlist.
- *EXT*-X-MEDIA-SEQUENCE – This tag indicates the Media Sequence Number of the first Media Segment that appears in a Playlist file. Its format is: #EXT-X-MEDIA-SEQUENCE:<number> where number is a decimal-integer. This tag can only appear in a Media Playlist.
- *EXT*-X-DISCONTINUITY-SEQUENCE – This tag allows synchronization between different versions of the same Variant Stream or different Variant Streams that have EXT-X-DISCONTINUITY tags in their Media Playlist. Its format is #EXT-X-DISCONTINUITY-SEQUENCE:<number> where number is a decimal-integer.
- *EXT*-X-ENDLIST – This tag indicates that no more Media Segments will be added to the Media Playlist File. Its format is #EXT-X-ENDLIST.
- *EXT*-X-MEDIA – Tag is used to relate Media Playlists that contain alternative Renditions of the same content. For example a Media Playlist that contains English, French and Greek audio should contain three EXT-X-MEDIA tags. Its format is: EXT-X-MEDIA: <attribute-list>. Valid attributes are TYPE, URI, GROUP-ID, LANGUAGE, ASSOC-LANGUAGE, NAME, DEFAULT, AUTOSELECT, FORCED, INSTREAM-ID,

CHARACTERISTICS. TYPE attribute can take values of AUDIO, VIDEO, SUBTITLES and CLOSED-CAPTIONS. GROUP-ID is a required tag that specifies the group to which the Rendition belongs to and its value is a quoted string. A set of EXT-X-MEDIA tags with the same GROUP-ID value and the same TYPE value forms a group of Renditions.

- EXT-X-STREAM-INF – This tag specifies a Variant Stream, basically a set of Renditions that can be combined to play the presentation. The URI following the tag line is marked as a Rendition of the Variant Stream. Its format is:
  #EXT-X-STREAM-INF:<attribute-list>
  <URI>
  Valid attributes are BANDWIDTH in bits per second, RESOLUTION describing the optimal pixel resolution, AUDIO, VIDEO, SUBTITLES and CLOSED-CAPTIONS. When an EXT-X-STREAM-INF tag contains an AUDIO, VIDEO, SUBTITLES or CLOSED-CAPTIONS attribute, it indicates that alternative versions of the content are available. The playlist file created must meet specific requirements in values of BANDWIDTH, RESOLUTION and Variant Stream.

In HLS, a playlist file corresponding to a live stream must be repeatedly downloaded by the client so that the client is updated with the most recently available chunks. The server cannot change the Media Playlist file with the exception of appending lines to it, removing Media Segments in a FIFO way, increment the value of EXT-X-MEDIA-SEQUENCE or EXT-X-DISCONTINUITY-SEQUENCE tags and add an EXT-X-ENDLIST tag to the playlist. The playlist is downloaded every time a chunk is played, so in order to minimize the number of requests, Apple recommends chunk duration of 10 seconds but this can be adjusted according to content provider needs. The size of the playlist files that is small compared to the size of the video segments and the fact that the client maintains an open TCP connection to the server makes the added network load for playlist download not significant. Shorter chunk durations can be used that will allow the client to adapt to network conditions more quickly.

3.3.5.1. HLS Advantages

- Simple protocol, easy to monitor and enhance. Playlists can be easily modified to provide functionality such as re-broadcast and ad-insertion.

- TS files are well-established and can be easily tested and verified for conformance.
- TS files can carry metadata such as SCTE-35 and ID3 tags, providing ad-insertion and timed metadata capabilities respectively.
- HLS is native to iOS devices giving easy access to millions of devices and users.

3.3.5.2. HLS Disadvantages

- HLS is not supported natively in Windows OS (mobile phones and desktops) and is partially supported in Android platform (Android 4.0+).
- TS files mux audio, video and other streams in a single file. This means that when a client requests a stream with multi-language support, all audio streams will be sent to the client effectively consuming more bandwidth.
- Management of HLS chunks files is not convenient as there is no mechanism of wrapping chunk files together in a format. For example the number of files produced for one channel in one day with 5 profiles of 10-second chunk duration can be more than 50,000.

3.3.6. IIS Smooth Streaming (MSS)

IIS Smooth Streaming, Microsoft's implementation of HTTP-based adaptive streaming, is an extension of IIS, first introduced in October 2008 as a feature of IIS version 7.0. Smooth Streaming uses the MPEG-4 Part 14 (ISO/IEC 14496-12) as its format both for storing content on disk and delivering content to clients. Each chunk/GOP is defined as a MPEG-4 Movie Fragment and is stored in a single contiguous MP4 file. For each encoding profile specified in the encoder/transcoder a single MP4 file is created. When a client requests a specific time segment or simply starts a streaming, the server virtually and dynamically creates file chunks and sends them to the client. The two parts of the Smooth Streaming format is the wire format and the disk file format. The wire format defines the structure of the chunks that are sent by IIS to the client, whereas the file format defines the structure of the contiguous file on disk. Storing a single contiguous file on disk while virtually creating and sending segments on client requests, gives the advantage of better file management. The wire format is a direct subset of the file format, as the MP4 specification allows MP4 to be internally organized as a series of fragments [6].

3.3.6.1. *Disk File Format*

The basic unit of an MP4 file is called a "box" and typically contains both data and metadata. For supporting live streaming scenarios and shorter startup times, the specification allows MP4 boxes to be organized in a fragmented manner, where the files can be written "ad-hoc" as a series of short metadata/data pairs and not as a single long file. Smooth Streaming files are often referred to as "Fragmented MP4 Files".

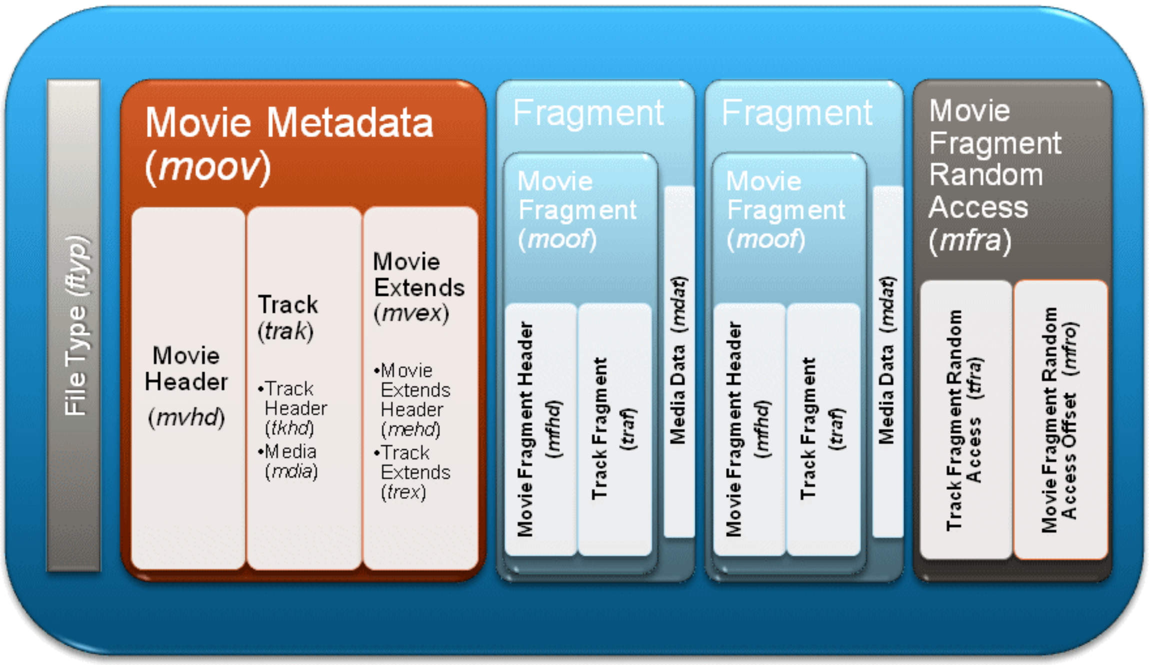


Figure 7. Smooth Streaming File Format

The MP4 file structure starts with a Movie Metadata box – *moov* - containing general metadata about the file. The actual content is contained in the Fragment boxes that carry more accurate metadata for the specific fragment – *moof* - and media data - *mdat*. The file closes with a Movie Fragment Random Access box – *mfra* – that allows accurate seeking within the file.

3.3.6.2. *Wire File Format*

As already described when a client requests a time-based video, IIS Web Server seeks to the appropriate starting fragment in the MP4 file using the mfra box, then lifts the related fragments and sends the video over the 'wire' to the client. This technique involves no re-muxing or rewriting and greatly enhances the efficiency of the IIS Web Server.

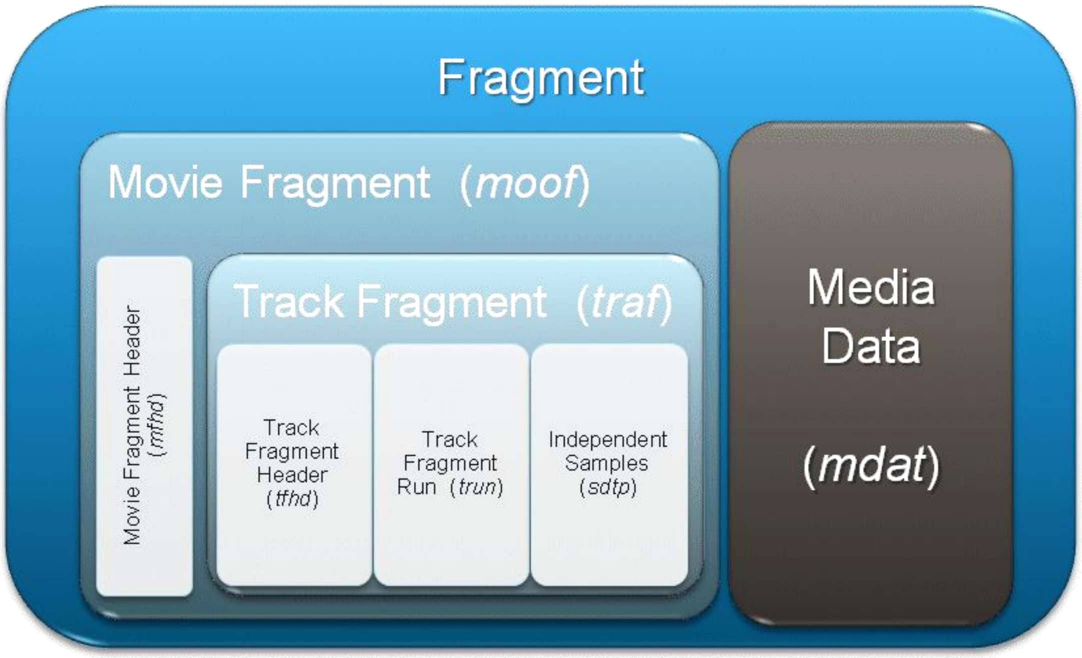

Fragment
Movie Fragment (moof)
Movie Fragment Header (mfhd)
Track Fragment (traf)
Track Fragment Header (tfhd)
Track Fragment Run (trun)
Independent Samples (sdtp)
Media Data
(mdat)

Figure 8. Smooth Streaming Wire Format

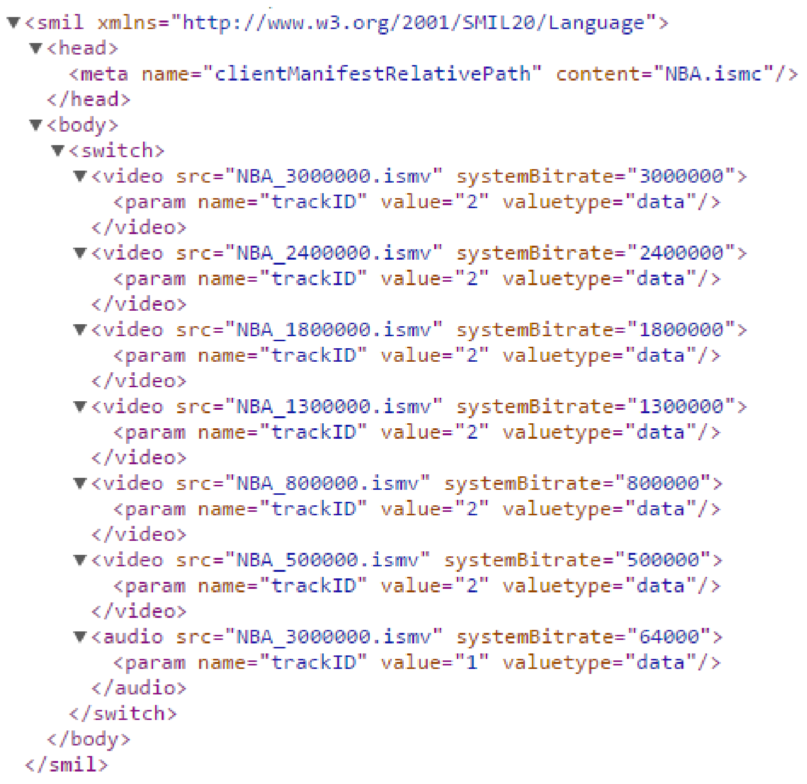
3.3.6.3. *Media Assets*

Typically Smooth Streaming Media Assets consist of the following files:

- MP4 files containing video/audio
  - Files with **ismv** extension contain video and audio or only video. One ISMV file is required per encoded video bitrate.
  - Files with **isma** extension contain audio only.

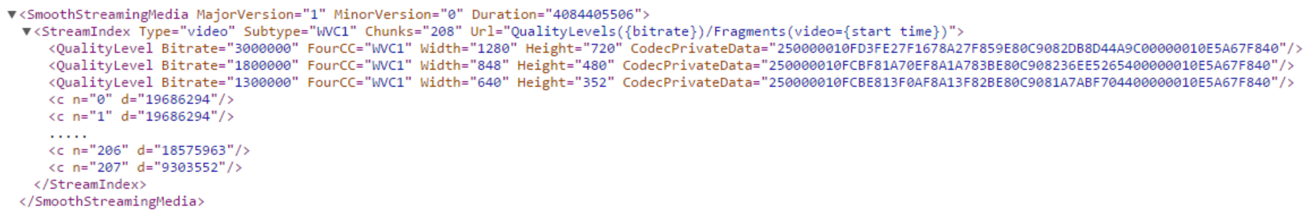
3.3.6.4. *Manifest Files*

- Files with **ism** extension are Server Manifest Files and describe the relationships between the media tracks, bit rates and files on disk.
- Files with **ismc** extension are Client Manifest Files and describe the available streams to the client e.g. codecs used, bitrates encoded, video resolutions etc.



```
▼<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  ▼<head>
     <meta name="clientManifestRelativePath" content="NBA.ismc"/>
   </head>
  ▼<body>
    ▼<switch>
      ▼<video src="NBA_3000000.ismv" systemBitrate="3000000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<video src="NBA_2400000.ismv" systemBitrate="2400000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<video src="NBA_1800000.ismv" systemBitrate="1800000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<video src="NBA_1300000.ismv" systemBitrate="1300000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<video src="NBA_800000.ismv" systemBitrate="800000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<video src="NBA_500000.ismv" systemBitrate="500000">
         <param name="trackID" value="2" valuetype="data"/>
       </video>
      ▼<audio src="NBA_3000000.ismv" systemBitrate="64000">
         <param name="trackID" value="1" valuetype="data"/>
       </audio>
     </switch>
   </body>
 </smil>
```

Figure 9. Sample ISM File



```
▼<SmoothStreamingMedia MajorVersion="1" MinorVersion="0" Duration="4084405506">
  ▼<StreamIndex Type="video" Subtype="WVC1" Chunks="208" Url="QualityLevels({bitrate})/Fragments(video={start time})">
     <QualityLevel Bitrate="3000000" FourCC="WVC1" Width="1280" Height="720" CodecPrivateData="250000010FD3FE27F1678A27F859E80C9082DB8D44A9C00000010E5A67F840"/>
     <QualityLevel Bitrate="1800000" FourCC="WVC1" Width="848" Height="480" CodecPrivateData="250000010FCBF81A70EF8A1A783BE80C908236EE52654000000010E5A67F840"/>
     <QualityLevel Bitrate="1300000" FourCC="WVC1" Width="640" Height="352" CodecPrivateData="250000010FCBE813F0AF8A13F82BE80C9081A7ABF70440000010E5A67F840"/>
     <c n="0" d="19686294"/>
     <c n="1" d="19686294"/>
     .....
     <c n="206" d="18575963"/>
     <c n="207" d="9303552"/>
   </StreamIndex>
 </SmoothStreamingMedia>
```

Figure 10. Sample ISMC File

The Smooth Streaming format specification supports the following features:

- VC-1, WMA, H.264 and AAC codecs
- Text Streams
- Multi-language audio tracks
- Alternate video and audio tracks
- Multiple hardware profiles
- Live encoding and streaming

3.3.6.5. Request and Playback

When a client wants to playback a Smooth Streaming content, the first thing it requests from the IIS server is the **ismc** client manifest. Using the metadata information in the manifest the client gets the information needed for the correct playback of the content such as codec used, resolution, bit rates and available chunks.

Client can request fragments using the following restful URL structure:

http://{servername}/{PublishingPointPath}/{PublishingPointName}.isml/QualityLevels(quality-level)/Fragments(video/audio-eng)

The QualityLevel indicates the encoding profile requested and the video and audio-eng indicate the specific chunks. The Fragments part of the request is given using a time stamp that the IIS server uses to extract the correct chunk from the MP4 file.

The part of the adaptive streaming switching is entirely implemented at the client-side. Client side takes into account chunk download times, buffer fullness, rendered frame rates and decides when to request higher or lower bitrates from the server. All profiles of the same source are frame-aligned so adaptive switching is seamless.

3.3.6.6. MSS Advantages:

- IIS creates a contiguous file for each stream which makes management easier.
- Small recommended chunk size – typically 2 seconds – allows for quick adapt of clients to network conditions.
- The segregated video and audio files mean that delivery of different audio tracks involves just a change in manifest file.
- Supports multiple data tracks that can be used to store metadata about ad insertion, subtitling.
- Client can request chunks in a more developer-friendly way e.g. using timecode. This makes implementation of players-clients simpler.
- For live streaming the client doesn’t need to repeatedly download a manifest as the server computes the URLs for the chunks directly.

3.3.6.7. MSS disadvantages:

- IIS server, needed as Origin Server, adds an extra point of failure and complicates the network.
- MSS requires installation of the separate Silverlight plug-in in Windows OS.
- Support in other platforms is unofficial and minimal.

3.3.7. Adobe HTTP Dynamic Streaming (HDS)

Adobe HDS was defined after HLS and MSS and makes use of characteristics found on both of them. HDS makes use of the MP4 fragment format (F4F) for both live and on-demand streaming and Flash Media Manifest (F4M) as manifest file, containing information about the available profiles.

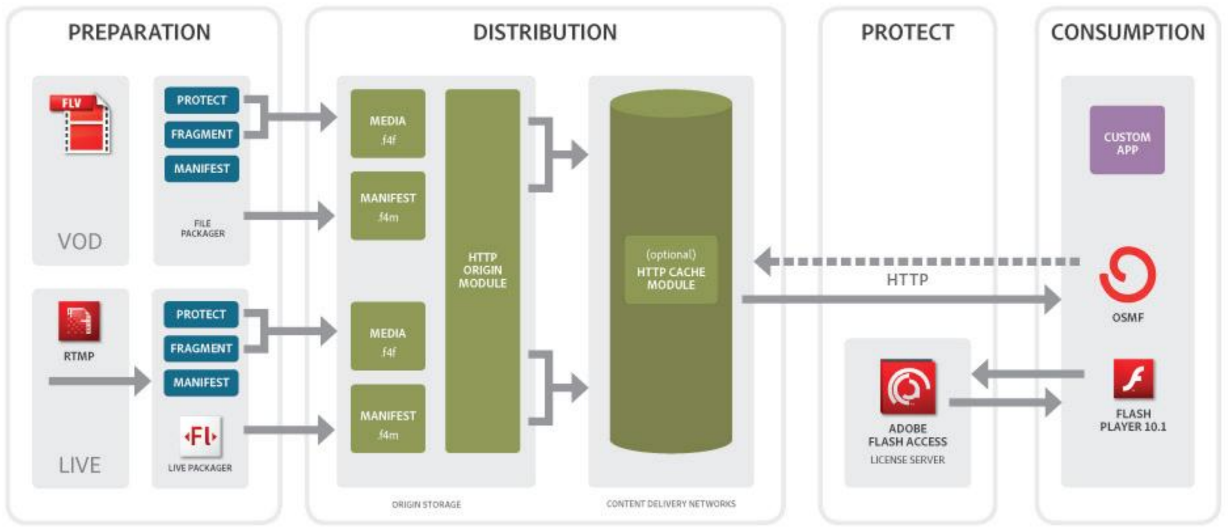


Figure 11. HTTP Dynamic Streaming delivery workflow for live and VOD

Adobe has created tools to make packaging of both VOD assets and Live Streaming easier. Packagers perform three tasks: generate MP4 fragment-compliant files, generate F4M manifest files and can optionally apply content protection on assets. For VOD assets, the packager creates F4F files taking as input pre-encoded FLV or F4V assets while for live streams the packager takes as input live RTMP streams. Codecs supported include H.264, AAC, VP6 and MP3. To ensure seamless playback during switching, keyframes need to be aligned between all bitrates of the same source content. HDS is using keyframes for fragments creation and client can only change bitrates at a keyframe. Files created are then placed on an HTTP Server that plays the role of the Origin Server and is responsible for receiving fragment requests and returning the appropriate fragment of the file. Adobe is recommending Apache for the Origin Server and provides a module to support HDS delivery on Apache. Files can then be pushed to cache servers for optimal management of the network.

Playback of HDS streams is initiated by the client and at the start of the streaming session the media player downloads the F4M manifest file that provides all the information needed to playback the media, including fragment format, available bitrates, Flash Access license server location and metadata information. After the client requests the part of the video needed, the server retrieves portions from the complete file called fragments, downloads them and they are played by the Flash Player Client. Adobe provides full support for HTTP Dynamic Streaming with OSMF (Open Source Media Framework) which simplifies requesting and parsing the manifest files, assembling the media fragments and monitoring QoS.

3.3.7.1. F4F Media Format

F4F file format groups media and metadata into three different data structures.

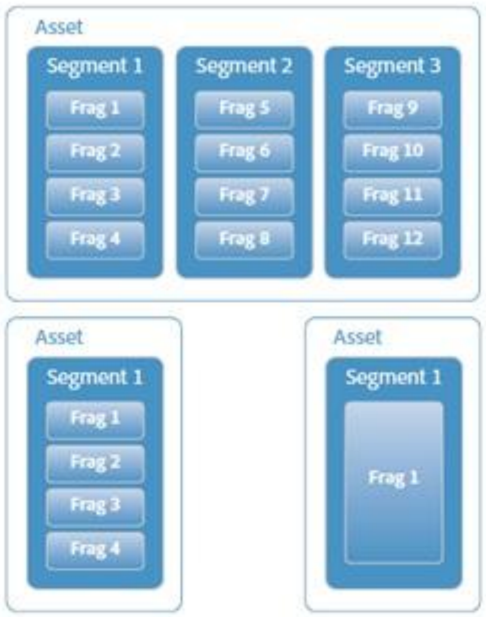


Figure 12. Segment and Fragment Structure in F4F Media Format.

Fragments contain both metadata – moof – and media data – mdat. The MP4 file format supports an efficient fragment structure that interleaves metadata inside multiple moof boxes and media data in multiple mdats in a single file. Each fragment begins with a random access point and can contain multiple seekable random access points. Segments group together a collection of fragments and act as the storage format. Both segments and fragments are addressable using a URL with the following structure.

http://server_and_path/QualityModifierSeg'segment-number'-Frag'fragment-number'

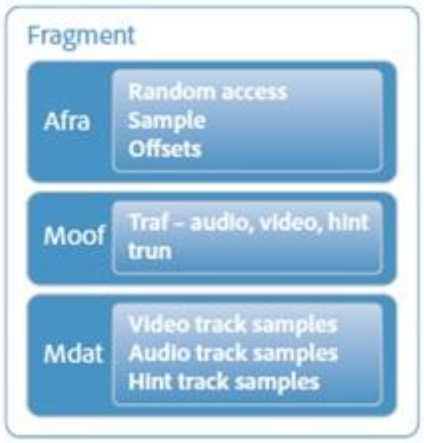


Figure 13. Fragment Structure in F4F Media

Bootstrap information is the initial information used to start consuming media. It is repeatedly downloaded by the client for updating the information of the available chunks. Bootstrap information is located in both the media file (F4F) and the manifest file (F4M) and contains a Segment Run Table containing information about the number of fragment in each segment, a Fragment Run Table containing information for the fragments duration and optionally a Metadata Table containing list of servers and DRM information [7].

3.3.7.2. HDS Advantages:

- Wide adoption of Flash on different devices and operating systems.
- HDS is part of Flash and can make use of Flash's environment.
- Use of fragmented MP4 format makes management of files easier.

3.3.7.3. HDS Disadvantages:

- Not open-source and rather closed format. Bootstrap file is a binary format and cannot be changed by user intervention.
- HDS is changing rapidly and makes deployment of stable ecosystem difficult.

3.3.8. Feature Comparison

3.3.8.1. Support of Multiple Audio Tracks

Support of multiple audio tracks is a must for most content publishers, especially in multicultural societies such as U.S. It is also becoming a necessity in other parts of the world as the content from emerging markets is reaching new audiences.

TS files, used in the HLS implementation, can carry multiple audio tracks but can only be packaged in a single container. That means that the bandwidth consumed to support multiple audio tracks is considerable and that the client downloads all audio tracks in segments even if it is not going to use them. Furthermore, HLS provides a static workflow as there is no mechanism to dynamically add or delete tracks while the stream is operational. An alternative, in order to reduce the bandwidth needed for delivering multiple audio tracks, would be to package each video/audio combination separately but that would increase the storage needs and the chunk management effort considerably.

Smooth Streaming, due to the implementation of fragmented MP4 disk and wire format makes support and delivery of multiple audio tracks easy. In addition, client can request on demand a different audio track and the server delivers it on the next chunk that will be sent to the client. The format used, makes bandwidth usage more efficient.

HDS supports multiple audio tracks with a late binding audio technique, without requiring duplication and repackaging of the video for each audio track. The term "late binding" refers to the ability to keep the audio and video tracks of a stream not muxed until the last possible moment. The best option is the binding to occur at the client player with the server keeping track of which audio track will be muxed with the video stream. This technique allows content publishers to provide multiple audio tracks for a specific content at any time, even after the asset’s initial packaging.

3.3.8.2. Digital Rights Management

Digital rights management (DRM) is one of many ways to protect valuable content from piracy and other misuse. A DRM solution typically constitutes of four elements: digital rights to manage, encryption, license management, and a DRM-enabled client.

- Digital Rights to Manage—Full DRM technologies enable a broad range of business models, including purchase, subscription, rental, gifting, playback on single and multiple platforms via streaming, downloading, or sideloading.
- Encryption—Full DRM technologies use encryption to protect the content during downloading or other transfer.
- License Management—Full DRM technologies require a server to manage the request and issuance of licenses.
- A DRM-Capable player can communicate with the license server and enforce all software and hardware-related playback restrictions. On computers and notebooks the installation and use of plug-ins such as Flash for Adobe Primetime, Silverlight for Microsoft PlayReady and a specific plug-in for Google Widevine is required. On mobile devices, most DRMs require a downloadable app created with SDKs provided by each DRM vendor, while DRM support on consumer electronics and other devices typically require some level of custom development.

HLS supports encryption of chunks using AES-128. All related information (IV, license server URI etc.) should be included in the playlist to inform the client that the chunks are encrypted and request the decryption keys. HLS does not specify a mechanism for authenticating clients making it is easy to capture the key in a browser session. This is why HLS is not considered to inherently support a full DRM solution.

Smooth Streaming uses Microsoft PlayReady as DRM solution. PlayReady is a complete DRM framework used for encryption, key management and support of DRM capable player using Silverlight plug-in. Only the payload of the MP4 file is encrypted as metadata contain valuable information for the encryption scheme. PlayReady is supported by the majority of software and hardware encoder solutions.

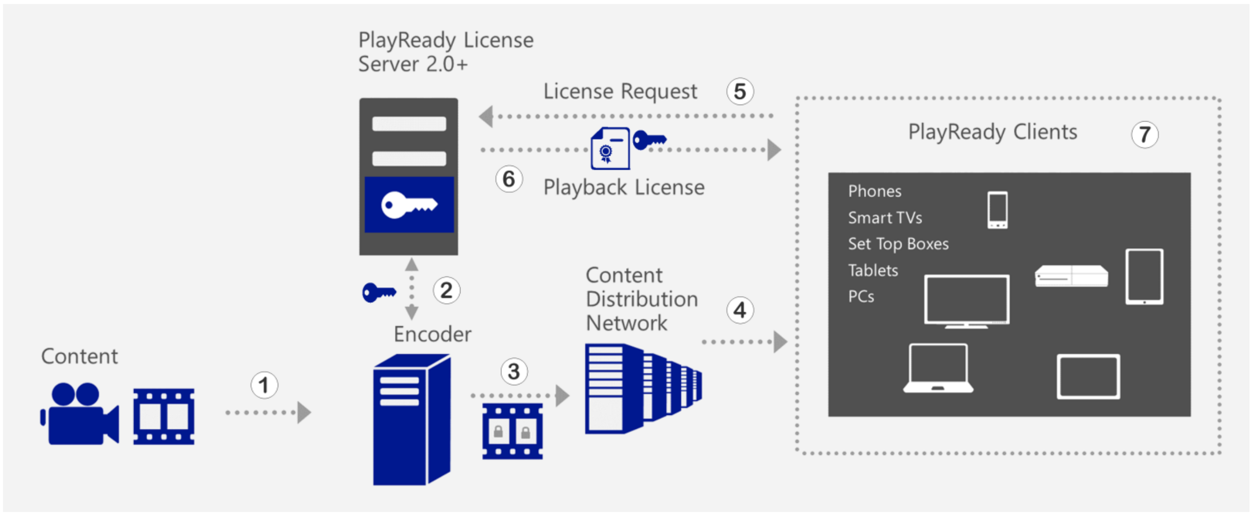


Figure 14. PlayReady DRM Scheme

HDS uses Adobe Primetime for providing DRM solutions. Adobe Primetime supports on-premises license server management or as a service. A critical difference from other DRM solutions is that no key exchange between DRM vendor and Encoder Vendor is needed. Decryption keys are encrypted and sent along with the content to the client but access is only granted at run time with no communication needed to the key management system or encoder.

3.3.8.3. Closed Captions/Subtitles

HLS supports closed captions and subtitles included in the TS chunks. Both should be referenced in the Playlist using the EXT-X-MEDIA tag.

MSS supports data tracks that hold Time Text Markup Language (TTML). Microsoft's implementation of MSS client currently offers support for W3C TTML but not for SMPTE TTML.

HDS supports data tracks that can hold DFXP subtitles, a format based on TTML.

3.3.8.4. Ad Insertion

It is a common requirement for content publishers to deliver content – movies, TV shows or even Live Content – with advertisements displayed in front of each new asset. Encoding the material with the advertisement as a single file is not a good approach as the workflow is not dynamic and ad-insertion in live content cannot be supported. Another solution could be to deliver the advertisements as a movie, but Adaptive Streaming technology limitations will lead to low quality on start of content. The advertisement will start playing with low data rate to ensure the client has enough bandwidth to sustain the rate and then if possible move to higher bitrate chunks. The main content will also have the same behavior leading to end user frustration and bad user experience.

HLS has the simplest and richest implementation for ad-insertion. The EXT-X-DISCONTINUITY tag is used to let the client know where a change is coming, indicating a discontinuity between the media file that follows it and the one that preceded it. Delivery of different ads to different clients is also supported by HLS playlist.

Smooth Streaming doesn't have a dedicated mechanism to support ad insertion. There is no explicit statement indicating a change in the normal playlist thus tricks like changing material on chunk's request or proxy to redirect chunk requests should be deployed.

HDS supports ad-insertion into live, linear and video-on-demand content using Adobe Primetime capabilities. It allows for dynamic ad-insertion using sophisticated inventory management, ad-routing and IAB-compliant impression tracking and reporting capabilities.

3.3.8.5. Trick Modes (Fast-forward, Rewind, Random Access to Chapters)

Users used in traditional forms of content delivery (VHS Tapes, DVD, Blu-ray disks) expect from online video content the same level of functionality including fast-forward, rewind or random access to chapter capabilities.

HLS supports trick modes – the term is used to describe the functionality in the online video era – by use of I-Frame playlists, without the need to produce special purpose content to support them. The publisher just needs to specify where the I-Frames in the video content are. With the use of the EXT-X-I-FRAMES-ONLY tag, indicating that each media segment in the playlist describes a single I-Frame, the EXTINF tag gets a new meaning referring to the span of the I-Frame. The EXT-X-BYTERANGE tag must also be used, to identify the sub-range of the media resource containing the I-frame.

MSS supports trick modes – appearing as *trickplay* mode in Microsoft documentation – and specifically Fast Forward and Rewind with the use of SetPlaybackRate. When running forward or backward in this mode, the player shows frames of the video stream to let the user know where in the stream the video will start.

HDS supports trick-play modes – fast-forward or reverse, slow motion and random access to chapters – 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).

3.3.8.6. Custom VOD Playlists

Online video content monetization involves offering users the ability to create custom VoD playlists.

HLS offers the simplest implementation for creation of custom VoD playlists as the playlist can contain URLs that reference chunks from different encodings and locations.

In Smooth Streaming and HDS, both based on fragmented MP4, is difficult to construct a playlist with chunks from different assets.

3.3.8.7. Stream Latency

Latency in HTTP Adaptive Streaming technologies is typically calculated by the segment size and how many segments are on the network a specific time. Typically, one segment is playing in the client, one is cached and another one is downloaded.

HLS recommends a 10-second chunk, so the latency can be quite long.

MSS implements a low latency mode where sub-chunks are delivered to the client as soon as they are available. This technique reduces the latency.

HDS recommends 2-5 seconds chunks, so the latency is typically lower than HLS.

3.3.9. Comparison of Adaptive Streaming Technologies

Based on the above presentation of the three Adaptive Streaming technologies most commonly used in the market and the advantages/disadvantages of each of these technologies, an evaluation from 0 to 10 will be given in the following table.

	HLS	MSS	HDS
Infrastructure Cost	10	8	10
Client Compatibility	6	6	8
Fast Start/ Channel Change	5	8	8
Playlist Manipulation	10	5	5

Bandwidth Efficiency	6	10	10
Developer Friendly	7	8	8
Chunk Management	5	10	10
Support of Multiple Audio Tracks	7	9	9
Digital Rights Management	6	9	10
Closed Captions/Subtitles	9	9	7
Ad Insertion	10	6	8
Trick Modes	8	7	8
Custom VOD Playlists	10	5	5
Stream Latency	5	10	7

Table 2.Feature Comparison Table

3.4. Towards Standardization – MPEG-DASH

Each of the three main HTTP Adaptive Streaming technologies described before uses different methods and formats creating a challenge for content delivery operators who must separately encode, store and transport each piece of video content in order to be compatible with all three formats and consequently end user devices. This costly, time-consuming process creates a scalability issue for operators and, in turn, reduces the quality of streaming video content for end users. A standard for HTTP streaming of multimedia content would allow a client to stream content from any standard-based server, enabling consistent playback and unification of servers and clients of different vendors.

To resolve the frustrations associated with multiple HTTP delivery technologies, MPEG (Moving Picture Experts Group) in an effort to create an open HTTP streaming standard issued a call for proposal in 2009. Over the next two years a specification was developed by MPEG with close collaboration by standards groups such as Third Generation Partnership Project (3GPP), Digital Entertainment Content Ecosystem (DECE), OIPF and World Wide Web Consortium (W3C). MPEG-DASH (Dynamic Adaptive Streaming over HTTP) was born, providing a single delivery format for efficient, high-quality streaming of multimedia content over the Internet. MPEG-DASH was published as an ISO standard in March 2012 (ISO/IEC 23009-1:2012). A second edition of the standard was published in May 2014 as ISO/IEC 23009-1:2014.

3.4.1. MPEG DASH Technology

All HTTP adaptive streaming technologies consist of two components, the encoded A/V streams usually delivered as segments to the client and the manifest files that identify the streams for the player and contain their URL addresses. In MPEG-DASH the encoded A/V streams are called the Media Presentation and the manifest files are called Media Presentation Descriptions.

A Media Presentation consists of one or more consecutive periods that break up the media content in pieces. Each period can contain multiple adaptation sets containing the actual media content (audio, video and data). The DASH specification allows the content to be muxed or represented in elementary streams, supporting features like multiple audio tracks. Each adaptation set contains multiple representations, effectively enabling the adaptive streaming. As seen in Figure 15 representations vary in resolutions and bitrates.

Each representation is then divided into segments that will be delivered to the client. There are two file segment types supported in DASH – MPEG2 TS and ISO Base media file format (ISO BMFF). MPEG2 TS is what HLS currently uses, and ISO BMFF is what Smooth Streaming and HDS currently use. It is obvious that MPEG tried to promote easy migration of existing adaptive streaming content to MPEG DASH, as the segmented content can remain the same and only the MPD file changes.

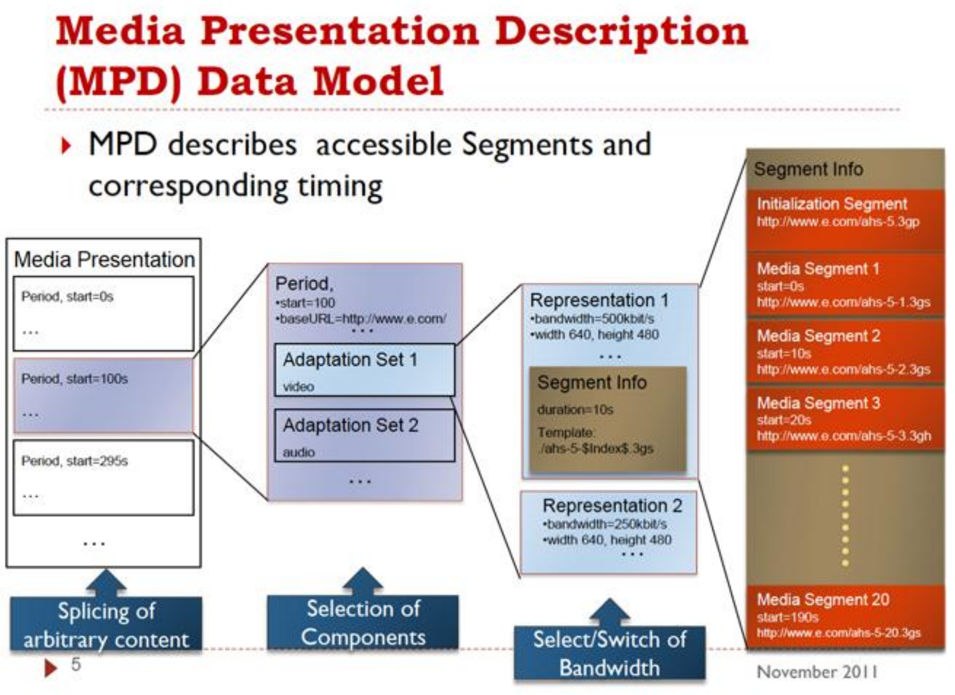
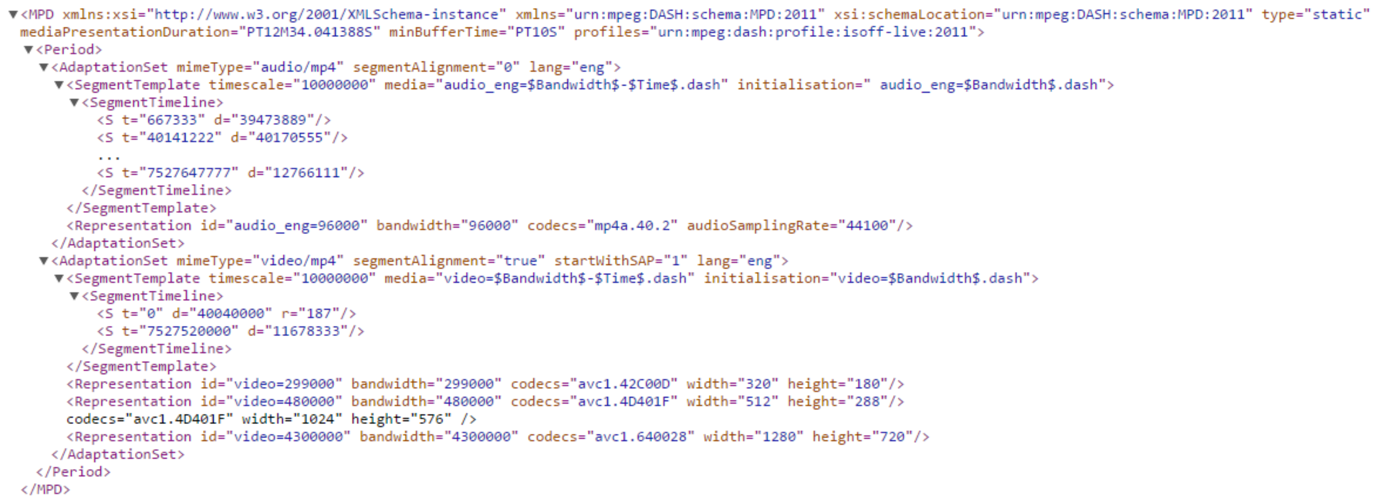


Figure 15. MPD Data Model

The Media Presentation Description (MPD) is an XML file that lists available content components and the location of all alternative streams. The client has the ability to adapt to network and CPU status as well as to user input enabling/disabling subtitles or changing audio tracks [8].



```
▼<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:mpeg:DASH:schema:MPD:2011" xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" type="static"
 mediaPresentationDuration="PT12M34.041388S" minBufferTime="PT10S" profiles="urn:mpeg:dash:profile:isoff-live:2011">
  ▼<Period>
    ▼<AdaptationSet mimeType="audio/mp4" segmentAlignment="0" lang="eng">
      ▼<SegmentTemplate timescale="10000000" media="audio_eng=$Bandwidth$-$Time$.dash" initialisation=" audio_eng=$Bandwidth$.dash">
        ▼<SegmentTimeline>
           <S t="667333" d="39473889"/>
           <S t="40141222" d="40170555"/>
           ...
           <S t="7527647777" d="12766111"/>
         </SegmentTimeline>
       </SegmentTemplate>
       <Representation id="audio_eng=96000" bandwidth="96000" codecs="mp4a.40.2" audioSamplingRate="44100"/>
     </AdaptationSet>
    ▼<AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" lang="eng">
      ▼<SegmentTemplate timescale="10000000" media="video=$Bandwidth$-$Time$.dash" initialisation="video=$Bandwidth$.dash">
        ▼<SegmentTimeline>
           <S t="0" d="40040000" r="187"/>
           <S t="7527520000" d="11678333"/>
         </SegmentTimeline>
       </SegmentTemplate>
       <Representation id="video=299000" bandwidth="299000" codecs="avc1.42C00D" width="320" height="180"/>
       <Representation id="video=480000" bandwidth="480000" codecs="avc1.4D401F" width="512" height="288"/>
       codecs="avc1.4D401F" width="1024" height="576" />
       <Representation id="video=4300000" bandwidth="4300000" codecs="avc1.640028" width="1280" height="720"/>
     </AdaptationSet>
   </Period>
 </MPD>
```

Figure 16. Sample MPD File

The MPEG-DASH specification supports the following features:

- It is codec-independent, supporting H.264, WebM and other codecs.
- ISO Base Media File Format and MPEG-2 transport streams.
- ISO/IEC 23001-7: Common Encryption.
- Trick modes for seeking, fast forwards and rewind.
- Ad-insertion.
- Multiple Audio Tracks, Closed Captions/Subtitles.
- Clock Drift Control for Live Content.
- Metrics for Reporting the Client Session Experience.

Even though MPEG-DASH was created to be a true universal protocol for video delivery over Internet and been supported by standard groups and media companies, it is still far from being adopted by the market. Analysts and market leaders state the following reasons as key factors of why DASH didn’t achieve widespread adoption yet [9]:

- Intellectual Property and Royalties. Some of the components that composed DASH come with royalty obligations such as MPEG-2. On top of that it is unclear if all participants want a royalty-free solution or not.

- Key market players like Apple are still hesitant to adopt DASH, mainly because of the widespread adoption of their proprietary protocols, HLS in Apple's case.
- Incomplete standardization of HTML5 video codec.
- Early DASH adopters, like Netflix and Hulu, tend to use only a portion of the DASH specification or even mix it with technologies out of DASH scope of work.
- The original DASH specification tried to be everything for everyone and suffered from excessive ambiguity.

The lack of a mandatory audio/video codec and delivery format in the initial specification ended up creating an interesting opening to all kinds of technology assemblies, but not a precise set that could be implemented on a lot of devices in an interoperable manner. To address this issue and push media industry to adopt DASH, the DASH-IF (DASH Industry Forum) defined a subset of the standard to serve as a base profile that all implementations have to include. The first decision made was to focus on H.264/MPEG-4 encoding and drop support of the MPEG-2/TS implementation. As a result, DASH-AVC/264 was announced in May 2013 in an effort to achieve broad adoption. Furthermore, DASH-IF has identified what it calls "interoperability points" that use AAC as the audio codec, AVC as the video codec (also known as H.264 or MPEG-4 Part 10), fMP4 as the base container format also known as the ISO Base Media File Format (ISOBMFF), the SMPTE-TT subtitle format and MPEG Common Encryption for content protection (DRM) [10]. A newer version was released in August of the same year, adding HD support up to 1080p (through the new interoperability point DASH-AVC/264 HD) and bringing multi-channel audio extensions from Dolby, DTS, MPEG Surround, and HE-AACv2 level 4 and 6. DASH-IF didn't stop its efforts to promote standardization of DASH. It released preliminary DASH-HEVC/265 guidelines, adding three new interoperability points (DASH-HEVC/265, DASH-HEVC/265 1080p 8bit and 1080p 10bit) with support for HEVC up to 1080p 10bit.

The inclusion of MPEG Common Encryption (CENC - ISO/IEC23001-7) in the MPEG-DASH specification is an important contribution towards a simple, effective system that enables different DRM systems to share keys, key identifiers, encryption algorithms, parameters and signaling. The underlying implementation is left to each DRM system thus decryption and playback of encrypted content is dependent on specific DRM support [11]. Common Encryption enables competing DRM technologies such as Microsoft PlayReady, Adobe Access and Google

Widevine to be used in end user devices, giving content publishers and customers the desired flexibility and interoperability.

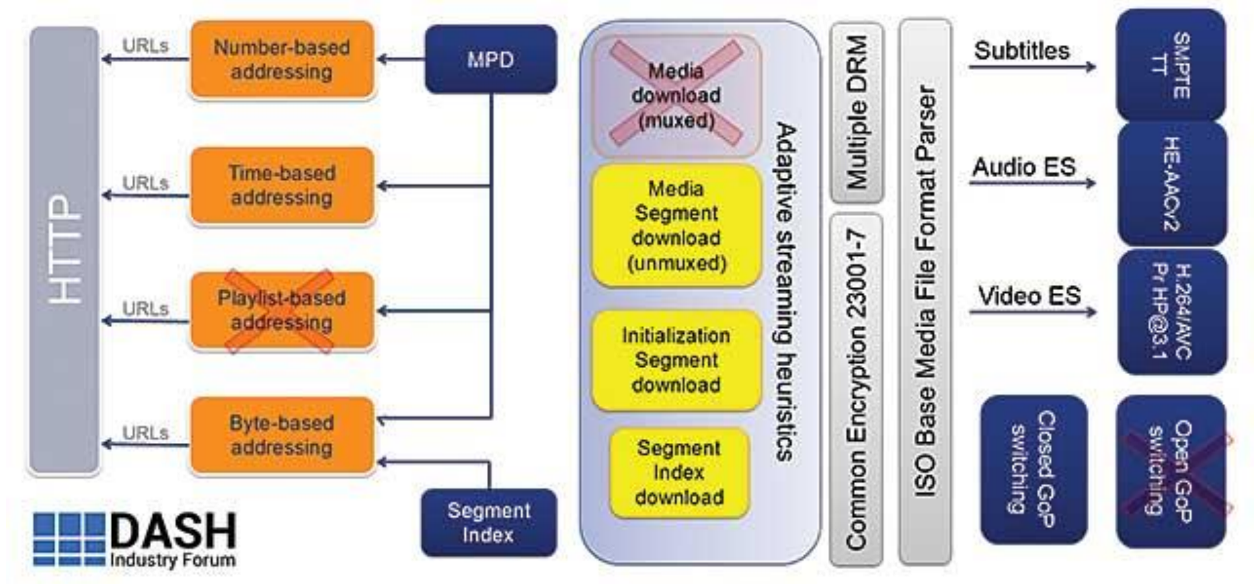


Figure 17. DASH IF Guidelines

3.4.1.1. *Feature Comparison with Proprietary Technologies*

Support of Multiple Audio Tracks – DASH natively supports multiple audio tracks, as the base container format ISO Base Media File Format (ISOBMFF) accepts elementary audio streams.

Digital Rights Management (DRM) – DASH specifies MPEG Common Encryption as a system to support different DRM systems with common keys, key identifiers, encryption algorithms, parameters and algorithms. This allows DASH to support the majority of the DRM systems including proprietary PlayReady, Adobe Access and Widevine.

Closed Captions/Subtitles – DASH is supporting closed captions and subtitles using ISO Base Media File Format. Media Source Extensions (MSE) specification allows creation of DASH-capable, HTML5 web players with ability to enable/disable closed captions and subtitles.

Ad Insertion - The difficulty of inserting ads with the three existing delivery methods is that the protocols don't support the same ad insertion methods. DASH helps adaptive bitrate advertising on many different types of client devices by supporting the dynamic insertion of advertising content into multimedia streams. In both live and on-demand use cases, commercials can be inserted either as a period between different multimedia periods or as a segment between different multimedia segments.

Trick Modes - DASH supports VoD trick modes for pausing, seeking, fast forwarding and rewinding content. The way that DASH supports these modes is a major improvement over the way that the three existing streaming protocols handle these functions now.

Custom VoD playlists – DASH supports custom VoD playlists using MPD files.

Stream Latency - DASH supports ISOBMFF which inherently supports use of segments before they are entirely downloaded in the client. In addition, DASH supports several tweaks in the MPD such as *availabilityStartTime* and *minBufferTime* that can be used to fine tune the latency experienced in the client.

3.4.1.2. DASH Future

MPEG, in the continuous effort to prove that DASH is here to stay, has launched a program called "Core Experiments" in order to feed the new versions of the DASH specification.

Until today, all adaptive streaming technologies including MPEG-DASH are based only on client feedback to take decisions and adapt the quality. MPEG wants to change that, so it launched the Server and Network Assisted DASH (SAND) initiative. SAND introduces the concept of DASH Aware Network Element (DANE) which understands that DASH content is being delivered inside the network and also provides a standardized information exchange between the DASH client, the Origin Server, the DANE and other third-party DASH distribution monitoring servers. Exchanged information include SAND parameters passed by the delivery network to assist client operations, metrics collected by the clients and Parameters for enhancing delivery (PED) through DANE [12]. The goal is to prevent clients from overusing the provided bandwidth (which under certain cases triggers continuous bitrate switches) and offer network operators better control over content delivered on top of their networks.

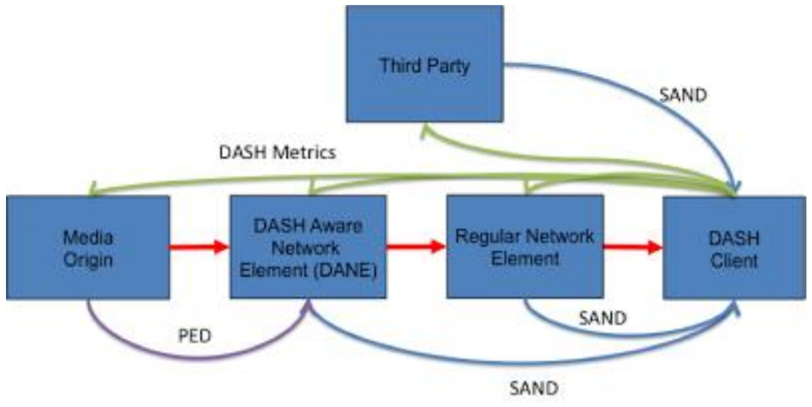


Figure 18. SAND Architecture

DASH over Full Duplex HTTP-based Protocols (FDH) aims to provide new options for large scale and low-latency streaming, based on the ability of DASH to push newly created segments to clients as soon as they enter the delivery network, thus not taking into consideration the client requests and feedback. FDH will be compatible with existing DASH ecosystem and the goal is to provide low latency of the broadcast distribution and scalability for OTT video delivery.

URI Signing for DASH (CE-USD) aims to prevent a client from skipping pre-roll and mid-roll advertisements, grant temporary access to the contents based on tokens or prevent deep linking, while Spatial Relationship Description (SRD) is designed to explore how we can combine several spatially related videos in a tiled or zoomed combination, providing an approach to tiled streaming for UltraHD video delivery.

3.4.2. Interoperability of DRM Technologies

Everyone in the media industry, including end customers, expect to have the ability to utilize a cross-platform web video delivery technology enabling easy access without the disadvantages of current technologies. DRM plays a key role in this experience as it is responsible for non standardization of currently provided end-to-end solutions. The Media Industry invests in the DASH Media Framework to be the technology that will also simplify the use of digital rights management technologies, to protect valuable content and enable the distribution on the open web.

The DASH Media Framework consists of numerous standards, the most important of which that enable DRM interoperability are: ISO MPEG Live Dynamic Adaptive Streaming over HTTP (DASH), ISO MPEG DRM-interoperable Common Encryption (CENC), Fragmented MP4 encoded with technologies such as the UltraViolet Common File Format (CFF) or the Common Streaming Format (CSF), HTML5 Media Extensions such as Media Source Extensions (MSE), Encrypted Media Extensions (EME) and Web Crypto Extensions and OAuth 2.0-based authentication (AuthN) and authorization (AuthZ) protocols such as the Online Multimedia Authorization Protocol (OMAP) [13].

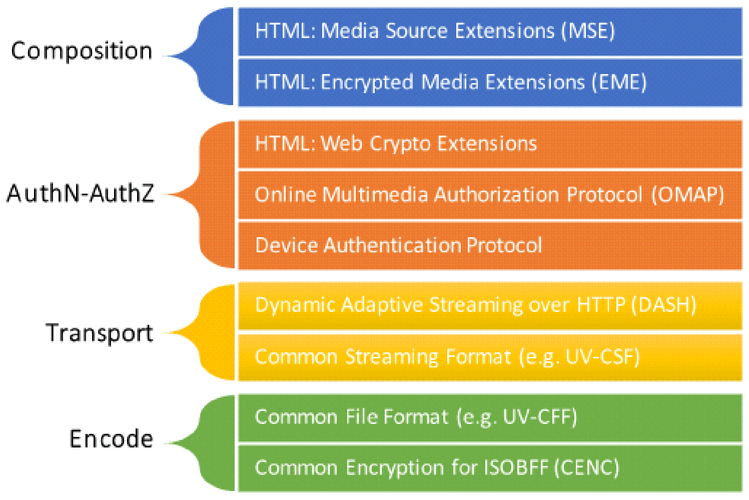

Composition
HTML: Media Source Extensions (MSE)
HTML: Encrypted Media Extensions (EME)
AuthN-AuthZ
HTML: Web Crypto Extensions
Online Multimedia Authorization Protocol (OMAP)
Device Authentication Protocol
Transport
Dynamic Adaptive Streaming over HTTP (DASH)
Common Streaming Format (e.g. UV-CSF)
Encode
Common File Format (e.g. UV-CFF)
Common Encryption for ISOBFF (CENC)

Figure 19. DASH DRM Standards

3.4.2.1. Encoding DRM Interoperable Content

Until the CENC specification emerged, content was tied to a specific application on a specific device with a specific DRM technology. Even encoding formats were DRM-specific, making encoded content to work across DRMs difficult.

The Common Encryption (CENC) protection scheme specifies standard encryption and key mapping methods that can be utilized by one or more digital rights management (DRM) and key management systems to enable decryption of the same file using different DRM systems. Such an approach supports a broader range of accessible clients from a single content stream. The scheme operates by defining a common format for the encryption and signaling related metadata necessary to decrypt the protected streams, yet leaves the details of rights mappings, key acquisition and storage, DRM compliance rules, etc., up to the DRM system or systems supporting CENC, enabling multi-DRM protection for adaptive streaming. For instance, DRM systems supporting the Common Encryption protection scheme must support identifying the decryption key via 'cenc' key identifier (KID) but identification of the decryption key is left to the specific DRM. It is transparent to the end user that the key acquisition requires a different backend server. With common encryption the content plays regardless of the application, the device or the underlying DRM [14].

3.4.2.2. *How CENC works*

A client understands that encoded content is encrypted with a DRM system supporting CENC, using specific information in the delivered material. This information is declared in Protection Scheme Information Box (‘sinf’) that contains a Scheme Type Box (‘schm’) so that the scheme is identifiable. The Scheme Type Box contains the following information:

- `scheme_type` field is set to value ‘cenc’
- `scheme_version` field is set to 0x00010000 to mark Major Version 1

The Protection Scheme Information Box shall also contain a Scheme Information Box (‘schi’).

The encryption metadata defined by the ‘cenc’ Common Encryption Scheme can be categorized as follows:

- Protection System Specific Data – Defines a place where protection systems store their own data using a common mechanism. Common Encryption Scheme handles only the information related to the storage place and not the specific storing mechanisms.
- Common Encryption Information for a media track – These metadata include default values for the key identifier (KID), initialization vector size and encryption flag.
- Common Encryption Information for groups of media samples – These metadata override default values for key identifier (KID), initialization vector size and encryption flag to allow groups of samples to use different keys, or provide a mix of clear and encrypted content within the same track. This data is contained in a SampleGroupDescriptionBox ('sgpd') that is referenced by a SampleToGroupBox (‘sbgp’).
- Encryption information for individual media samples - Provides initialization vectors and, if required, sub sample encryption data. This data is sample auxiliary information, referenced by using a SampleAuxiliaryInformationSizesBox (‘saiz’) and a SampleAuxiliaryInformationOffsetsBox ('saio').

Each sample in a protected track shall be associated with an IsEncrypted flag, IV_Size, and KID. These values can be the default ones specified in the TrackEncryptionBox, specified for the specific sample group or using a combination of these two techniques.

CENC is compatible with ISO Base Media File Format (ISO BMFF) where content metadata are declared in Movie (‘moov’) or Movie Fragment (‘moof’) containers. These containers include the Protection System Specific Header Box (‘pssh’) that contains information needed by a Content Protection System to play back the content. The data format is specified by the system identified by the SystemID parameter. A client can successfully decode a fragment by collecting metadata from the initial movie box and metadata from the specific movie fragment box. The data encapsulated in the Data field may be read by the identified Content Protection System client to enable decryption key acquisition and decryption of media data.

The key differentiation of CENC is that a single encoded file can be constructed in a way to be playable by multiple key and digital rights management (DRM) systems. This is accomplished by including Protection System Specific Header boxes for each supported system. Clients follow the steps below to get all of the Protection System Specific data:

- Examine all Protection System Specific Header boxes in the Movie Box and in the Movie Fragment Box associated with the sample that need to be decrypted.
- Match the SystemID field in this box to the SystemID(s) of the DRM System(s) they support.
- Match the KID associated with the sample with one of the KID values in the Protection System Specific Header Box.

The TrackEncryptionBox (‘tenc’) contains default values for the IsEncrypted flag, IV_size, and KID for the entire track. For files with only one key per track, this box allows the basic encryption parameters to be specified once per track instead of being repeated per sample.

CENC defines that media data shall use the Advanced Encryption Standard, using 128-bit keys in Counter Mode (AES-CTR). The scheme defines two elementary stream encryption formats, full sample encryption and subsample encryption. In full sample encryption the entire sample is encrypted as a single encryption unit whereas in subsample encryption the sample is broken into smaller units each containing a clear area and an encrypted area. The fields listed below provide needed information used by the common encryption scheme:

- `IsEncrypted` takes two values, 0x0 (Not encrypted) and 0x1 (Encrypted) and is the identifier of the encryption state of the samples in the track.
- `IV_size` is the size in bytes of the InitializationVector field. Supported values are 0 (samples not encrypted), 8 (64-bit initialization vectors) and 16 (128-bit initialization vectors).
- `KID` is the unique identifier for the decryption key of the associated samples. For clear tracks KID value is 0x0.
- `InitializationVector` specifies the initialization vector needed for decryption of a sample.
- `subsample_count` specifies the number of subsample encryption entries present for this sample. Supported values are greater than 0 if present.
- `BytesOfClearData` specifies the number of bytes of clear data at the beginning of the subsample encryption entry.
- `BytesOfEncryptedData` specifies the amount of encrypted data following the clear data.

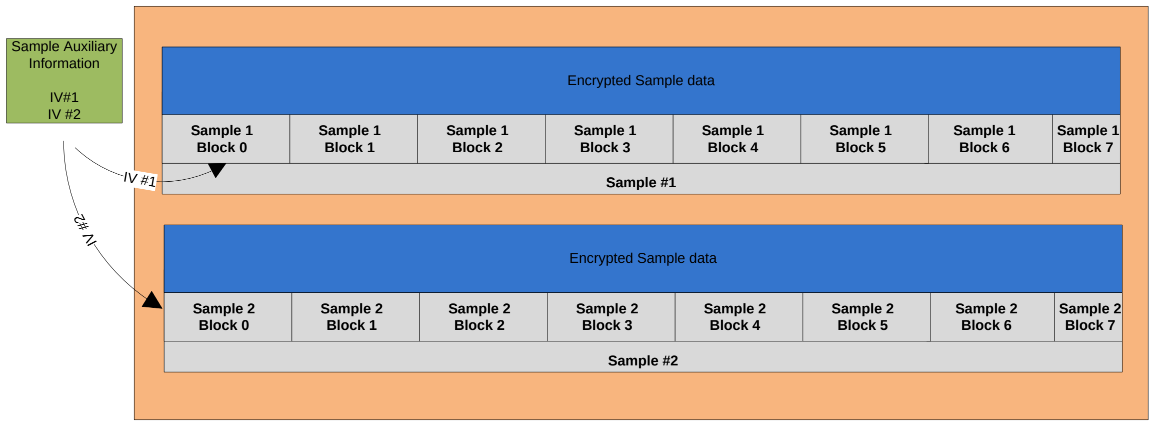


Figure 15. Sample-based encryption for AES-CTR

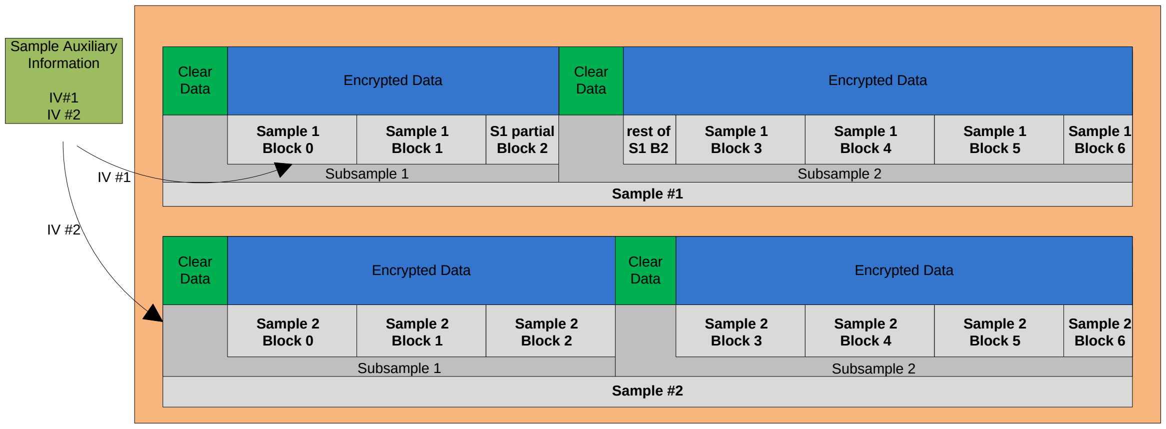


Figure 16. Subsample-based encryption scheme for AES-CTR

In MPEG DASH streaming, it is desired to include the default_KID field in the Track Encryption Box of an ISO Media Track, in an XML accessible by the client before the media. The client can then get the license with the key in advance.

The syntax of the DASH ContentProtection Descriptor, specified in DASH, allows the addition of an attribute in a declared namespace different from the DASH namespace. This can be used to add the cenc:default_KID attribute. Below is the specified schema:

```
<!— DASH ContentProtection Descriptor -->
<xs:sequence>
    <xs:element name="ContentProtection" type="DescriptorType"
minOccurs="0"maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>

<!— DASH Descriptor Complex Type -->
<xs:complexType name="DescriptorType">
   <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
    <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
    <xs:attribute name="value" type="xs:string"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

A valid XML of the above schema could be the following. In this scheme Common Protection is used with PlayReady and Verimatrix DRM systems compatibility.

```
<!-- Common Encryption -->
<ContentProtection
  schemeIdUri="urn:mpeg:dash:mp4protection:2011"
  value="cenc"
  cenc:default_KID="8D1A585A-A0B4-A942-917A-C1B659142B2A">
</ContentProtection>
<!-- PlayReady -->
```

```
<ContentProtection
  schemeIdUri="urn:uuid:9A04F079-9840-4286-AB92-E65BE0885F95">
</ContentProtection>
<!-- Verimatrix -->
<ContentProtection
  schemeIdUri="urn:uuid:9A27DD82-FDE2-4725-8CBC-4234AA06EC09">
</ContentProtection>
```

3.4.2.3. Playback DRM-protected media

Common Encryption is providing the basis for interoperable encoded media but a truly interoperable DRM solution requires a way to playback DRM-protected media in standard browsers, without using proprietary plug-ins.

In 2011, Microsoft and Netflix gave presentations at a W3C Web and TV Interest group meeting in Berlin, focusing on the need to support DRM-protected media in browsers. By 2012 Microsoft, Google and Netflix had submitted a joint proposal to the W3C HTML Working Group that would enable browsers to consume DRM-protected content without the use of plug-ins – the Encrypted Media Extensions (EME). Before EME, all DRM-capable players were DRM-specific: Flash Player for Adobe Primetime DRM, Silverlight for PlayReady, and downloadable plug-ins for the others e.g. Google Widevine.

This proposal allows browsers through a JavaScript API to select content protection mechanisms, control license/key exchange, and implement custom license management algorithms. It supports a wide range of use cases without requiring client-side modifications in each user agent for each use case. This also enables content providers to develop a single application solution for all devices. The diagram below shows an example flow of EME architecture [15].

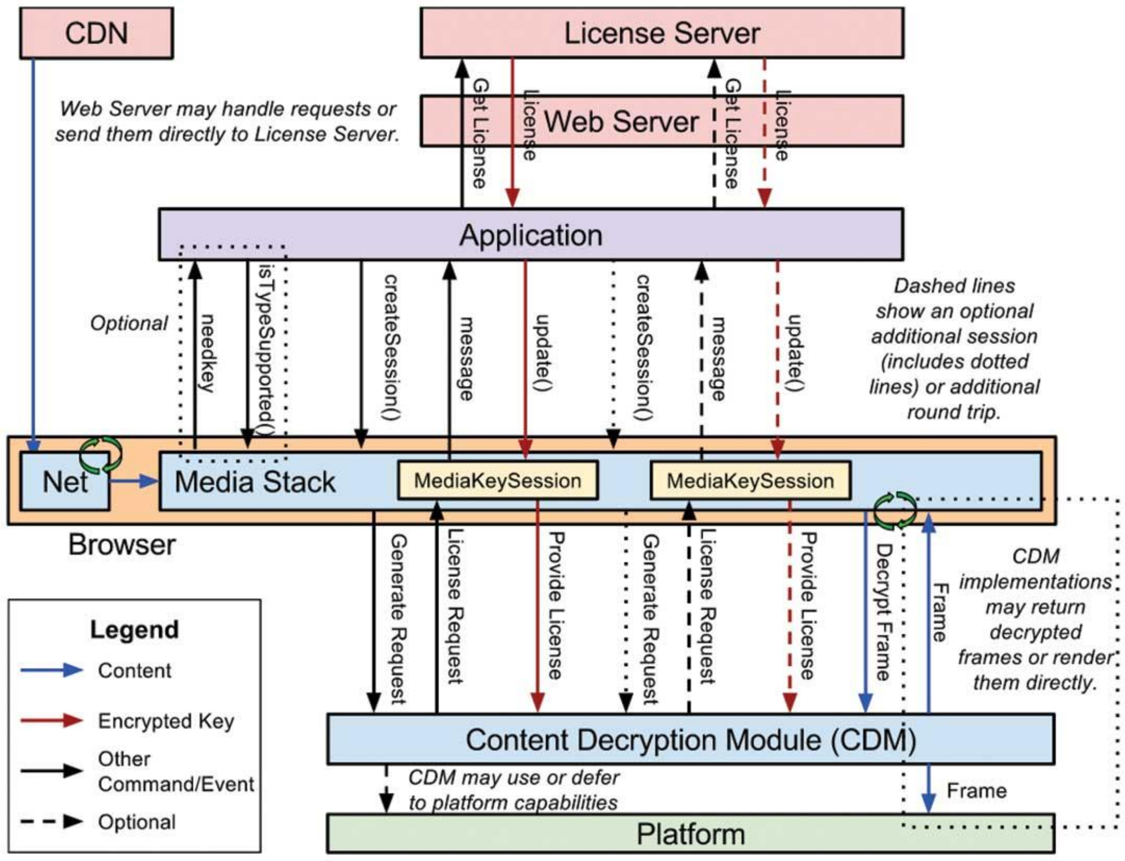


Figure 17. EME Architecture

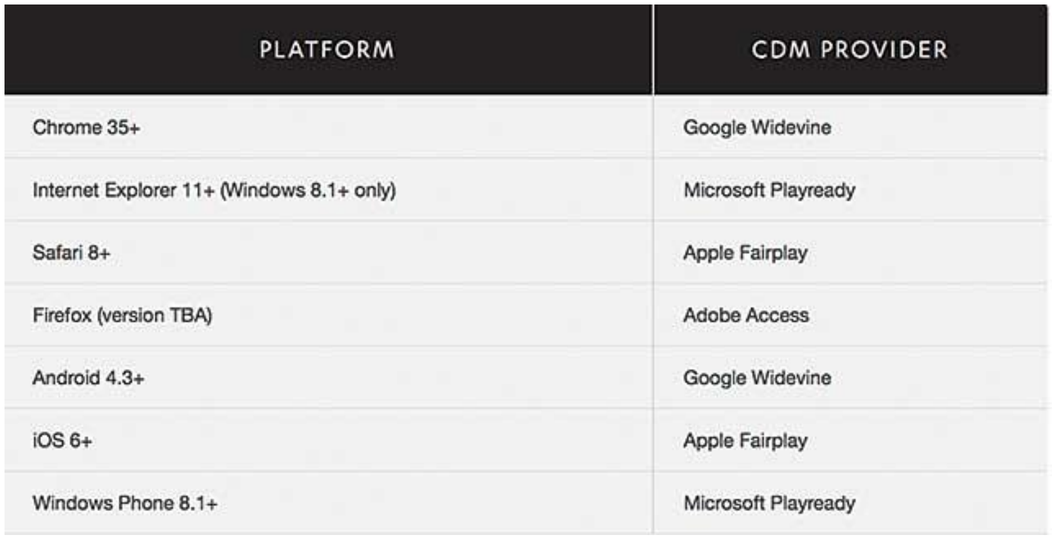
The critical parts of the architecture will be analyzed below:

- Content Decryption Module (CDM): Client component that provides the functionality, including decryption for one or more Key Systems.
- Key System: Key System describes a decryption mechanism and/or a content protection provider. Key System strings provide unique identification of a Key System and are used by the user agent to select a CDM and identify the source of a key-related event.
- Key Session: Provides the mechanism for message exchange with the CDM, including delivery of keys to CDM. Sessions can be uniquely identified using a Session ID, generated by the CDM.
- Key: Refers to a decryption key that can be used to decrypt media data. Each key is uniquely identified by a Key ID and is associated with the session used to provide it to the CDM.

- License: A key system-specific state information that includes one or more keys each associated with a Key ID.

All EME-compatible DRM systems have agreed to support the ISO base media file format (ISO BMFF), H.264 video codec, AAC audio compression and Common Encryption (CENC) scheme as well as the HTML Media Source Extensions (MSE) to facilitate standard adaptive bitrate streaming protocols such as HLS and DASH. This means that the basic packages of encoded and encrypted files produced for any particular EME DRM technology should play on all EME DRM technologies with the proper license key.

The Content Decryption Module is the client that provides the decryption functionality for one or more key systems. Until today though, browsers and Mobile OS vendors supported their own proprietary CDMs, as shown in Figure 24, which are not interoperable. At first it seems that the industry has not accomplished the initial goal to move from proprietary DRM systems to proprietary CDMs, but that is not the whole truth. The only system in the DRM chain that is not interoperable is the DRM servers that the client requests and downloads the content license and the decryption key.



PLATFORM	CDM PROVIDER
Chrome 35+	Google Widevine
Internet Explorer 11+ (Windows 8.1+ only)	Microsoft Playready
Safari 8+	Apple Fairplay
Firefox (version TBA)	Adobe Access
Android 4.3+	Google Widevine
iOS 6+	Apple Fairplay
Windows Phone 8.1+	Microsoft Playready

Figure 18. Platforms and CDM providers

3.4.2.4. So what makes CENC and EME a truly interoperable solution?

- Compatibility with all web browsers (desktop and mobile) without the need of installing proprietary plug-ins.

- A single encoded stream works for all DRM solutions. Current DRM solutions require encoding of the stream to be specific for their needs. It was required from content publishers to produce 3-4 different streams only for the purpose of DRM compatibility.
- Compatible with existing DRM solutions giving backward compatibility to content publishers.
- Support of MPEG-DASH, an effort to standardize HTTP Adaptive Streaming.

4. Hybrid broadcast broadband TV - HbbTV

During the past decade, television consumption has been transformed by the rise of on-demand services and the introduction of smartphones, tablets and other TV-enabled mobile devices. The number of devices connected simultaneously to the Internet and a TV, such as STBs, game consoles and Blu-Ray players is estimated at a few hundred millions, while the number of connected devices with access to Internet will exceed six billion units this year. In an effort to harmonise the broadcast, IPTV and broadband delivery of entertainment to the end user through connected TVs and set top boxes, the HbbTV consortium promotes HbbTV initiative and standard (ETSI TS 102 796).

Several broadcasters in Europe have already adopted the HbbTV standard and more are expected to follow. HbbTV services were launched in Australia in 2014, and there is also great interest in China, Brazil, and the United States. In Germany, more than 90 percent of the broadcast market already supports HbbTV, with 10 million devices with HbbTV capabilities in the country [16].

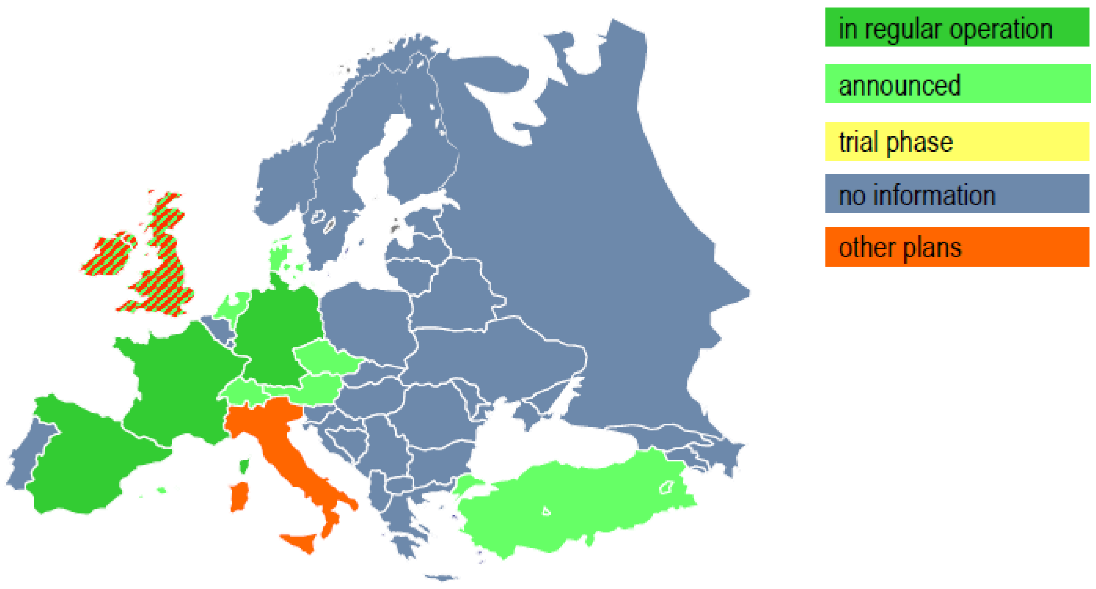


Figure 19. HbbTV Market Deployment

The founding members of the HbbTV Association together with a large group of supporting organisations jointly developed the HbbTV specification to create a global standard for hybrid entertainment services. The standard includes components of widely deployed and established technologies such as CE-HTML, DVB, W3C, Open IPTV Forum (OIPF) and MPEG-DASH.

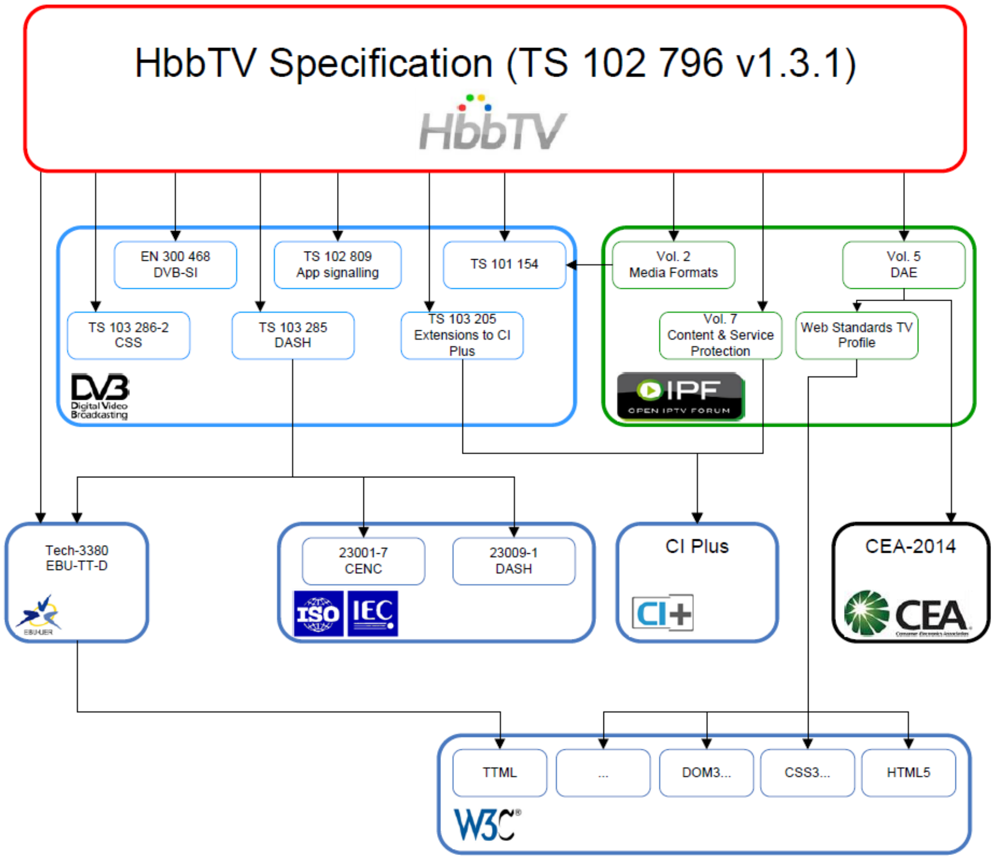


Figure 20. HbbTV Specification Overview

In the following, we will describe the most important specifications shown in the overview above.

- HTML5 provides the <video> element for properly presenting broadband video in an HTML page and the APIs for manipulating the contents of an HTML page using JavaScript.
- The OIPF Declarative Application Environment provides JavaScripts APIs for applications running in a TV environment, definitions for embedding linear A/V capabilities in an application and integration with content protection/DRM technologies.

- TS 102 809 provides components for application signaling and application transport via broadcast or broadband.
- ISO 23009-1 and TS 103 285 provide components for supporting live streaming and transporting of MPEG-DASH content via broadband and broadcast interfaces.
- ISO 23001-7 provides components for supporting MPEG-CENC capable of delivering DRM-interoperable content.
- EBU-TT-D sets the specifications for the subtitle format.

HbbTV will use as network infrastructure third party, unmanaged broadband networks, which could be a risk for the Quality of Experience delivered to end users. Currently, users enjoy a certain QoE and they demand new services to match them. To avoid lower than expected QoE, HbbTV will use two key technologies. Firstly, HEVC (High Efficiency Video Coding) can be used to reduce the bandwidth needed for the on demand services by up to 50% and, secondly, MPEG-DASH will be used for the adaptive bitrate streaming. MPEG-DASH is included as part of the HbbTV 1.5 specification and as proved is a more efficient and future-proof alternative to HLS or HDS.

HTML5 is widely adopted in mobile and desktop browsers and lately in the majority of Smart TVs. Support of HTML5 will allow video rendering without need of proprietary plug-ins and without the need of set top boxes or other devices connected to TVs. HTML5 is part of the HbbTV 2.0 specification.

HbbTV sets the specifications for a platform delivering enhanced and interactive applications designed for running on hybrid terminals that include both a DVB compliant broadcast connection and a broadband connection to the Internet. These connections work in parallel to deliver a consistent end user experience.

Uses of broadcast connection:

- Transmission of standard TV, radio and data services.
- Signaling of broadcast-related applications.
- Transport of broadcast-related applications and associated data.
- Transport of On Demand Content for Push-services.
- Synchronisation of applications and TV/Radio/Data services.

Uses of broadband connection:

- Delivery of On Demand and Live Content.
- Transport of broadcast-related and broadcast-independent applications and associated data.
- Starting applications on a Companion Screen Device (smartphones, tablets).
- Communication with applications on a Companion Screen Device or another hybrid terminal.
- Synchronising media and applications between a hybrid terminal and a Companion Screen Device or another hybrid terminal.

HbbTV platform has some key characteristics that differentiate it from previously proposed platforms and make it an interoperable solution for delivering hybrid content to end user devices:

- Open specifications, not controlled by a single authority.
- Services and content from different providers are accessible by the same terminal.
- Services and content can be DRM-protected.
- Both broadcast-related and broadcast-independent applications are supported.

4.1. HbbTV System Overview

Hybrid terminals have the capability to be connected to two networks in parallel. Through the broadcast DVB network (terrestrial, satellite or cable) the terminal can receive standard A/V services, non-realtime A/V content (typically Push Services), application data and application signaling information. The connection via the broadband interface gives the terminal capabilities for bi-directional communication with the application provider. The terminal can receive over the broadband interface application data and non-linear A/V content (typically using video streaming). Companion Screen Devices are also communicating with the terminal via the broadband interface [17].

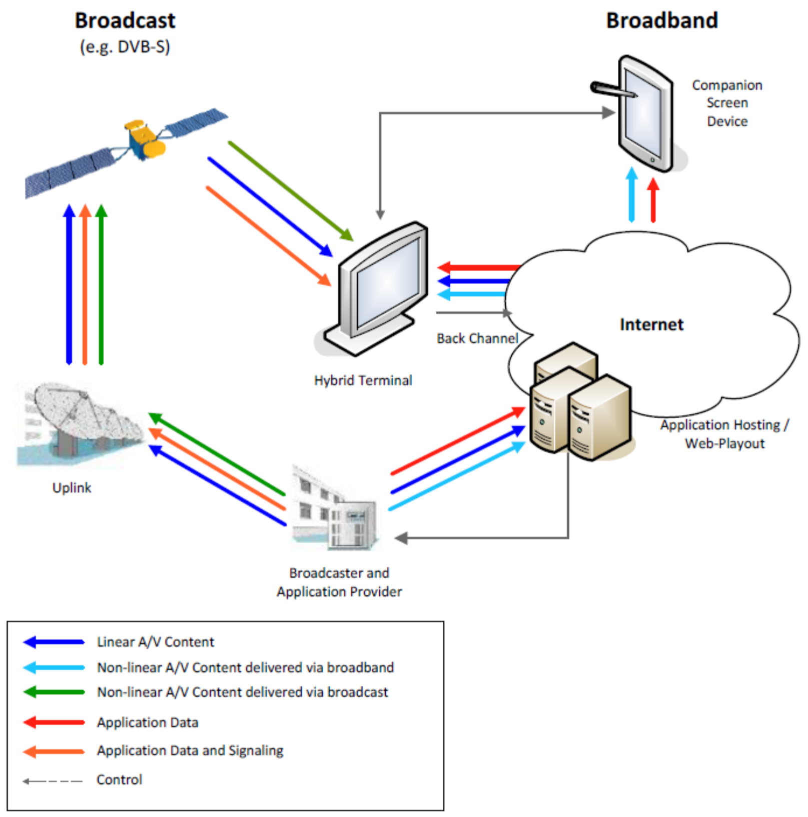

Broadcast
(e.g. DVB-S)
Broadband
Companion Screen Device
Hybrid Terminal
Back Channel
Internet
Application Hosting / Web-Playout
Uplink
Broadcaster and Application Provider
Linear A/V Content
Non-linear A/V Content delivered via broadband
Non-linear A/V Content delivered via broadcast
Application Data
Application Data and Signaling
Control

Figure 21. HbbTV System Overview

4.2. Hybrid Terminal Components

The hybrid terminal used to receive HbbTV content has two interfaces, Broadcast and Broadband. From the Broadcast interface the terminal receives AIT data, linear A/V content, non-realtime A/V content, application data and stream events. The last two streams are received by a Digital storage media command and control (DSM-CC) and then transferred to the runtime environment. Non-realtime content is received by a File Download Protocol Decoder.

The Runtime Environment is an abstract component, formed by the browser and an Application Manager, where the interactive application is presented and executed. The Application Manager evaluates the AIT (Application Information Table) to control the interactive application while the Browser is responsible for presenting and executing the application.

Linear A/V content is processed by the Broadcast Processing component in the same way as on a standard non hybrid terminal. Some information and functions from the Broadcast Processing component can be accessed by the Runtime Environment as seen in the Other Data stream flow in Figure 27. The Media Player component can also embed linear A/V content in the user interface provided by an interactive application.

The Broadband Interface of a hybrid terminal provides a connection to the Internet and a second way to request application data from multiple providers. Also, this connection is used to receive A/V content for On Demand Applications or Streaming content. All data received from the broadband interface are handled by the Internet Protocol Processing component and accordingly delivered to the Runtime Environment. A/V content is forwarded to the Media Player also controlled by the Runtime Environment. The Synchronisation Manager is used to synchronise content delivered to the hybrid terminal via the different interfaces.

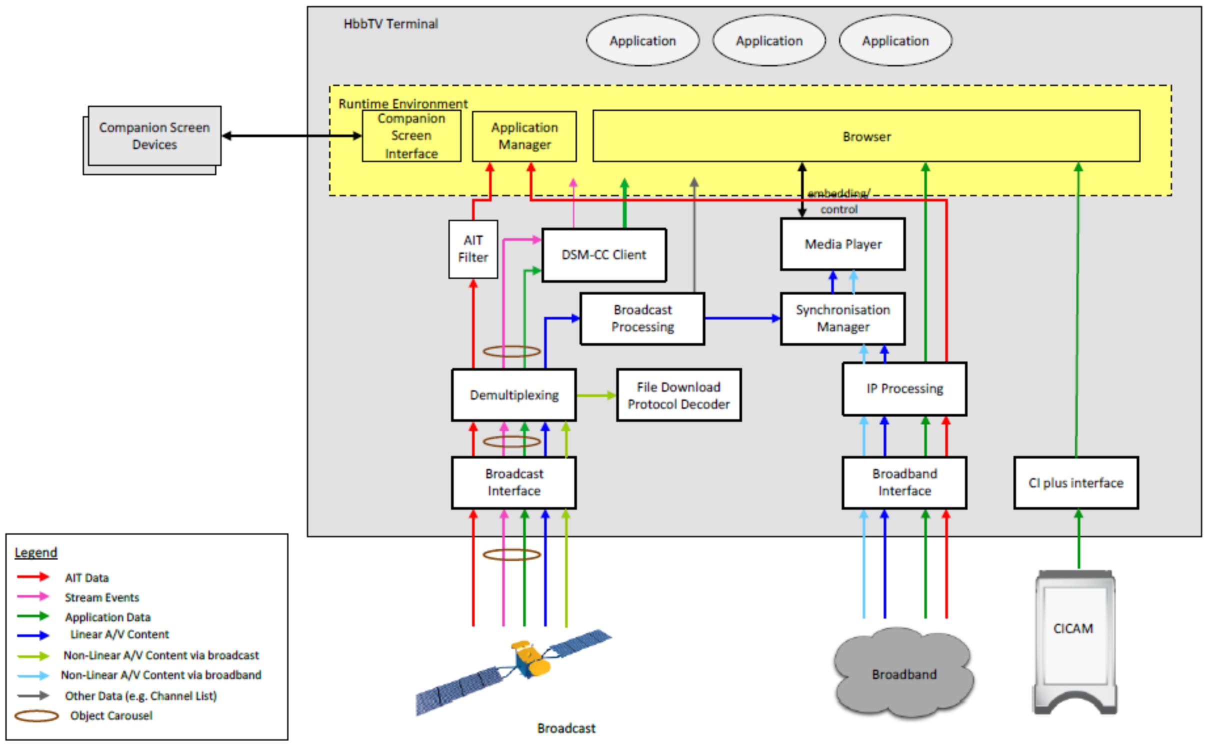


Figure 22. HbbTV Terminal Components

Through the Companion Screen Interface, the hybrid terminal can discover Companion Screen Devices and hybrid terminals and also be discovered by Companion Screen Devices. The interface allows interactive applications running in the Browser to request the installation or launch an application on a Companion Screen Device, or an application running on a Companion Screen Device can request the Browser to start an interactive application.

4.3. User Experience

HbbTV, apart from the technical specifications, also sets usability guidelines in order to provide to the end-user a homogenous experience. The behavior of the terminal as seen by the end user is something both the manufacturer and the application developer should deliver following framework guidelines provided.

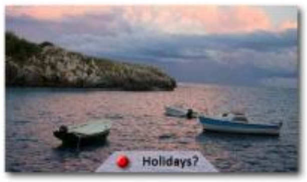
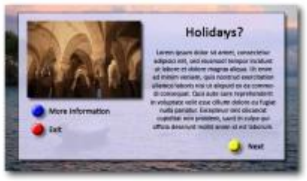
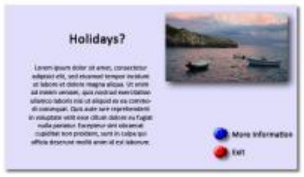
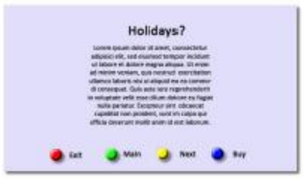
	1. Conventional TV
	2. TV with visual prompt of available information ("Red Button")
	3. TV with information overlaid (still picture only in the overlaid information, no A/V in overlay)
	4. Information with video, audio or picture inset
	5. Just information (without A/V)

Figure 23. End user screens

Some of the guidelines will be described below:

- Balance of video and application. The experience provided by the terminal should be balanced between “conventional TV” content and information delivered by an interactive application.
- Change in appearance is either initiated by the user or when a service changes through time.
- The user controls interactive applications using a user input device supplied with the terminal. Additionally users can interact with applications via a Companion Screen Application which can support more features that a conventional user input device such as a remote control.
- The user can access interactive applications using the following ways: pressing the “Red Button” can start a broadcast-related application, pressing the TEXT button can start a digital teletext application, start a broadcast-independent application using Internet TV portal if provided, and start an application from a Companion Screen application.
- Suitable errors should be delivered on screen if relative interfaces are not connected and user tries to start an application provided through these networks.
- Applications may stop when they launch other applications or a channel change is performed. Applications may also kill themselves, either by end-user request or as a consequence of internal logic. Pressing the EXIT button terminates the application.

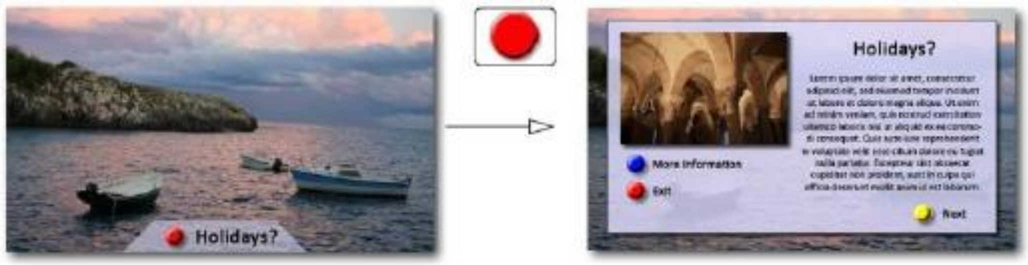


Figure 24. Red Button Application Start

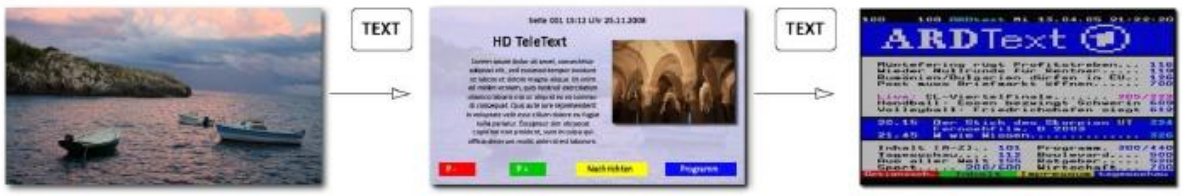


Figure 25. TEXT digital teletext application and standard teletext

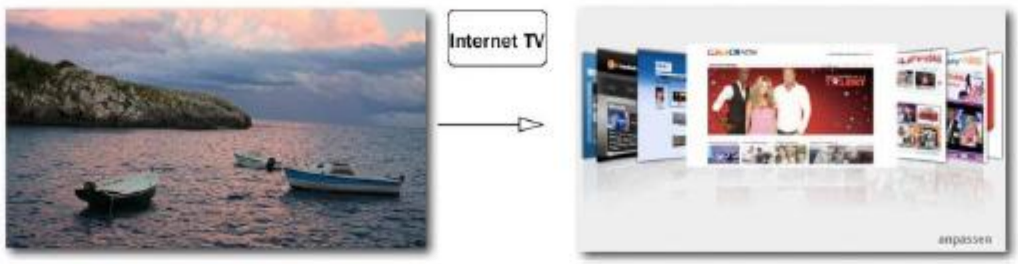
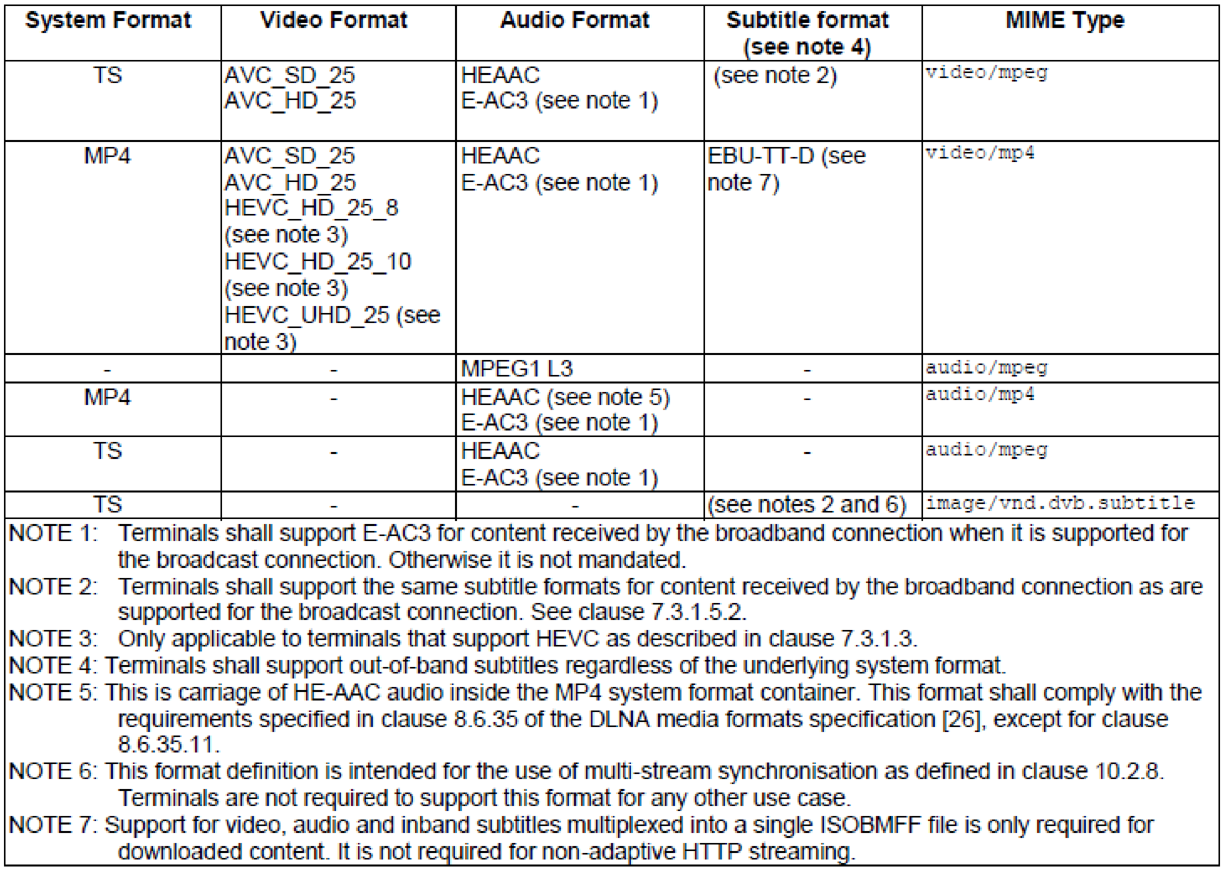


Figure 26. Start of Internet TV portal (broadcast-independent application)

4.4. System, video and audio formats

System, video and audio formats are specified in the OIPF Media Formats Specification. The table below shows the subset of the combinations of system, video, audio and subtitle formats that should be supported in HbbTV for non-adaptive HTTP streaming and video file download.



System Format	Video Format	Audio Format	Subtitle format (see note 4)	MIME Type
TS	AVC_SD_25 AVC_HD_25	HEAAC E-AC3 (see note 1)	(see note 2)	`video/mpeg`
MP4	AVC_SD_25 AVC_HD_25 HEVC_HD_25_8 (see note 3) HEVC_HD_25_10 (see note 3) HEVC_UHD_25 (see note 3)	HEAAC E-AC3 (see note 1)	EBU-TT-D (see note 7)	`video/mp4`
-	-	MPEG1 L3	-	`audio/mpeg`
MP4	-	HEAAC (see note 5) E-AC3 (see note 1)	-	`audio/mp4`
TS	-	HEAAC E-AC3 (see note 1)	-	`audio/mpeg`
TS	-	-	(see notes 2 and 6)	`image/vnd.dvb.subtitle`

NOTE 1: Terminals shall support E-AC3 for content received by the broadband connection when it is supported for the broadcast connection. Otherwise it is not mandated.
NOTE 2: Terminals shall support the same subtitle formats for content received by the broadband connection as are supported for the broadcast connection. See clause 7.3.1.5.2.
NOTE 3: Only applicable to terminals that support HEVC as described in clause 7.3.1.3.
NOTE 4: Terminals shall support out-of-band subtitles regardless of the underlying system format.
NOTE 5: This is carriage of HE-AAC audio inside the MP4 system format container. This format shall comply with the requirements specified in clause 8.6.35 of the DLNA media formats specification [26], except for clause 8.6.35.11.
NOTE 6: This format definition is intended for the use of multi-stream synchronisation as defined in clause 10.2.8. Terminals are not required to support this format for any other use case.
NOTE 7: Support for video, audio and inband subtitles multiplexed into a single ISOBMFF file is only required for downloaded content. It is not required for non-adaptive HTTP streaming.

Figure 27. System, video and audio formats for non-adaptive HTTP Streaming and content download

For MPEG-DASH the following system, video, audio and subtitle formats should be supported:

- ISOBMFF system format should be supported
- Video formats AVC_SD_25 and AVC_HD_25 should be supported. Regarding HEVC HEVC_HD_25_8, HEVC_HD_25_10 and HEVC_UHD_25 should be supported.
- Audio format HE-AAC should be supported.
- Subtitle format EBU-TT-D should be supported.

5. Conclusion

By 2018 video will dominate Internet traffic making streaming technologies a key factor for sustainability and growth. Streaming technologies used today, HLS, HDS and MSS, are closed, proprietary implementations tied to specific devices, operating systems or plug-ins. Consumers and the media industry demand a technology open, but also supported by market leaders, that will eliminate the weaknesses of the previous implementations, give new perspectives and facilitate new business models. The media industry believes that this technology is MPEG-DASH. A technology already implemented but not widely adopted mainly because it tries to do everything for everybody and is not setting strict guidelines. HbbTV, a hybrid model of broadcast and broadband delivery of content to internet-connected TVs that extensively uses DASH can strengthen its market share especially in Europe. Even if DASH has not seen widespread adoption yet, several standards it supports like CENC and EME have already found their way to the toolkit of several streaming implementations.

6. Bibliography

[1] "Bell Labs Video Traffic Study," 2012.

[2] "Streaming White Paper," EdgeCast Networks, Santa Monica, CA, 2012.

[3] T. Spangler, "Variety," [Online]. Available: http://variety.com/2014/digital/news/netflix-streaming-eats-up-35-of-downstream-internet-bandwidth-usage-study-1201360914/. [Accessed 18 4 2015].

[4] "RTSP" [Online]. Available: http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol. [Accessed 18 4 2015].

[5] E. R. Pantos, "HTTP Live Streaming draft-pantos-http-live-streaming-16," 15 April 2015.

[6] A. Zambelli, "IIS Smooth Streaming Technical Overview," 2009.

[7] "HTTP Dynamic Streaming on the Adobe Flash Platform," Adobe, 2010.

[8] W. L. T. F. M. W. D. P. I. S. Mike Luby, *Streaming Media West,* Los Angeles, CA, 2011.

[9] A. Pennington, "http://www.streamingmedia.com/," [Online]. Available: http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=90776&PageNum=1. [Accessed 12 05 2015].

[10] "http://dashif.org/," [Online]. Available: http://dashif.org/faq/#7. [Accessed 12 05 2015].

[11] N. Thorwirth. [Online]. Available: http://www.verimatrix.com/blog/201307/mpeg-dash-common-encryption-promise-broad-reach-and-interoperability. [Accessed 14 05 2015].

[12] D. C. Timmerer, "http://multimediacommunication.blogspot.gr/," 29 November 2013. [Online]. Available: http://multimediacommunication.blogspot.gr/2013/11/mpeg-news-report-from-106th-meeting.html. [Accessed 30 5 2015].

[13] J. C. Simmons and D. S. Arbanowski, "Interoperability, Digital Rights Management and the Web," September 2013.

[14] "Information technology— MPEG systems technologies — Part 7: Common encryption in ISO base media file format," 15 02 2013.

[15] G. I. David Dorwin, M. C. Jerry Smith, N. I. Mark Watson and M. C. Adrian Bateman (until May 2014), "Encrypted Media Extensions," [Online]. Available: http://www.w3.org/TR/encrypted-media/. [Accessed 20 5 2015].

[16] C. Dziadul, "http://www.broadbandtvnews.com/," [Online]. Available: http://www.broadbandtvnews.com/2014/01/15/hbbtv-a-new-perspective/.

[17] "HbbTV 2.0 Specification".