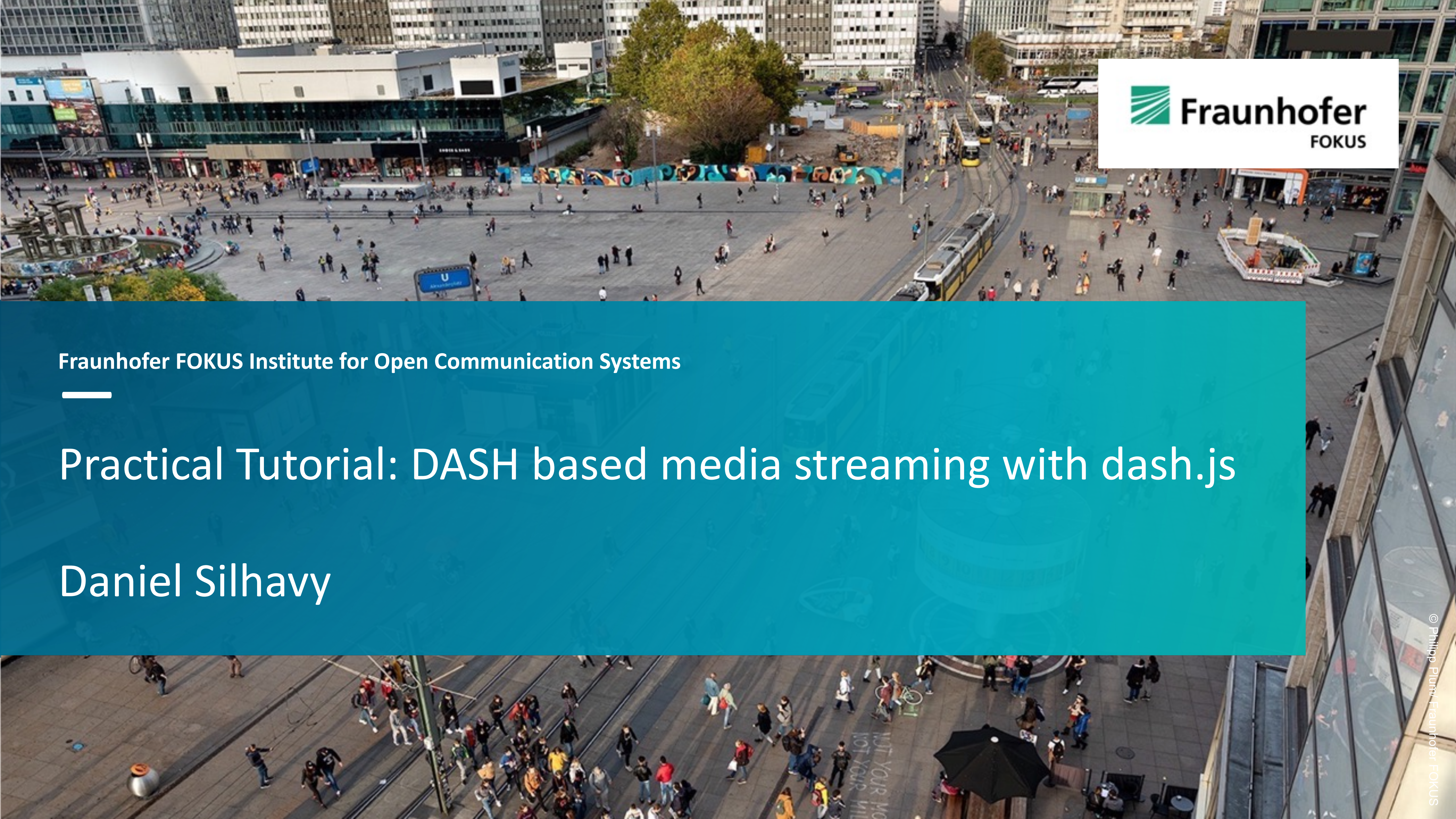


Fraunhofer FOKUS Institute for Open Communication Systems

Practical Tutorial: DASH based media streaming with dash.js

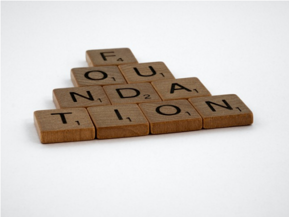
Daniel Silhavy

© Philipp Plum/ Fraunhofer FOKUS

About me

- **Daniel Silhavy (Fraunhofer FOKUS)**
- **Area of expertise**: Adaptive Media Streaming, Video Encoding, Media Player Development, Standardization, 5G Media Streaming
- **Related Open-Source Projects**:
  - Lead Developer of the dash.js project
  - 5G-MAG Reference Tools Development Team Coordinator
  - Joint Conformance Project (JCCP) Development Coordinator
- **Contact**
  - Email: daniel.silhavy@fokus.fraunhofer.de
  - LinkedIn: https://www.linkedin.com/feed/

Agenda

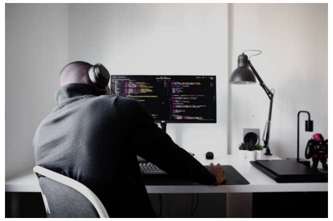


Foundations

- ABR Streaming
- MSE and EME based playback
- MPEG-DASH

dash.js

- Overview
- DRM
- Features
  - MPD Patching
  - Content Steering
  - CMCD
  - CMSD
  - CMAF Low Latency
  - Timing Problems
- Multiperiod and Gap Handling
- Testing



Stream Debugging

- Segment Inspection
- DASH Validator
- DASH-IF Livesim
- ABR Testbed

Workshop – dash.js

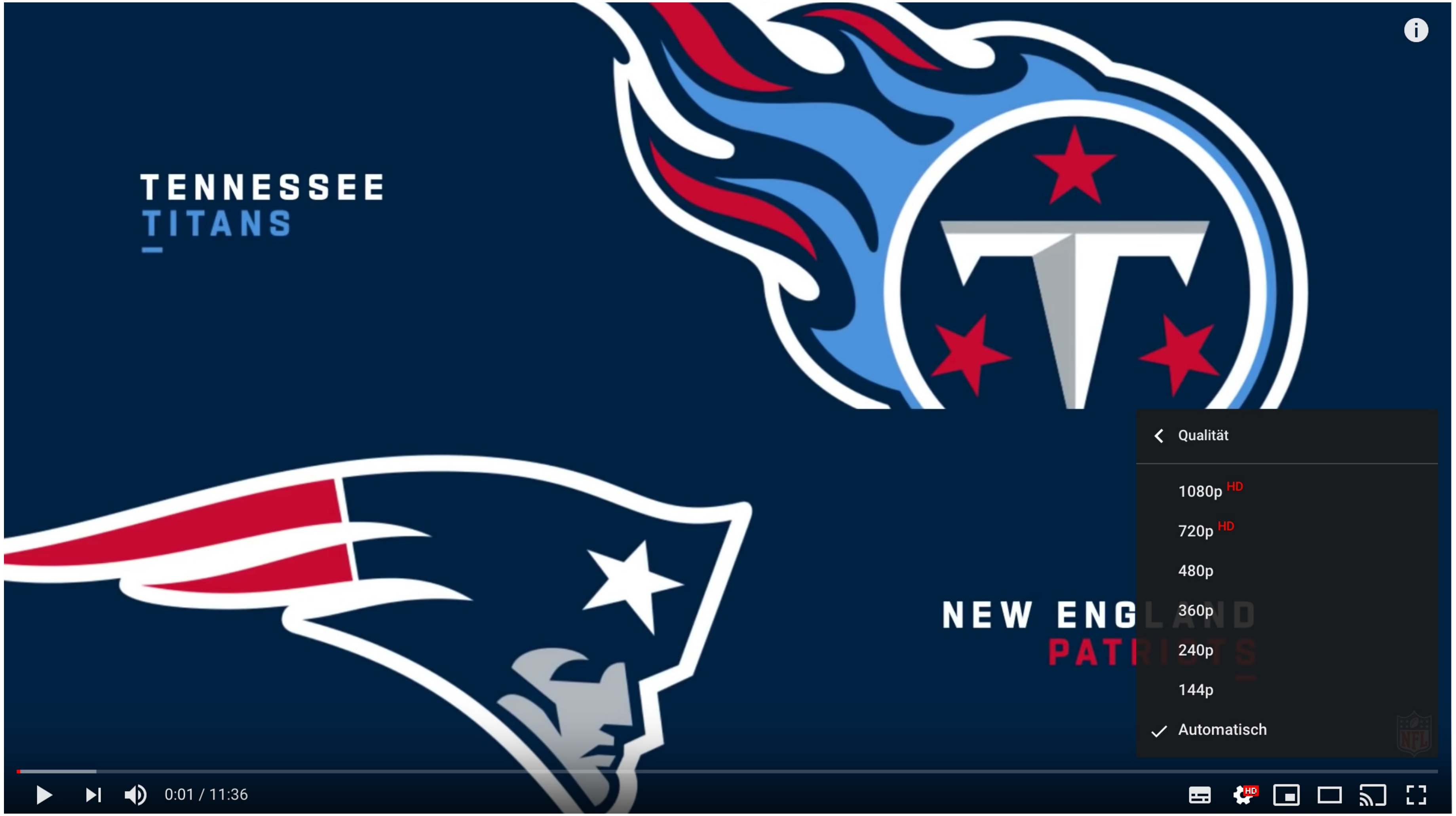
How did I setup this tutorial?

- Questions in between are **always** welcome
- I will show demos in between the different Chapters
- Some chapters will close with a slide on “Recommendations / Best practices / Hints”
- Slides will be shared later

Chapter 01

ABR Streaming

Quality Selection



Qualität

- 1080p HD
- 720p HD
- 480p
- 360p
- 240p
- 144p
- ✓ Automatisch

Adaptive Bitrate Streaming

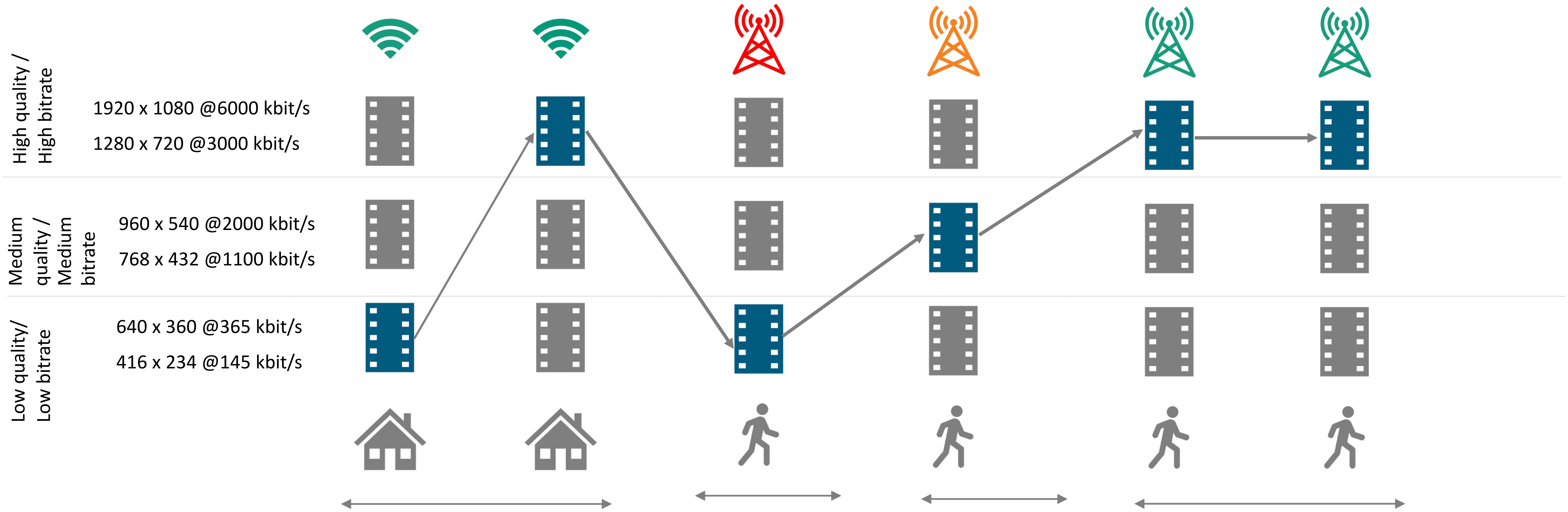
Encoding ladder

Resolution	Bitrate kbit/s	Frame Rate
400x224	417	25
640x360	1219	25
768x432	2189	25
1280x720	3375	50
1920x1080	5825	50
1920x1080	8816	50

Adaptive Bitrate Streaming

Dynamic quality switching

🟦 Single segment with a duration of 2-4 seconds



High quality / High bitrate

1920 x 1080 @6000 kbit/s

1280 x 720 @3000 kbit/s

Medium quality / Medium bitrate

960 x 540 @2000 kbit/s

768 x 432 @1100 kbit/s

Low quality/ Low bitrate

640 x 360 @365 kbit/s

416 x 234 @145 kbit/s

Local WIFI: High bandwidth, fast startup phase

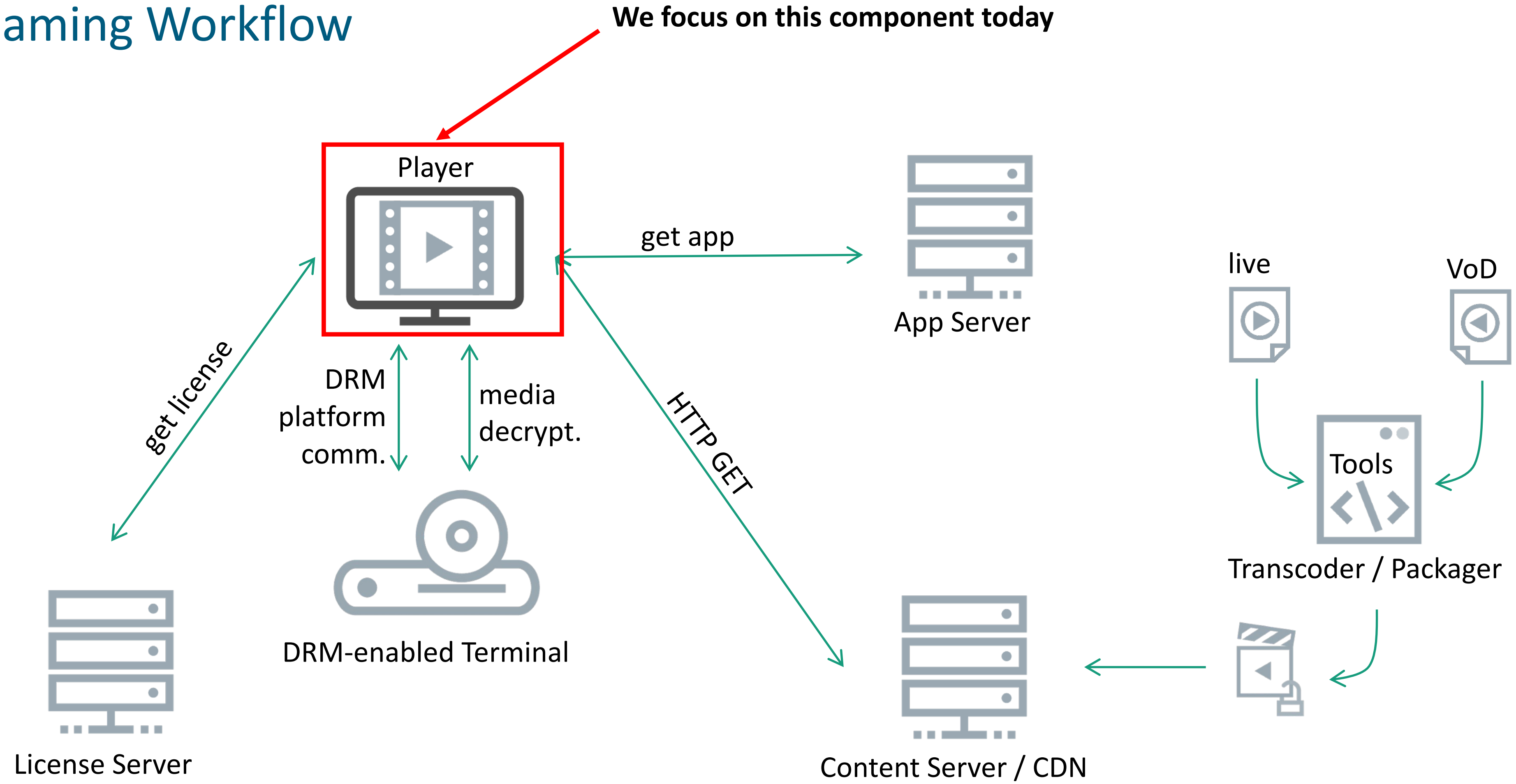
Mobile connection: Low bandwidth, quality is decreased

Mobile connection: Medium bandwidth, quality is increased

Mobile connection: High bandwidth, quality is increased

Adaptive Bitrate Streaming

Media Streaming Workflow



Chapter 02

MSE and EME based playback

Web based players – Web APIs

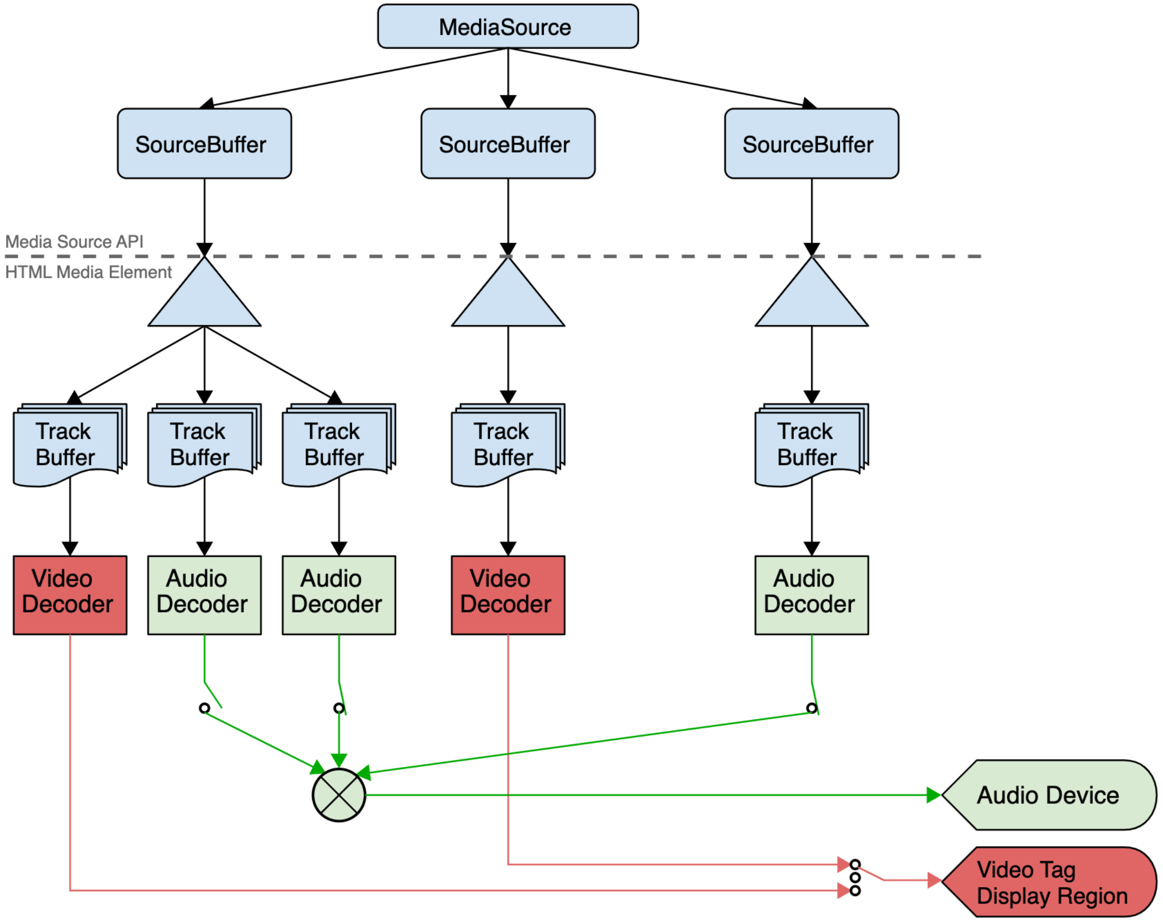
Types of browser-based playback

- **Type 1**
  - Direct playback via the HTML5 video element
    - *<video id="video" controls width=1280 height=720 src=”video.mpd"></video>*
  - No control over the playback and the ABR behavior of the player, more or less a blackbox
  - Examples: HbbTV, Samsung AVPlay, Safari HLS
- **Type 2**
  - HTML5 video element but ABR API to control ABR logic of the player
    - Examples: ?
- **Type 3**
  - HTML5 video element + Media Source Extensions + Encrypted Media Extensions
  - Full control over the playback but complete player logic needs to be implemented
    - Examples: dash.js, hls.js, Shaka Player

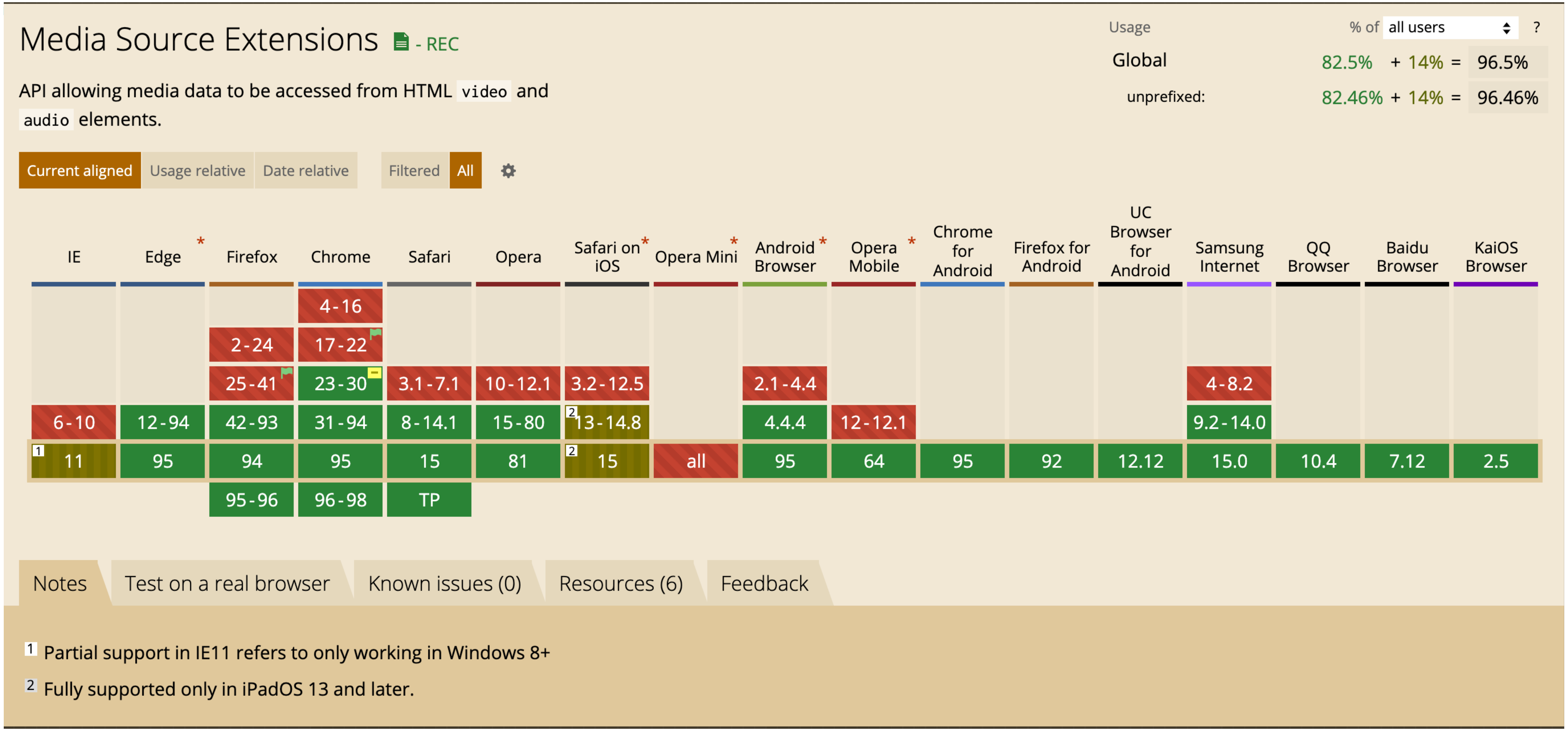
Web based players – Web APIs

W3C Media Source Extensions

- Enables JavaScript clients to append media segments to the HTML5 Video Element
- Defines a MediaSource object that can serve as a source of media data for an HTMLMediaElement.
- MediaSource objects have one or more SourceBuffer objects
- Applications append data segments to the SourceBuffer objects, and can adapt the quality of appended data based on system performance and other factors
- https://w3c.github.io/media-source/



W3C Media Source Extensions - Support



Media Source Extensions - REC

API allowing media data to be accessed from HTML `video` and `audio` elements.

Usage: % of all users ?

Global: 82.5% + 14% = 96.5%

unprefixed: 82.46% + 14% = 96.46%

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
			4-16													
		2-24	17-22													
		25-41	23-30	3.1-7.1	10-12.1	3.2-12.5		2.1-4.4					4-8.2			
6-10	12-94	42-93	31-94	8-14.1	15-80	[2] 13-14.8		4.4.4	12-12.1				9.2-14.0			
[1] 11	95	94	95	15	81	[2] 15	all	95	64	95	92	12.12	15.0	10.4	7.12	2.5
		95-96	96-98	TP												

[1] Partial support in IE11 refers to only working in Windows 8+

[2] Fully supported only in iPadOS 13 and later.

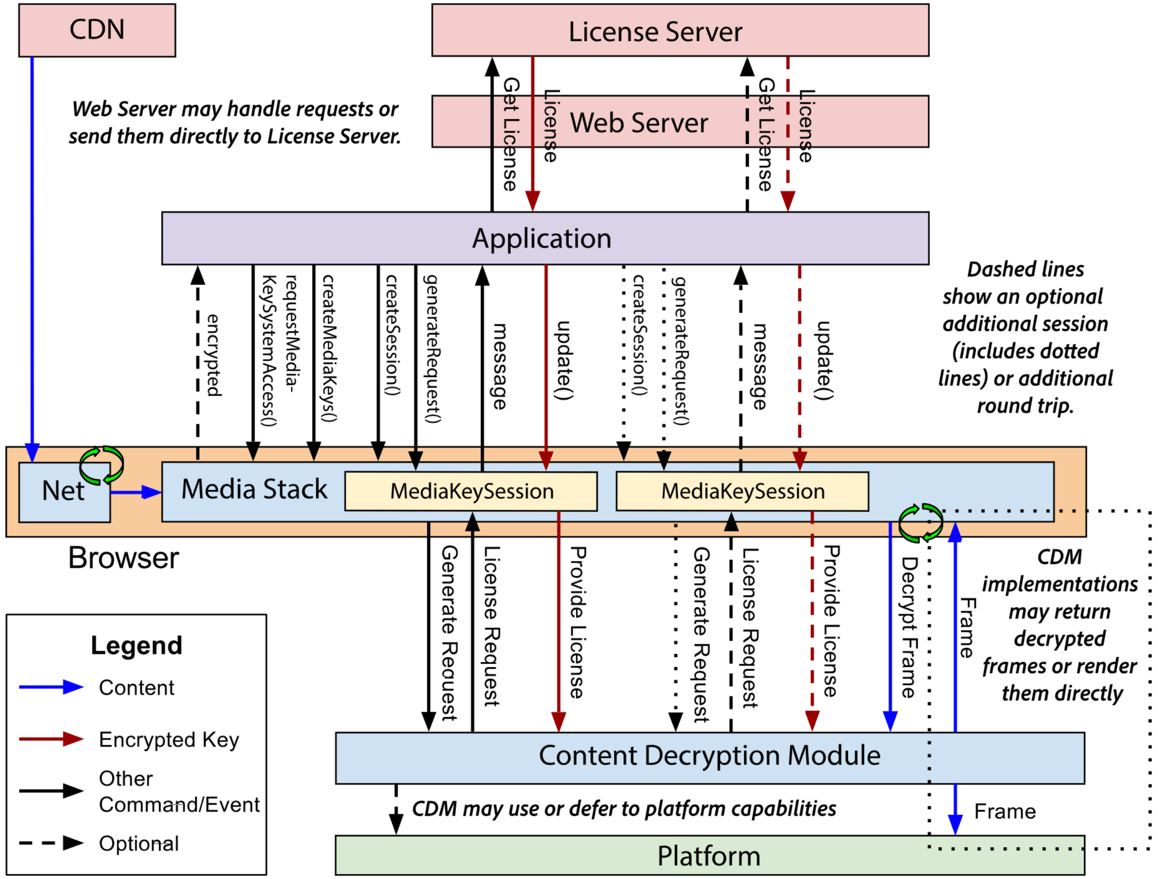
- Source: https://caniuse.com/?search=media%20source%20extensions

Web based players – Web APIs

W3C Encrypted Media Extensions

JavaScript interface between DRM License Server and CDM

- Minor differences across browsers
- Different versions of the EME over the years. Some embedded devices only support outdated EME versions.
- Secure origin and transport / mixed content → requires https at least in Chrome
- https://www.w3.org/TR/encrypted-media/



Web Server may handle requests or send them directly to License Server.

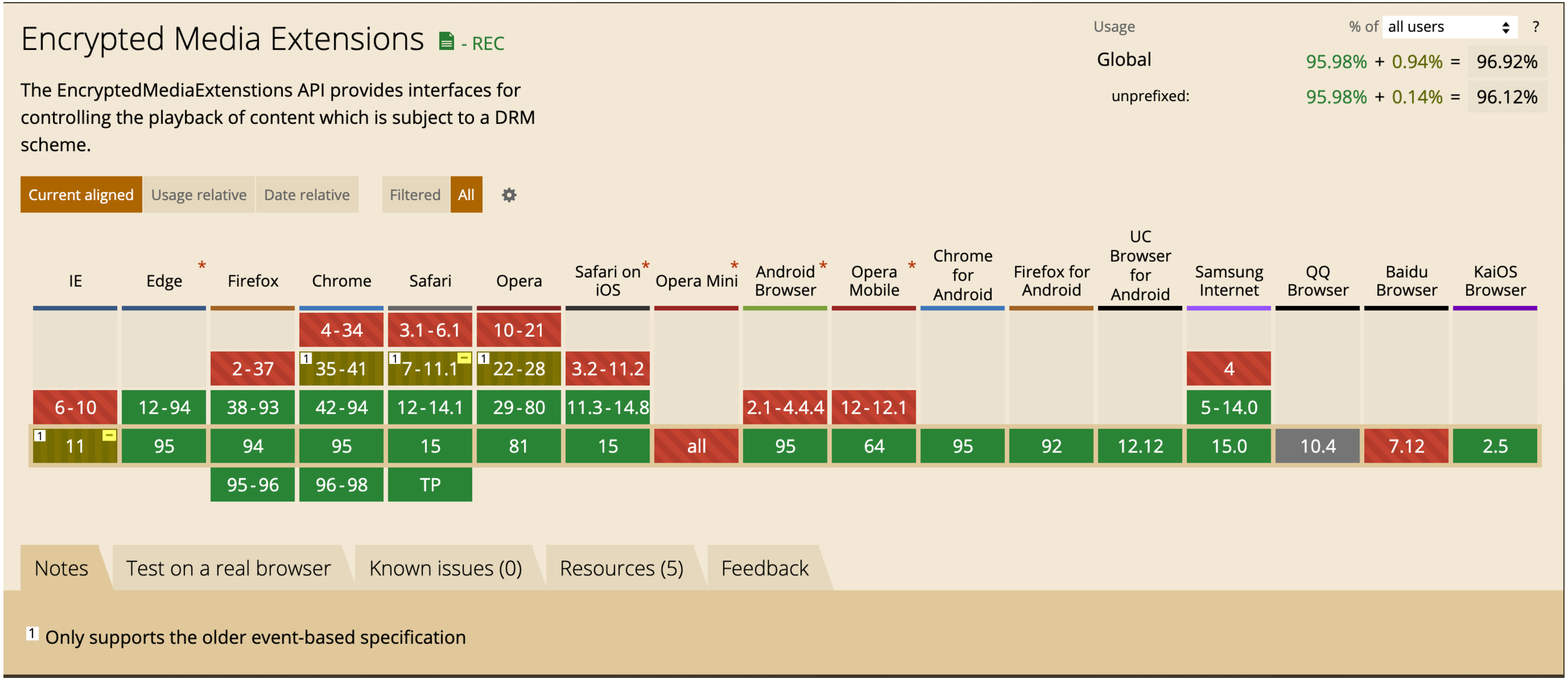
Dashed lines show an optional additional session (includes dotted lines) or additional round trip.

CDM implementations may return decrypted frames or render them directly

CDM may use or defer to platform capabilities

Web based players – Web APIs

W3C Encrypted Media Extensions - Support



Encrypted Media Extensions - REC

Usage | % of all users ?

Global	95.98% + 0.94% =	96.92%
unprefixed:	95.98% + 0.14% =	96.12%

The EncryptedMediaExtenstions API provides interfaces for controlling the playback of content which is subject to a DRM scheme.

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
			4-34	3.1-6.1	10-21											
		2-37	[1] 35-41	[1] 7-11.1 −	[1] 22-28	3.2-11.2							4			
6-10	12-94	38-93	42-94	12-14.1	29-80	11.3-14.8		2.1-4.4.4	12-12.1				5-14.0			
[1] 11 −	95	94	95	15	81	15	all	95	64	95	92	12.12	15.0	10.4	7.12	2.5
		95-96	96-98	TP												

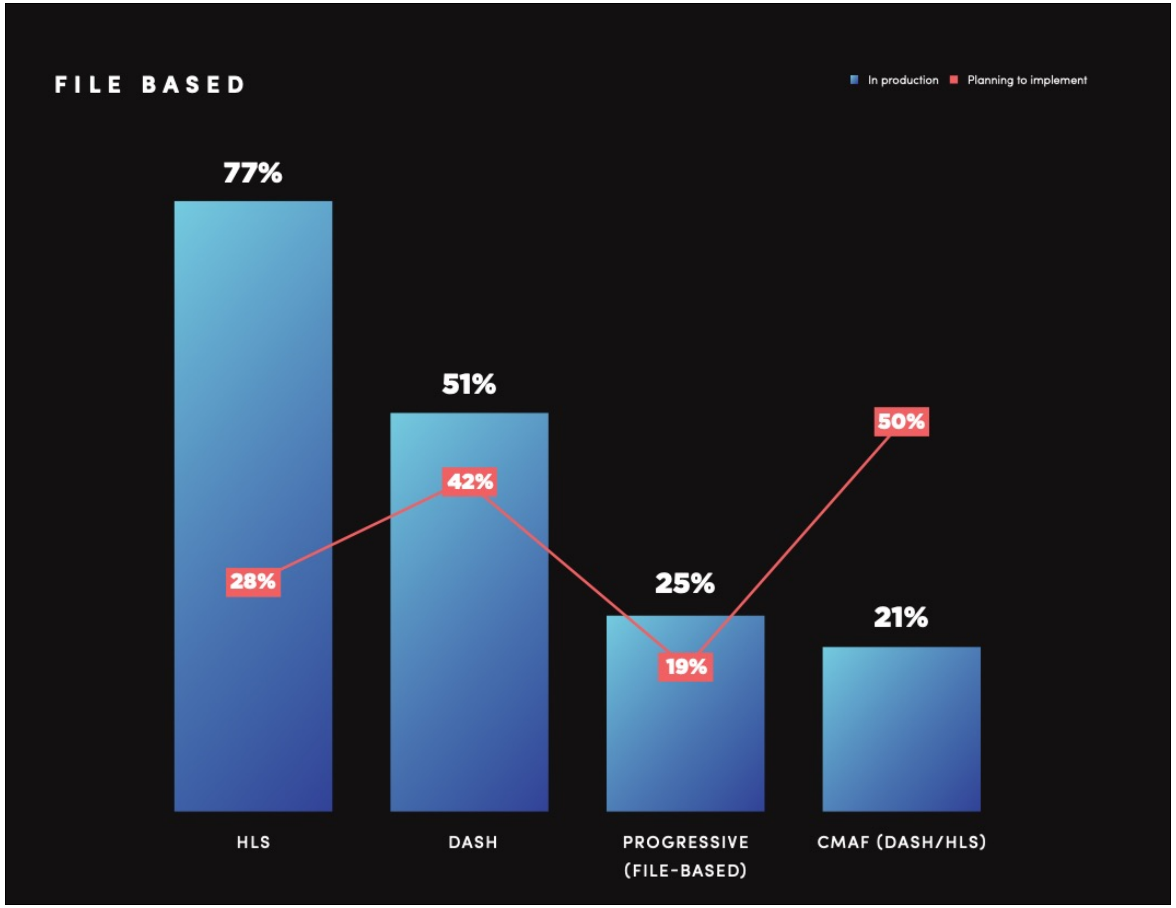
[1] Only supports the older event-based specification

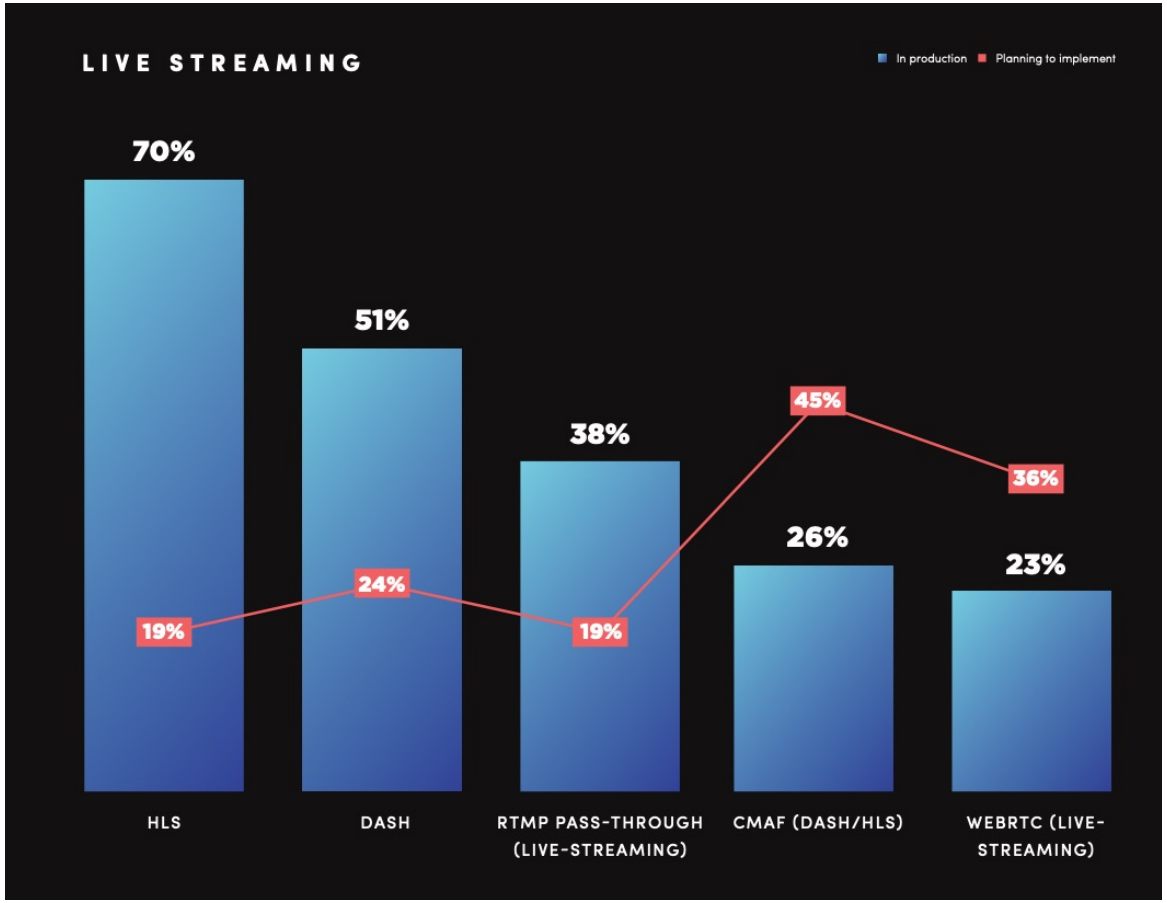
- Source: https://caniuse.com/?search=encrypted%20media%20extensions

Chapter 03

Adaptive Streaming Formats – MPEG-DASH

Live and VoD format usage in 2022





- Source: https://bitmovin.com/wp-content/uploads/2022/12/bitmovin-6th-video-developer-report-2022-2023.pdf

MPEG-DASH

Dynamic Adaptive Streaming over HTTP (DASH) - ISO/IEC 23009

- **Part 1: Media presentation description and segment formats**
- Part 2: Conformance and reference software
- Part 3: Implementation guidelines
- Part 4: Segment encryption and authentication
- Part 5: Server and network assisted DASH (SAND)
- Part 6: DASH with Server Push and Web Sockets

- Part 1 is freely available here: https://standards.iso.org/ittf/PubliclyAvailableStandards/c083314_ISO_IEC%2023009-1_2022(en).zip

- Different profiles: DASH-IF, DVB-DASH, HbbTV, CTA-WAVE etc.
  - E.g. "urn:mpeg:dash:profile:isoff-live:2011,urn:com:dashif:dash264"

Adaptive Streaming Formats – MPEG-DASH

DASH-IF Interoperability Guidelines

- Goal: Create a baseline recommendation that everyone could use to build interoperable products and services without painful integration
- With version 5 of DASH-IF Interoperability Guidelines, DASH-IF decided to introduce different parts that each address specific aspects of DASH-based service delivery. Each part is developed and updated within its own timescale
- Download here: https://dashif.org/guidelines/iop-v5/
- Also very useful: https://dashif-documents.azurewebsites.net/Guidelines-TimingModel/master/Guidelines-TimingModel.html

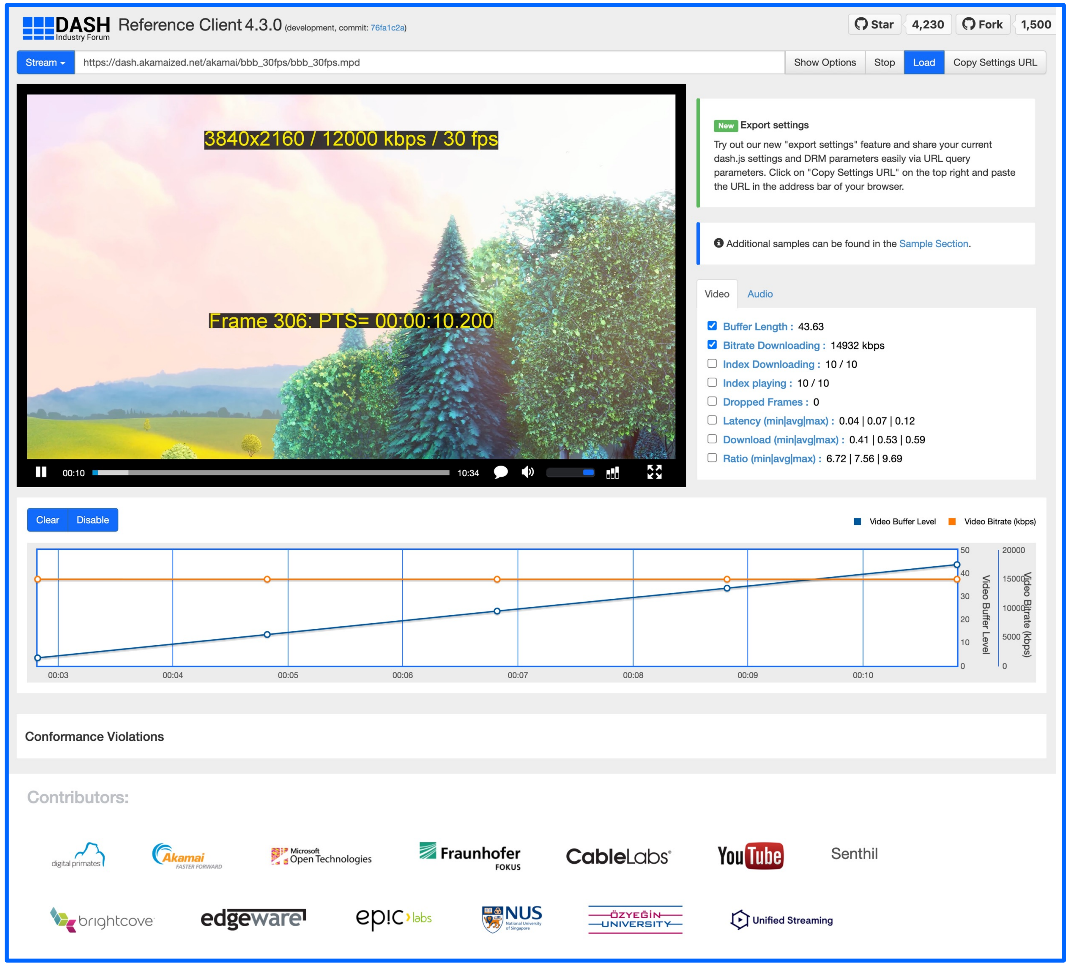
Chapter 04

dash.js – Overview

dash.js – Overview

Overview & Status

- dash.js is the official reference player by the DASH Industry Forum for playback of MPEG-DASH content
- Maintained by Fraunhofer FOKUS, community driven development
- Open-source project on Github - https://github.com/Dash-Industry-Forum/dash.js/ , last released version 4.7.0
- Written in JavaScript uses the W3C Media Source Extensions (MSE) and Encrypted Media Extensions (EME)
- Works on all MSE and EME based platforms including Desktop browsers, smartphones, SmartTVs, Set-Top Boxes.
- Various features including flexible ABR logic, multiperiod, DRM support, MPD patching, Gap handling, CMCD, CMAF low latency support, support for various subtitle formats (TTML, IMSC1, WebVTT) and many more.



https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd

New Export settings

Try out our new "export settings" feature and share your current dash.js settings and DRM parameters easily via URL query parameters. Click on "Copy Settings URL" on the top right and paste the URL in the address bar of your browser.

Additional samples can be found in the Sample Section.

☑ Buffer Length : 43.63
☑ Bitrate Downloading : 14932 kbps
☐ Index Downloading : 10 / 10
☐ Index playing : 10 / 10
☐ Dropped Frames : 0
☐ Latency (min|avg|max) : 0.04 | 0.07 | 0.12
☐ Download (min|avg|max) : 0.41 | 0.53 | 0.59
☐ Ratio (min|avg|max) : 6.72 | 7.56 | 9.69

https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html

Application areas

Reference platform

- Implements latest features from DASH-IF IOP guidelines and ISO/IEC specification.
- Used by other organizations in their reference implementations
  - CTA-WAVE
  - DVB-I
  - HbbTV
  - 5G-MAG

Industry

- Used in production for instance by BBC, Deutsche Telekom, Orange
- Used to compare behavior of commercial players against reference player

Research

- Used for research purposes, for instance to test and compare new ABR algorithms (Twitch challenge)
- Evaluate new features such as MPD patching and CMCD

Numbers

GitHub

- 51 releases
- > 4.700 stars
- 258 watchers
- > 1.600 forks
- Used by over 2.300 other projects
- 172 contributors

npm

- 96 dependents
- 50.000 – 80.000 downloads a week
- 2.090.885 downloads in 2022

- Different DASH-IF calls every week
- Monthly developer calls
- Discussions on
  - Slack (1708 members)
  - Github
  - Google Groups (1223 members)

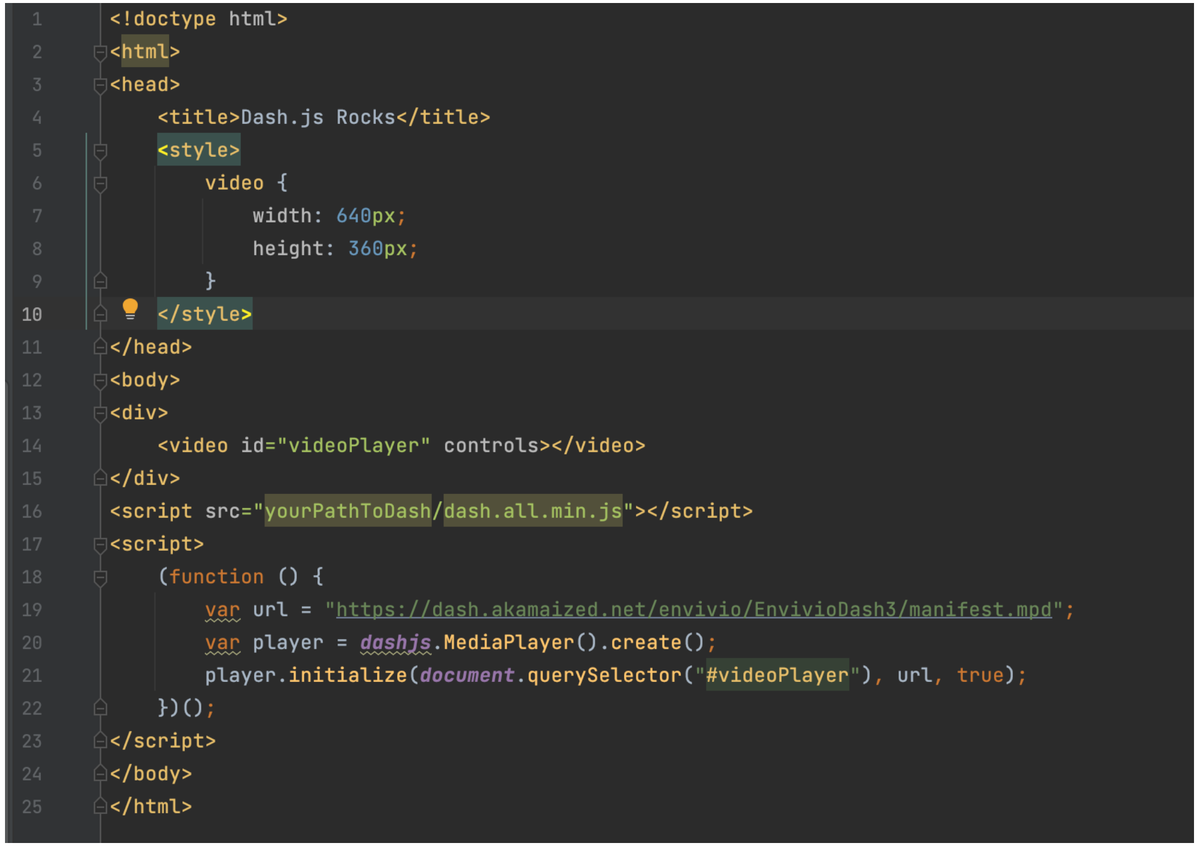
dash.js - Overview

Important links

- Github project: https://github.com/Dash-Industry-Forum/dash.js
- Reference client: https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html
- Samples: https://reference.dashif.org/dash.js/nightly/samples/index.html
- Wiki: https://github.com/Dash-Industry-Forum/dash.js/wiki
- API documentation: http://cdn.dashjs.org/latest/jsdoc/module-MediaPlayer.html
- Slack Channel: https://dashif-slack.azurewebsites.net/
- Google Groups: https://groups.google.com/g/dashjs
- How to contribute: https://github.com/Dash-Industry-Forum/dash.js/blob/development/CONTRIBUTING.md

Hands-On – Getting started

Try it out yourself:
http://reference.dashif.org/dash.js/nightly/samples/getting-started/manual-load-single-video.html



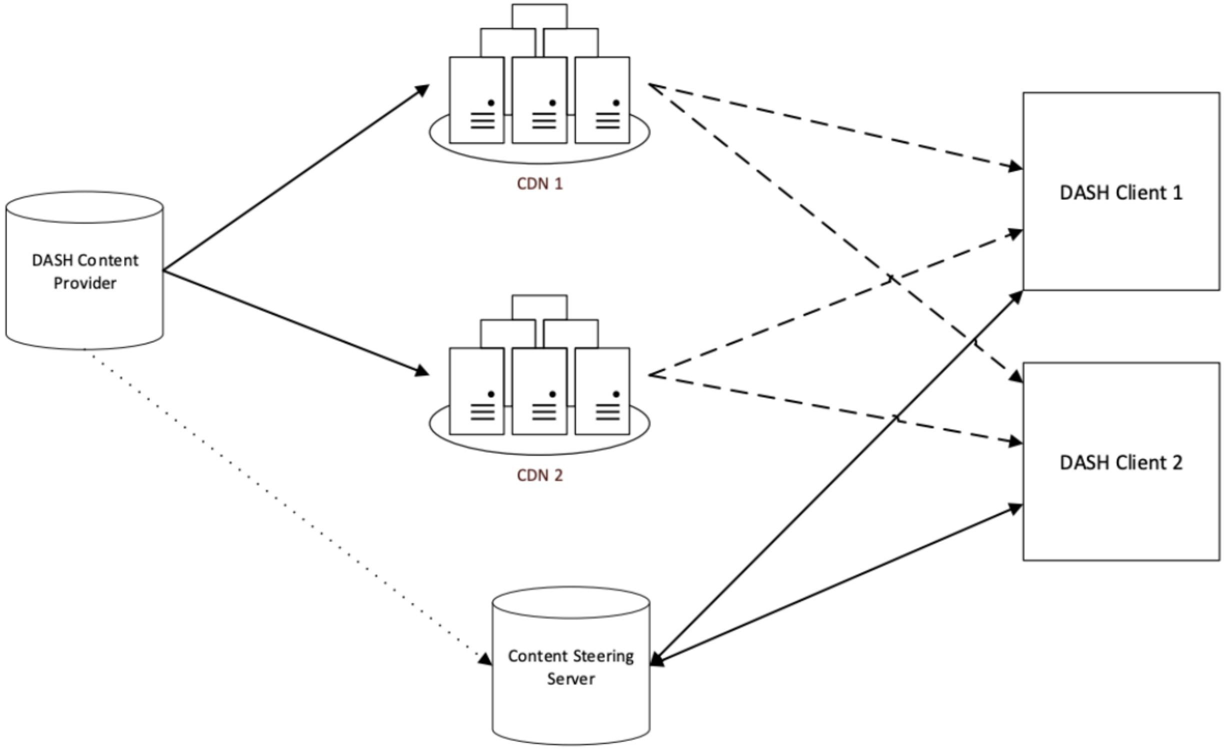
```
<!doctype html>
<html>
<head>
    <title>Dash.js Rocks</title>
    <style>
        video {
            width: 640px;
            height: 360px;
        }
    </style>
</head>
<body>
<div>
    <video id="videoPlayer" controls></video>
</div>
<script src="yourPathToDash/dash.all.min.js"></script>
<script>
    (function () {
        var url = "https://dash.akamaized.net/envivio/EnvivioDash3/manifest.mpd";
        var player = dashjs.MediaPlayer().create();
        player.initialize(document.querySelector("#videoPlayer"), url, true);
    })();
</script>
</body>
</html>
```

Chapter 05

dash.js Features

Content Steering

- Content steering describes a deterministic capability for a content distributor to switch the content source that a player uses either at start-up or midstream by means of a remote steering service
- Introduced in the 2nd edition of the HLS specification, DASH-IF has taken the task to define a corresponding DASH specification
- Adds new <ContentSteering> element to the MPD
- <BaseURL> elements contain „serviceLocation“ attribute that can be used as an identifier
- Steering Server returns a „PATHWAY_PRIORITY“ list
- New elements can be synthesized with „PATHWAY_CLONES“

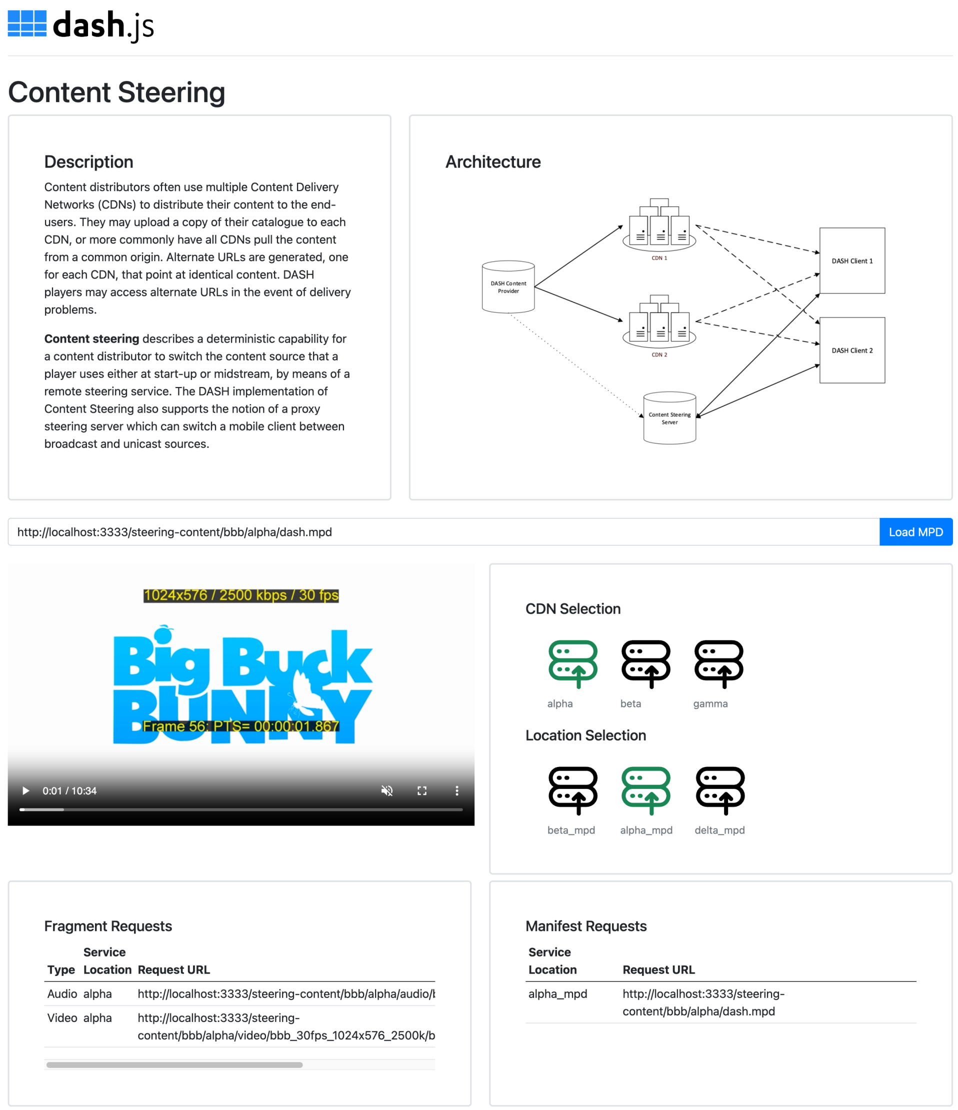


- Try it out yourself (requires a steering server): https://reference.dashif.org/dash.js/nightly/samples/advanced/content-steering.html

Content Steering

dash.js demo

Try it out yourself:
https://reference.dashif.org/dash.js/nightly/samples/advanced/content-steering.html



dash.js

Content Steering

Description

Content distributors often use multiple Content Delivery Networks (CDNs) to distribute their content to the end-users. They may upload a copy of their catalogue to each CDN, or more commonly have all CDNs pull the content from a common origin. Alternate URLs are generated, one for each CDN, that point at identical content. DASH players may access alternate URLs in the event of delivery problems.

Content steering describes a deterministic capability for a content distributor to switch the content source that a player uses either at start-up or midstream, by means of a remote steering service. The DASH implementation of Content Steering also supports the notion of a proxy steering server which can switch a mobile client between broadcast and unicast sources.

Architecture

http://localhost:3333/steering-content/bbb/alpha/dash.mpd

CDN Selection

Location Selection

Fragment Requests

Type	Service Location	Request URL
Audio	alpha	http://localhost:3333/steering-content/bbb/alpha/audio/l
Video	alpha	http://localhost:3333/steering-content/bbb/alpha/video/bbb_30fps_1024x576_2500k/b

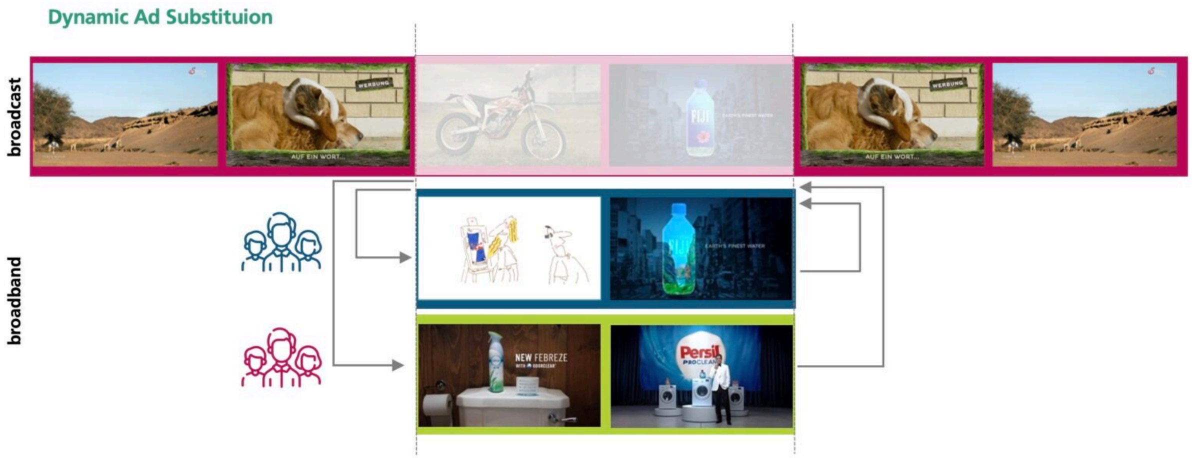
Manifest Requests

Service Location	Request URL
alpha_mpd	http://localhost:3333/steering-content/bbb/alpha/dash.mpd

dash.js Features

Preload

- Some platforms like HbbTV terminals have only a single decoder. It is not possible to initialize MSE based playback while the broadcast content is rendered
- To support Broadcast-Broadband ad insertion on HbbTV terminals segments should be prebuffered for a seamless transition between main content (broadcast) and ad content (broadband)
- Solution: Virtual buffer that is emptied once MSE is attached to video element



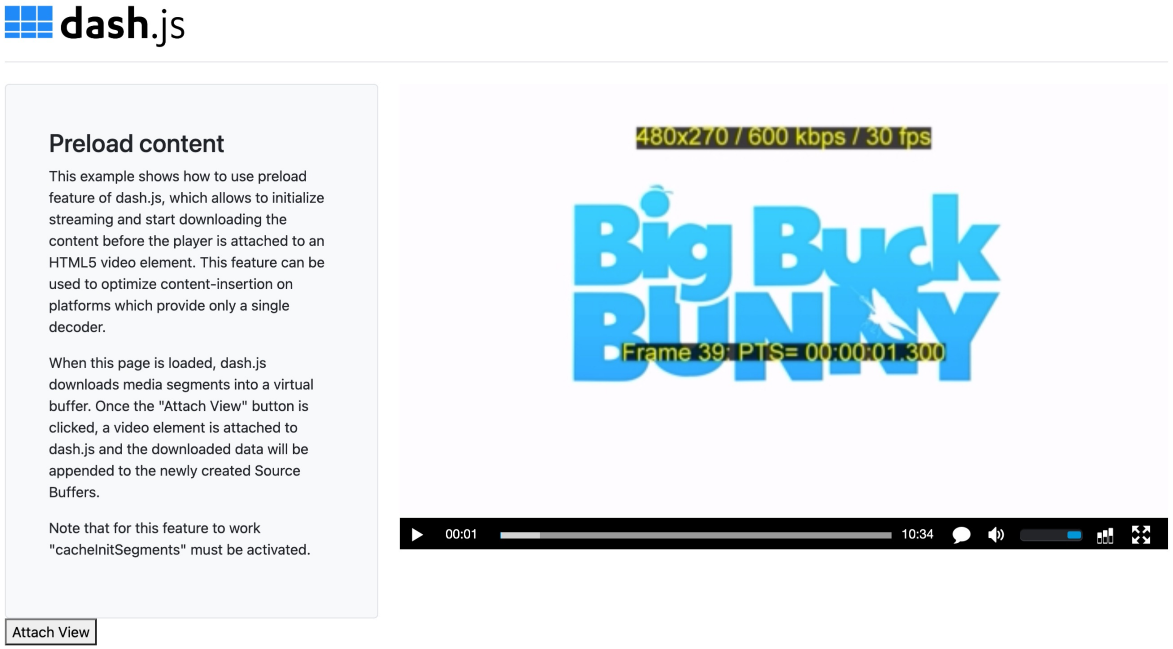
```
function init() {
    var url = 'https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd';

    player = dashjs.MediaPlayer().create();
    player.initialize(null, url, true);
    player.updateSettings({
        debug: {logLevel: 5},
        streaming: {cacheInitSegments: true}
    });
    player.preload();
}
```

- Try it out yourself: https://reference.dashif.org/dash.js/nightly/samples/advanced/preload.html

Preload

dash.js demo

Try it out yourself: https://reference.dashif.org/dash.js/nightly/samples/advanced/preload.html



Preload content

This example shows how to use preload feature of dash.js, which allows to initialize streaming and start downloading the content before the player is attached to an HTML5 video element. This feature can be used to optimize content-insertion on platforms which provide only a single decoder.

When this page is loaded, dash.js downloads media segments into a virtual buffer. Once the "Attach View" button is clicked, a video element is attached to dash.js and the downloaded data will be appended to the newly created Source Buffers.

Note that for this feature to work "cacheInitSegments" must be activated.

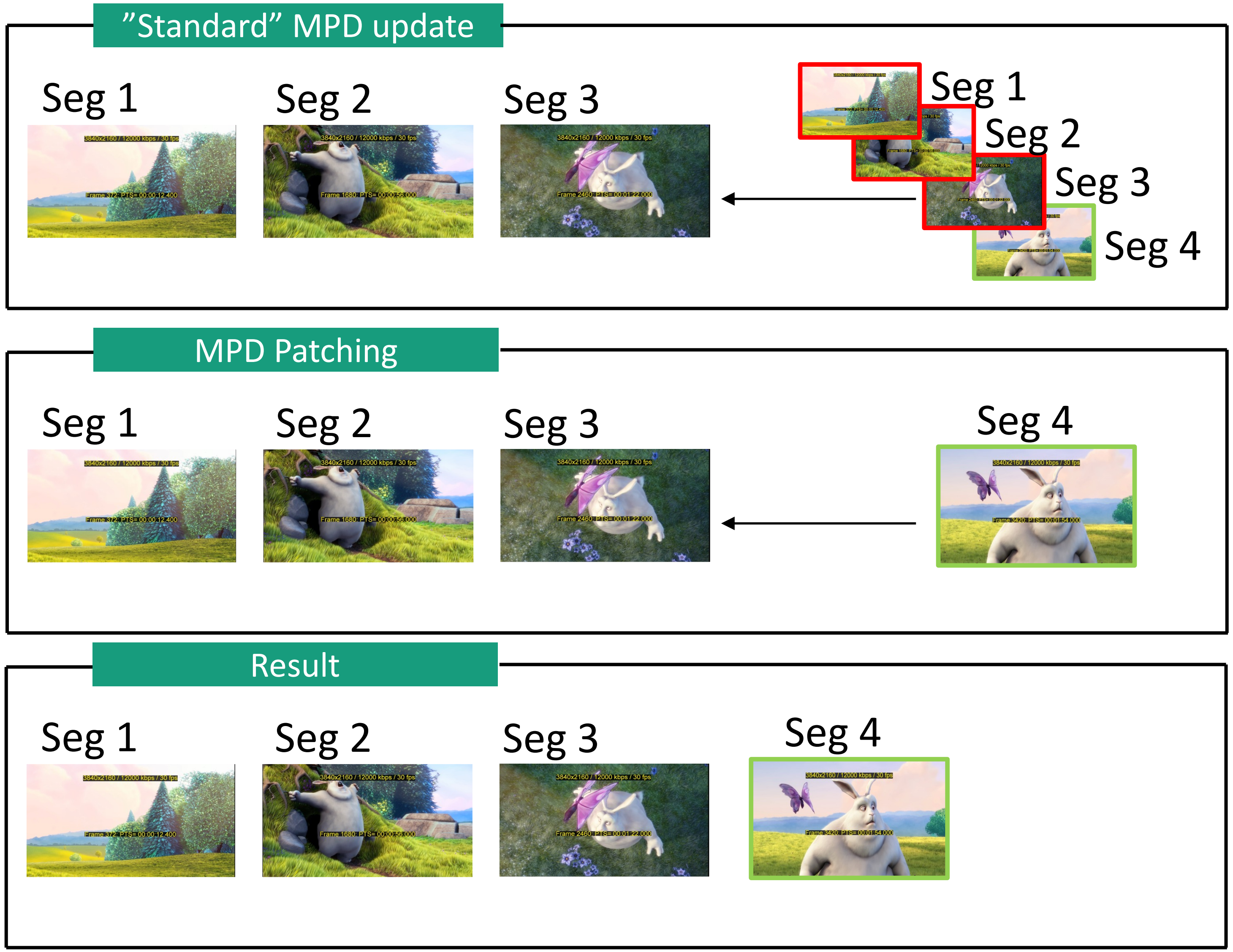
dash.js Features

MPD Patching

- Added in 5th edition of MPEG-DASH
- Although some parts of the MPD can change between two consecutive MPD updates, most parts of it remain unchanged.
- Idea: Provide only mandatory MPD information to the client.
- Updates to the MPD are provided through MPD patches. MPD patches only contain new information, such as additional media segment
- Allows addition, removal and change of information in the manifest

✓ Reduced traffic

✓ Reduced parsing time on the client side



Common Media Client Data

- CTA-5004 - Common Media Client Data (CMCD) defines data that is collected by the media player and is sent as a custom HTTP header or query parameter alongside each object request to a CDN
- Enables
  - Log analysis
  - Quality of service monitoring
  - Prioritization of clients
  - Cross correlation of performance problems with specific devices and platforms
  - Improved edge caching
- dash.js
  - allows whitelisting of the parameters
  - Dispatches all the CMCD data via events to be used for custom metric reporting
- Try it out: https://tinyurl.com/cmcd-dashjs

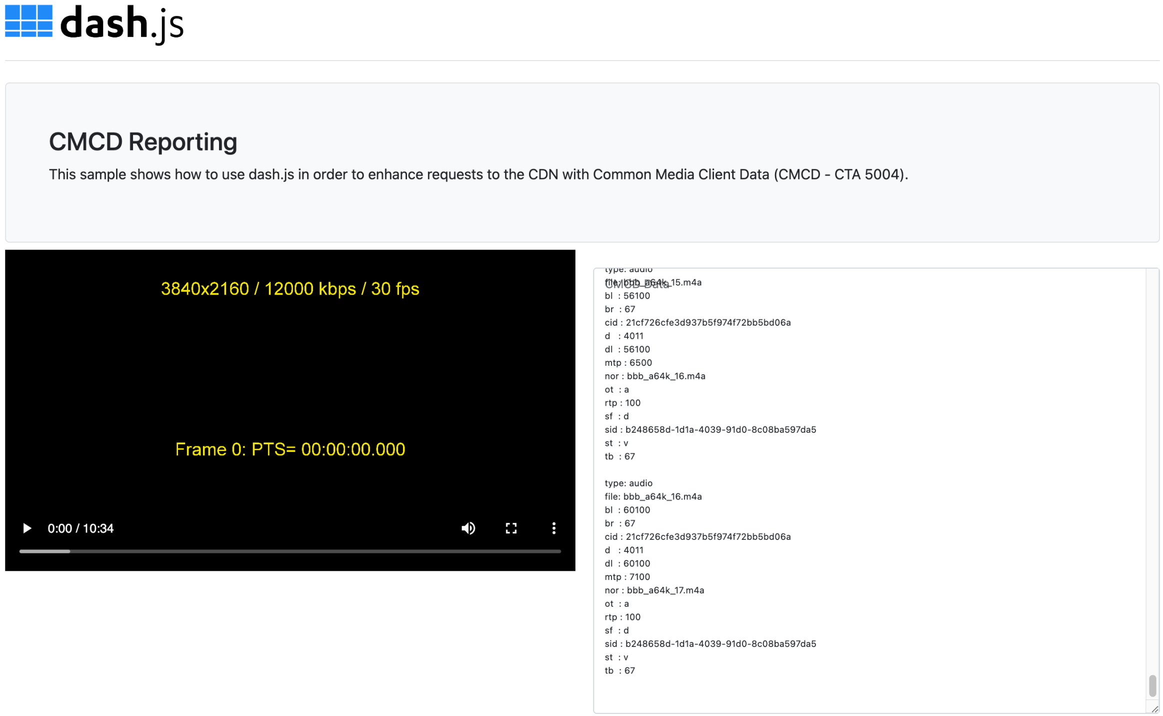
CMCD parameters

- bl: Buffer length
- br: Encoded bitrate
- bs: Buffer starvation
- cid: Content ID
- d: Object duration
- dl: Deadline
- mtp: Measured throughput
- nor: Next object request
- nrr: Next range request
- ot: Object type
- pr: Playback rate
- rtp: Requested maximum throughput
- sf: Streaming format
- sid: Session ID
- st: Stream Type
- su: Startup
- tb: top bitrate

Common Media Client Data

dash.js demo

Try it out yourself:
https://reference.dashif.org/dash.js/nightly/samples/advanced/cmcd.html



dash.js

CMCD Reporting

This sample shows how to use dash.js in order to enhance requests to the CDN with Common Media Client Data (CMCD - CTA 5004).

```
type: audio
file: bbb_a64k_15.m4a
bl  : 56100
br  : 67
cid : 21cf726cfe3d937b5f974f72bb5bd06a
d   : 4011
dl  : 56100
mtp : 6500
nor : bbb_a64k_16.m4a
ot  : a
rtp : 100
sf  : d
sid : b248658d-1d1a-4039-91d0-8c08ba597da5
st  : v
tb  : 67

type: audio
file: bbb_a64k_16.m4a
bl  : 60100
br  : 67
cid : 21cf726cfe3d937b5f974f72bb5bd06a
d   : 4011
dl  : 60100
mtp : 7100
nor : bbb_a64k_17.m4a
ot  : a
rtp : 100
sf  : d
sid : b248658d-1d1a-4039-91d0-8c08ba597da5
st  : v
tb  : 67
```

Common Media Server Data

- CTA-5006 – Common Media Server Data defines structure for data transmitted in the response to a request from a media player for an HTTP adaptive streaming media object.
- The response usually originates at an origin server and is then propagated through a series of intermediaries to the player.
- Examples:
  - Edge servers can provide information about throughput or the cache status of objects.
  - Coordinate multiple clients that are competing for the available bandwidth

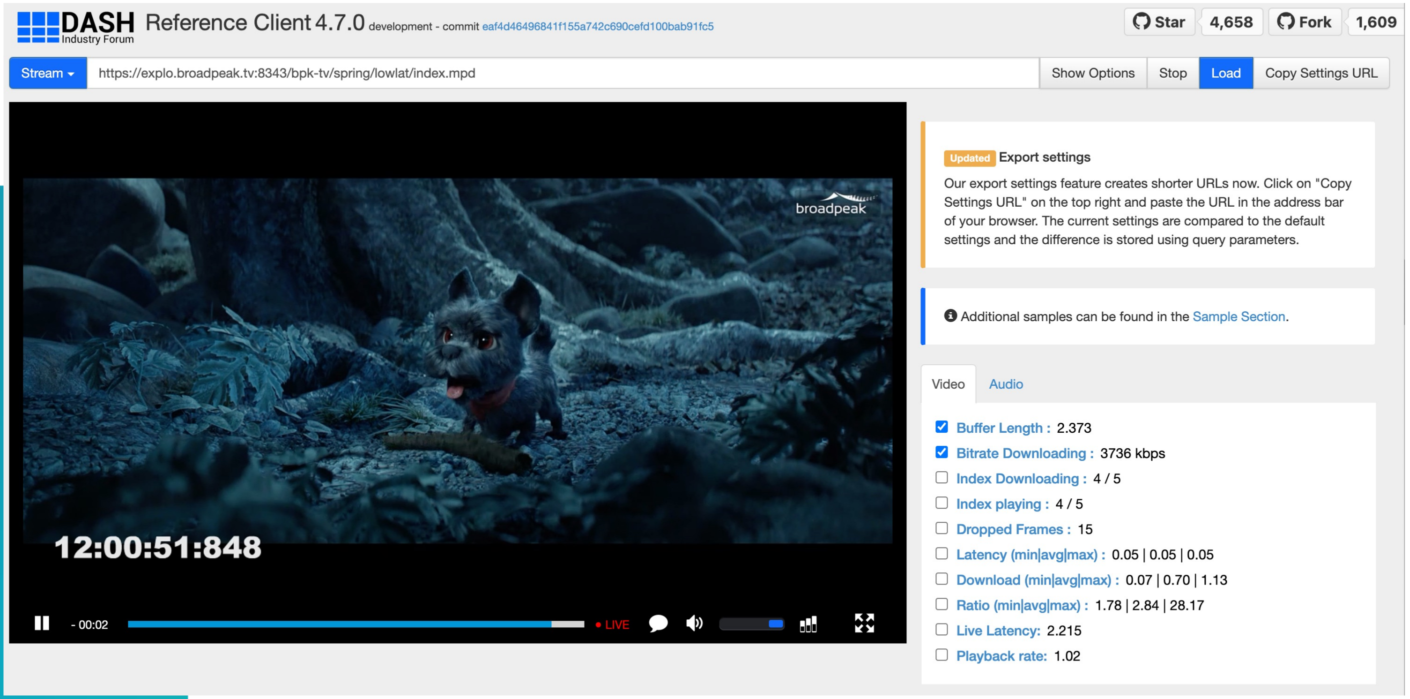
CMSD parameters

- at: Availability Time
- du:Duress
- br: Encoded Bitrate
- etp: Estimated Throughput
- ht: Held time
- n: Intermediary identifier
- mb: Max suggested bitrate
- nor: Next object response
- nrr: Next range response
- d: Object duration
- ot: Object type
- rd: Response delay
- rtt: Round trip time
- su: Startup
- st: Stream Type
- sf: Streaming format
- v: Version

Common Media Server Data

dash.js demo

Try it out yourself:
https://shorturl.at/jsxW7



DASH Industry Forum Reference Client 4.7.0 development - commit eaf4d46496841f155a742c690cefd100bab91fc5

https://explo.broadpeak.tv:8343/bpk-tv/spring/lowlat/index.mpd

Updated Export settings

Our export settings feature creates shorter URLs now. Click on "Copy Settings URL" on the top right and paste the URL in the address bar of your browser. The current settings are compared to the default settings and the difference is stored using query parameters.

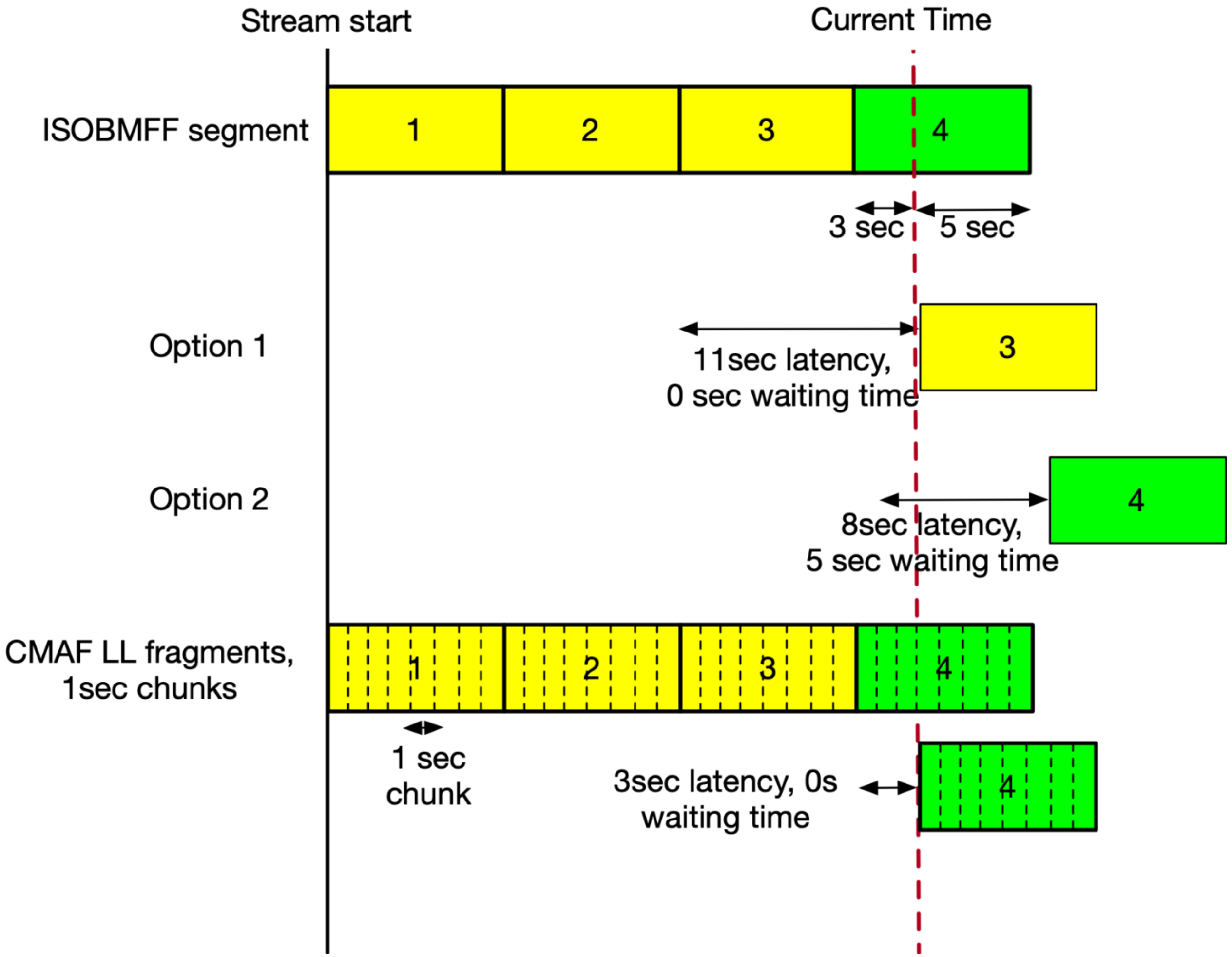
Additional samples can be found in the Sample Section.

Video Audio

- ☑ Buffer Length : 2.373
- ☑ Bitrate Downloading : 3736 kbps
- ☐ Index Downloading : 4 / 5
- ☐ Index playing : 4 / 5
- ☐ Dropped Frames : 15
- ☐ Latency (min|avg|max) : 0.05 | 0.05 | 0.05
- ☐ Download (min|avg|max) : 0.07 | 0.70 | 1.13
- ☐ Ratio (min|avg|max) : 1.78 | 2.84 | 28.17
- ☐ Live Latency: 2.215
- ☐ Playback rate: 1.02

Low Latency Streaming

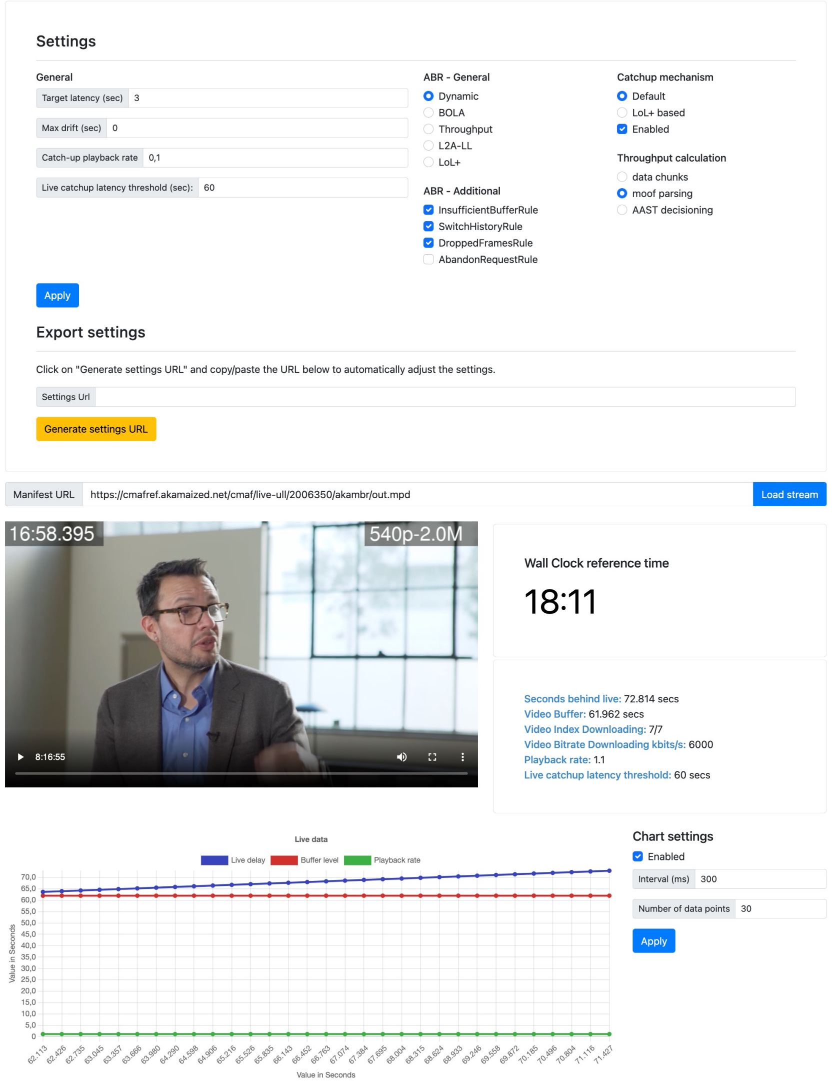
- Key concepts:
  - HTTP/1.1 chunked transfer encoding (CTE)
  - CMAF chunks
  - Adjustment of playback rate to maintain consistent live edge
- Specified in:
  - Section 10.20 of the DVB-DASHv.1.3.1 spec from February 2020
  - DASH-IF IOP v.5 – Low Latency Modes for DASH in March 2020
- Related DASH specific attributes
  - *availabilityTimeOffset*
  - *availabilityTimeComplete*
  - *ServiceDescription*
  - *UTCTiming*
  - *ProducerReferenceTime*



Low Latency Streaming

dash.js demo

Try it out yourself:
https://reference.dashif.org/dash.js/nightly/samples/low-latency/testplayer/testplayer.html



Settings

General

Target latency (sec): 3

Max drift (sec): 0

Catch-up playback rate: 0,1

Live catchup latency threshold (sec): 60

ABR - General

- ● Dynamic
- ○ BOLA
- ○ Throughput
- ○ L2A-LL
- ○ LoL+

ABR - Additional

- ☑ InsufficientBufferRule
- ☑ SwitchHistoryRule
- ☑ DroppedFramesRule
- □ AbandonRequestRule

Catchup mechanism

- ● Default
- ○ LoL+ based
- ☑ Enabled

Throughput calculation

- ○ data chunks
- ● moof parsing
- ○ AAST decisioning

Apply

Export settings

Click on "Generate settings URL" and copy/paste the URL below to automatically adjust the settings.

Settings Url:

Generate settings URL

Manifest URL: https://cmafref.akamaized.net/cmaf/live-ull/2006350/akambr/out.mpd

Load stream

Wall Clock reference time

18:11

Seconds behind live: 72.814 secs
Video Buffer: 61.962 secs
Video Index Downloading: 7/7
Video Bitrate Downloading kbits/s: 6000
Playback rate: 1.1
Live catchup latency threshold: 60 secs

Chart settings

☑ Enabled

Interval (ms): 300

Number of data points: 30

Apply

UTC Timing Synchronization

- During playback of dynamic presentations, a wall clock is used as the timing reference for DASH client decisions.
- This is a synchronized clock shared by the DASH client and service
- The reference clock is defined in the *<UTCTiming>* element in the MPD
- dash.js 4.x dynamically adjusts the interval between synchronization requests depending on the drift between two consecutive attempts.

UTC Timing Synchronization in dash.js

Parameter	Description	Value
background Attempts	Number of synchronization attempts to perform in the background after an initial synchronization request has been done	2
timeBetweenSync Attempts	The time in seconds between two consecutive sync attempts. Note: This value is used as an initial starting value and is adjusted during playback based on the drift between two consecutive synchronization attempts.	30
maximumTime BetweenSync Attempts	The maximum time in seconds between two consecutive sync attempts.	600
minimumTime BetweenSync Attempts	The minimum time in seconds between two consecutive sync attempts	2
timeBetweenSyncAttemptsAdjustmentFactor	The factor used to multiply or divide the timeBetweenSyncAttempts parameter after a sync	2
maximum allowedDrift	The maximum allowed drift specified in milliseconds between two consecutive synchronization attempts	100

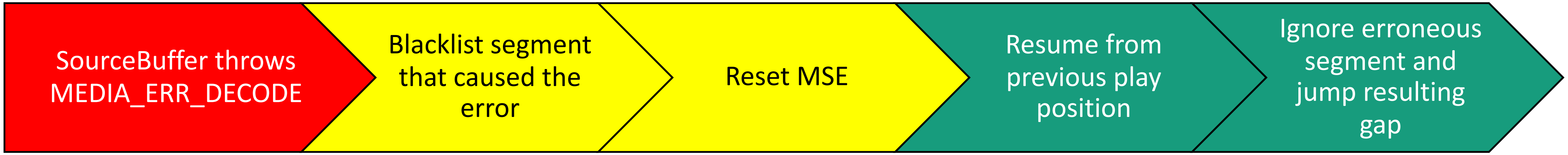
dash.js Features

Recovering from MSE errors

- A decode error for a specific segment should not lead to a complete shutdown of the player
- Idea: Reset MSE and resume playback

Error recovering in dash.js

Parameter	Description	Default
recoverAttempts. mediaErrorDecode	Defines the maximum number of recover attempts for decode errors	5



SourceBuffer throws MEDIA_ERR_DECODE

Blacklist segment that caused the error

Reset MSE

Resume from previous play position

Ignore erroneous segment and jump resulting gap

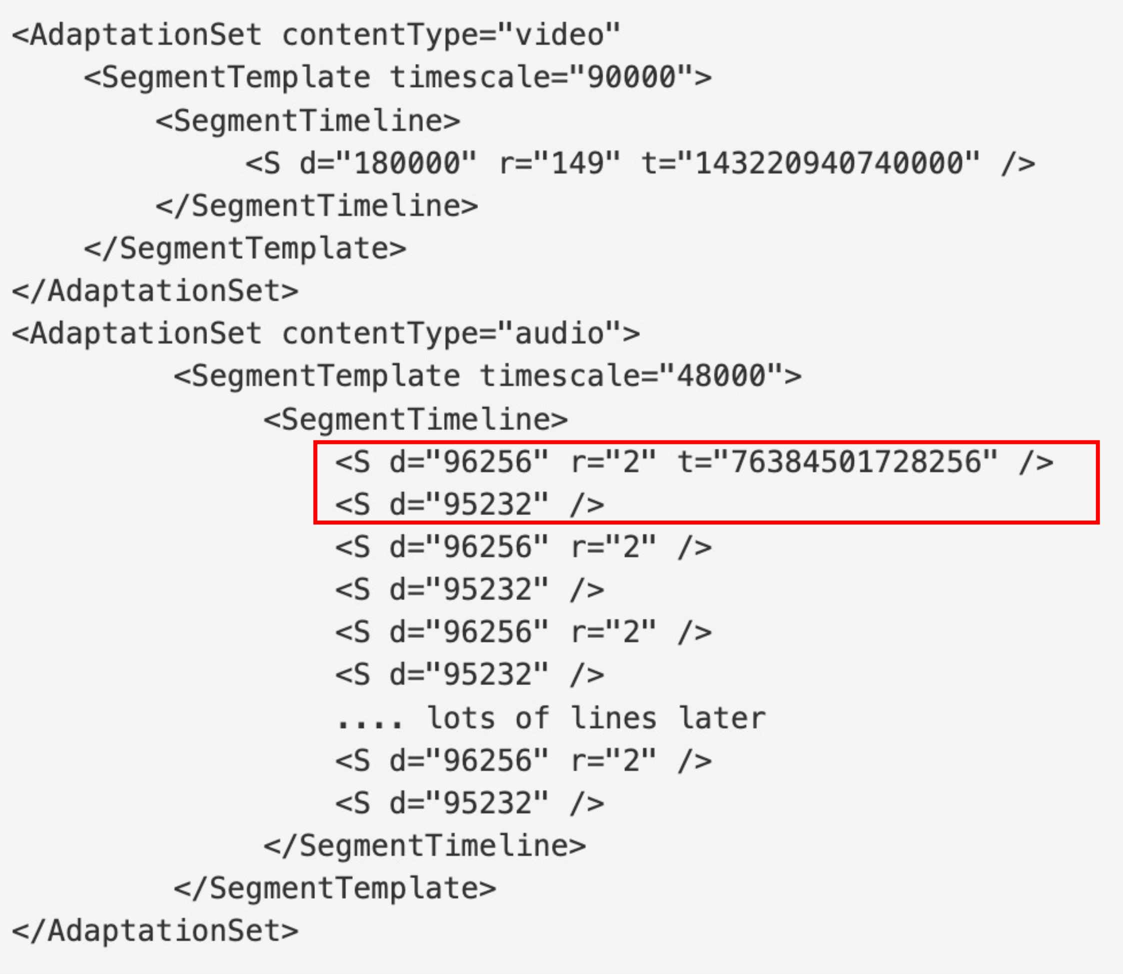
Advanced Topics

Segment Alignment

- Misalignments are mainly a result of segment durations that do not match sampling rate and fixed number of audio frames per packet size.
- Unaligned media segments can lead to
  - Large manifest files
  - Player performance problems (long parsing duration)

Aligned segments: 25fps with AAC 48kHZ

Segment duration in sec	Video frames	Audio Packets (1024 frames per packet)
1.92	48	90
3.84	96	180
6.4	160	300



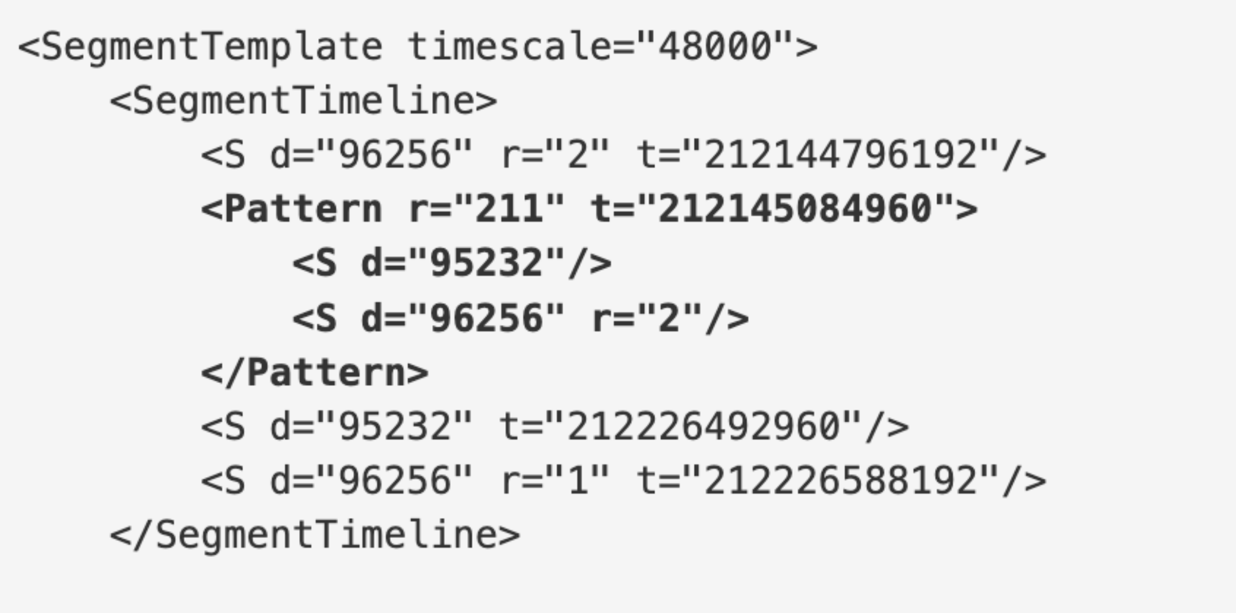
```
<AdaptationSet contentType="video"
    <SegmentTemplate timescale="90000">
        <SegmentTimeline>
             <S d="180000" r="149" t="143220940740000" />
        </SegmentTimeline>
    </SegmentTemplate>
</AdaptationSet>
<AdaptationSet contentType="audio">
         <SegmentTemplate timescale="48000">
              <SegmentTimeline>
                  <S d="96256" r="2" t="76384501728256" />
                  <S d="95232" />
                  <S d="96256" r="2" />
                  <S d="95232" />
                  <S d="96256" r="2" />
                  <S d="95232" />
                  .... lots of lines later
                  <S d="96256" r="2" />
                  <S d="95232" />
              </SegmentTimeline>
         </SegmentTemplate>
</AdaptationSet>
```

Repeating pattern in <SegmentTimeline> for the audio AdaptationSet

See https://websites.fraunhofer.de/video-dev/why-and-how-to-align-media-segments-for-abr-streaming/ for details

Excursus Segment Alignment: The <Patterns> tag

- Amazon Prime uses a <Pattern> tag to account for the repeating pattern of segment durations
- Not specification compliant i.e. not part of ISO/IEC 23009-1 or the DASH-IF IOP guidelines
- More details: https://websites.fraunhofer.de/video-dev/to-understand-is-to-perceive-patterns/



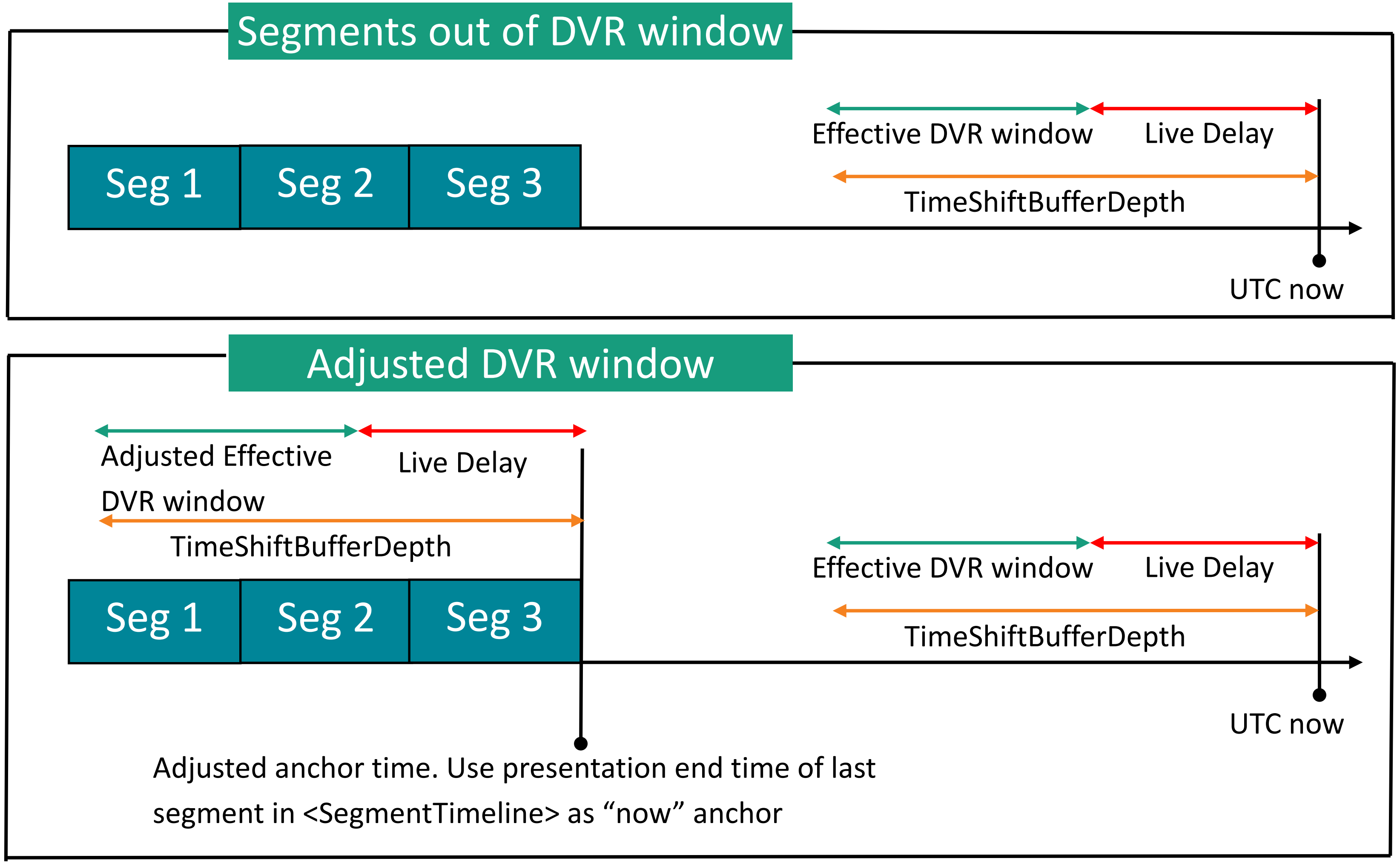
```
<SegmentTemplate timescale="48000">
    <SegmentTimeline>
        <S d="96256" r="2" t="212144796192"/>
        <Pattern r="211" t="212145084960">
            <S d="95232"/>
            <S d="96256" r="2"/>
        </Pattern>
        <S d="95232" t="212226492960"/>
        <S d="96256" r="1" t="212226588192"/>
    </SegmentTimeline>
```

Timing Violations: Fallback with <SegmentTimeline>

- In some cases, the media segments signaled via <SegmentTimeline> are out of the DVR window
- Violation of the DASH timing model and related to a server-side problem.
- If application provider is aware of this a workaround on the client side can be used: Use last segment in <SegmentTimeline> as "now" anchor

Timeline fallback

Parameter	Description
calcFrom Segment Timeline	Enable calculation of the DVR window for SegmentTimeline manifests based on the entries in <SegmentTimeline>



Adjusted anchor time. Use presentation end time of last segment in <SegmentTimeline> as "now" anchor

dash.js Features

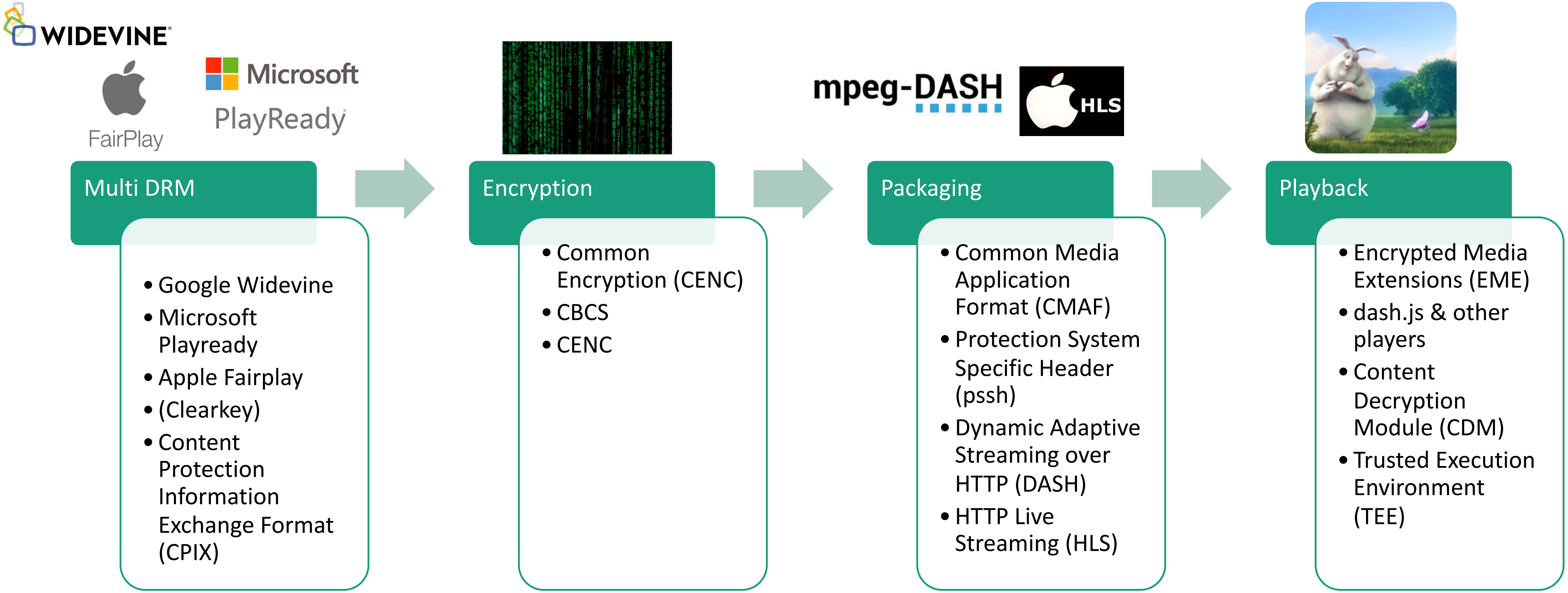
Recommendations / Best practices / Hints

- Aligning the duration of the audio and video segments can reduce the size of the manifest and save parsing time on the client-side
- Pay close attention to your CMAF fragment duration. A quality switch on chunk level is currently not supported. Future work items:
  - Resync Representations
  - ARI track
- Always specify a *<UTCTiming>* element to synchronize the clocks between client and encoder/packager
- Streaming in low latency mode is always a tradeoff between latency and buffer.
- If you use preconditioned content, consider signaling *availabilityTimeOffset="INF"* for specific periods
  - Example: https://reference.dashif.org/dash.js/nightly/samples/live-streaming/availability-time-offset.html

Chapter 07

Digital Rights Management

DRM in the Media Streaming Stack



Multi DRM

- Google Widevine
- Microsoft Playready
- Apple Fairplay
- (Clearkey)
- Content Protection Information Exchange Format (CPIX)

Encryption

- Common Encryption (CENC)
- CBCS
- CENC

Packaging

- Common Media Application Format (CMAF)
- Protection System Specific Header (pssh)
- Dynamic Adaptive Streaming over HTTP (DASH)
- HTTP Live Streaming (HLS)

Playback

- Encrypted Media Extensions (EME)
- dash.js & other players
- Content Decryption Module (CDM)
- Trusted Execution Environment (TEE)

Different EME models in dash.js

01.b

- Initial implementation of the EME, implemented by Google Chrome prior to version 36.
- This EME version is not-promised based and uses outdated or prefixed events like “needkey” or “webkitneedkey

2014

- Implementation of EME APIs as of the 3 Feb 2014 state of the specification. Implemented by Internet Explorer 11 (Windows 8.1)

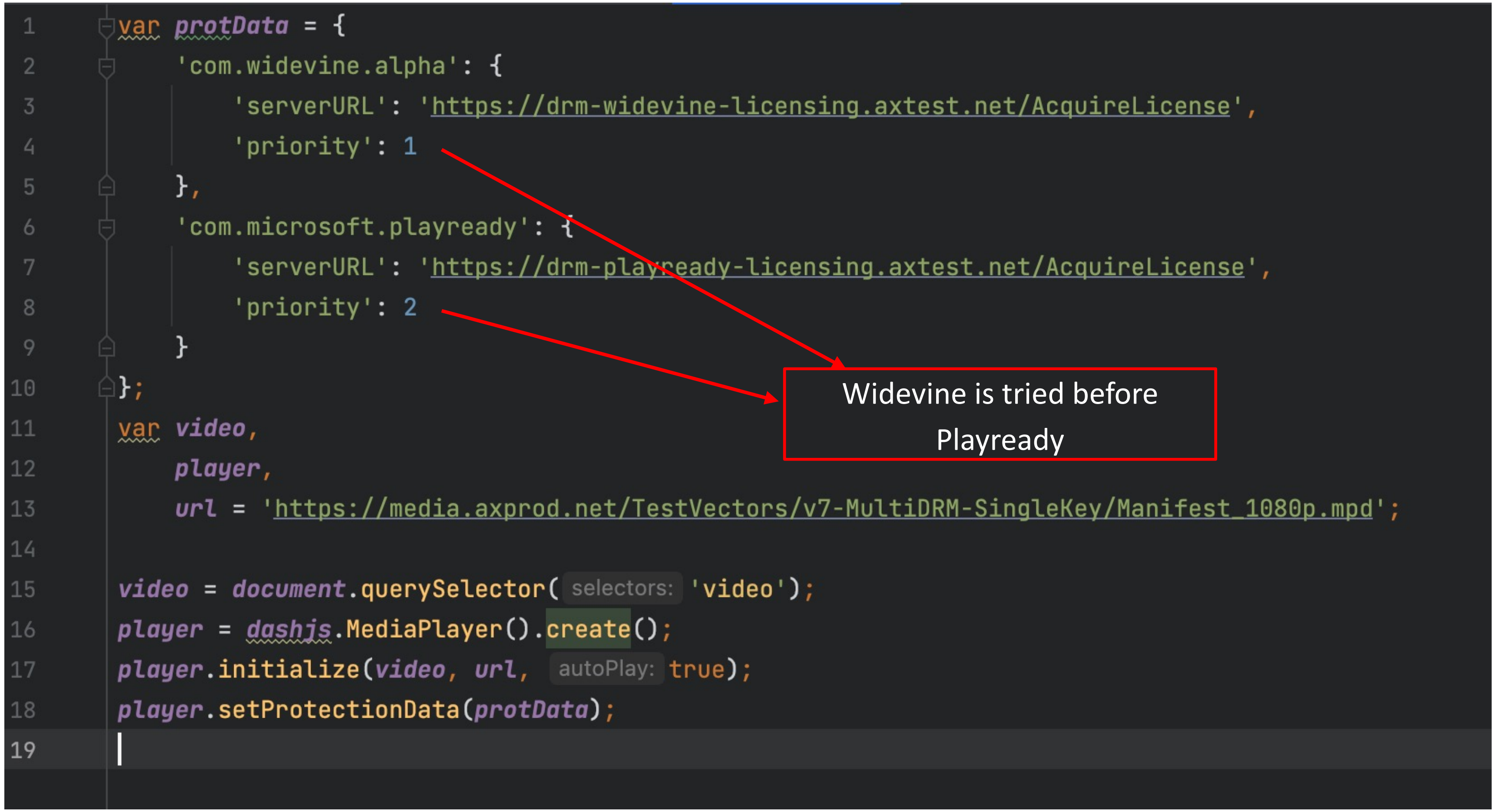
2015

- Most recent EME implementation. Latest changes in the EME specification are added to this model
- It supports the promised-based EME function calls.

Some platforms require customized (e.g. custom-prefixed) EME implementations. Stepping through the EME workflow helps identifying required changes.

DRM System Priority

- In case multiple DRM systems are supported on the target platform priorities to the systems can be assigned
- Example: https://reference.dashif.org/dash.js/nightly/samples/drm/system-priority.html

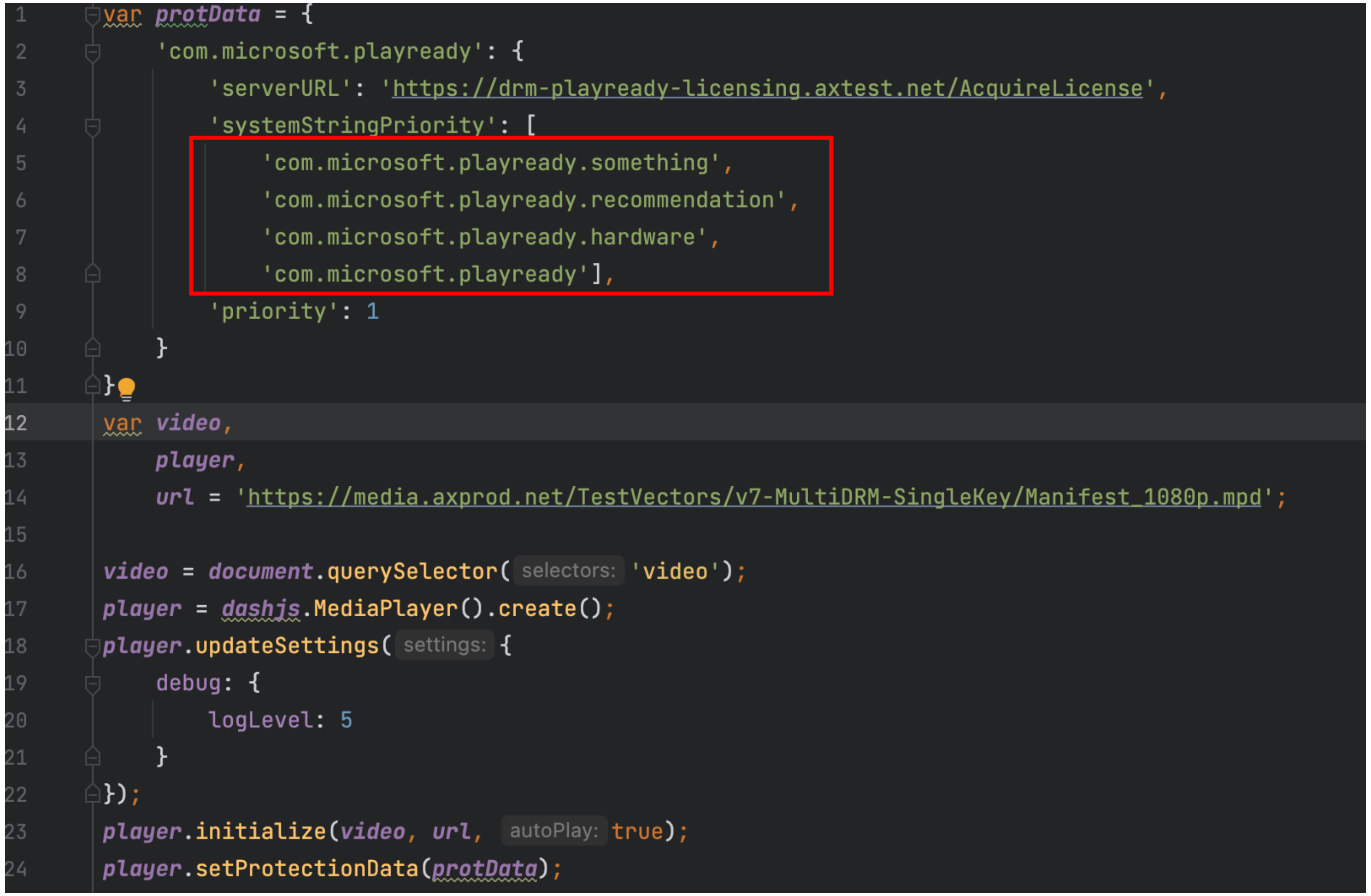


```
var protData = {
    'com.widevine.alpha': {
        'serverURL': 'https://drm-widevine-licensing.axtest.net/AcquireLicense',
        'priority': 1
    },
    'com.microsoft.playready': {
        'serverURL': 'https://drm-playready-licensing.axtest.net/AcquireLicense',
        'priority': 2
    }
};
var video,
    player,
    url = 'https://media.axprod.net/TestVectors/v7-MultiDRM-SingleKey/Manifest_1080p.mpd';

video = document.querySelector( selectors: 'video');
player = dashjs.MediaPlayer().create();
player.initialize(video, url,  autoPlay: true);
player.setProtectionData(protData);
```

Widevine is tried before Playready

Key System String Priority

- Initial EME call “*requestMediaKeySystemAccess*” requires a key system string for which the access is being requested
- Example Playready:
  - “*com.microsoft.playready.recommendation*”
    - Correct system string for Edge
  - “*com.microsoft.playready*”:
    - Fallback for legacy implementations
  - „*com.microsoft.playready.recommendation.3000*“
    - Forces HW DRM on Windows for video
  - Additional information: https://github.com/Dash-Industry-Forum/dash.js/issues/3852
  - Example: https://reference.dashif.org/dash.js/nightly/samples/drm/system-string-priority.html



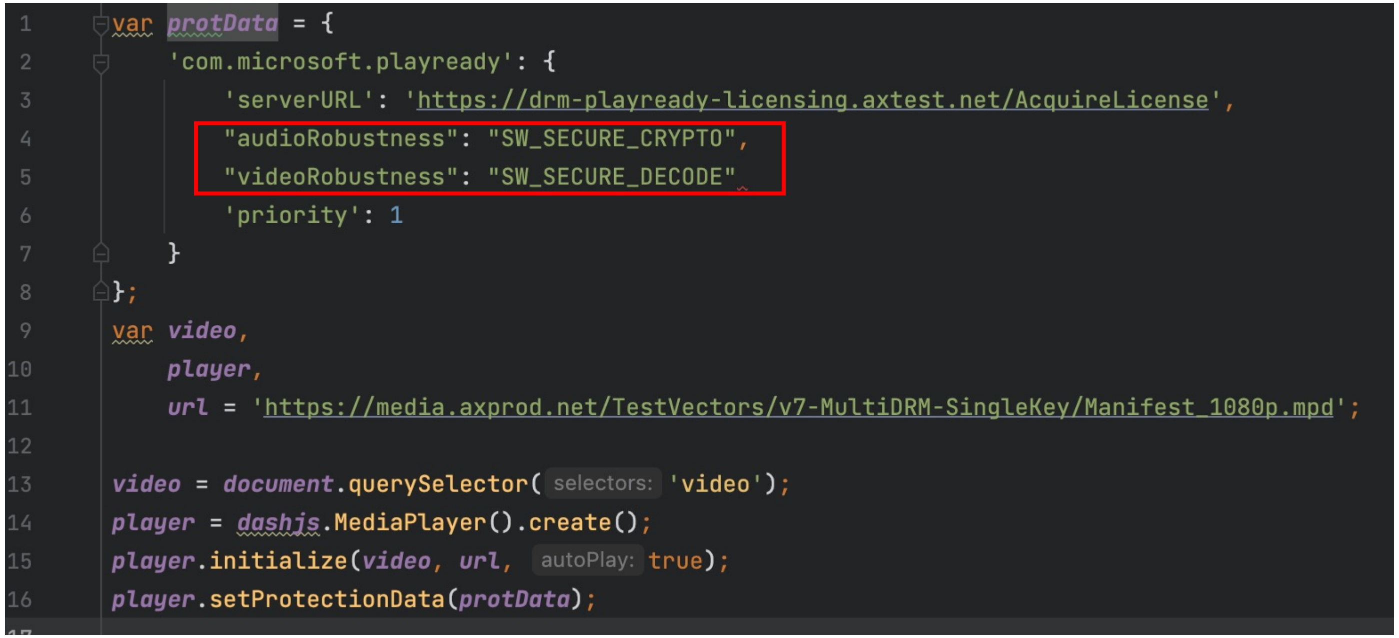
```
var protData = {
    'com.microsoft.playready': {
        'serverURL': 'https://drm-playready-licensing.axtest.net/AcquireLicense',
        'systemStringPriority': [
            'com.microsoft.playready.something',
            'com.microsoft.playready.recommendation',
            'com.microsoft.playready.hardware',
            'com.microsoft.playready'],
        'priority': 1
    }
}
var video,
    player,
    url = 'https://media.axprod.net/TestVectors/v7-MultiDRM-SingleKey/Manifest_1080p.mpd';

video = document.querySelector( selectors: 'video');
player = dashjs.MediaPlayer().create();
player.updateSettings( settings: {
    debug: {
        logLevel: 5
    }
});
player.initialize(video, url,  autoPlay: true);
player.setProtectionData(protData);
```

Digital Rights Management

Robustness levels

Initial call to EME should contain a robustness level that maps to a specific DRM security level:

EME Level	Playready	Widevine
1	2000	SW_SECURE_CRYPTO (L3)
2	2000	SW_SECURE_DECODE (L3)
3	2000	HW_SECURE_CRYPTO (L2)
4	2000	HW_SECURE_DECODE (L1)
5	3000	HW_SECURE_ALL (L1)



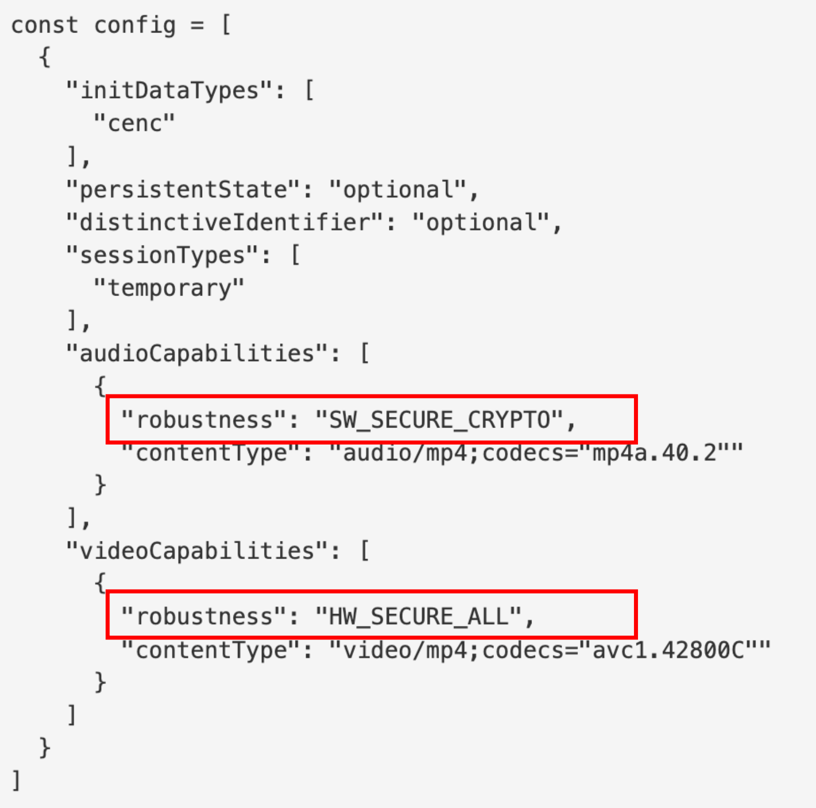
```
var protData = {
    'com.microsoft.playready': {
        'serverURL': 'https://drm-playready-licensing.axtest.net/AcquireLicense',
        "audioRobustness": "SW_SECURE_CRYPTO",
        "videoRobustness": "SW_SECURE_DECODE"
        'priority': 1
    }
};
var video,
    player,
    url = 'https://media.axprod.net/TestVectors/v7-MultiDRM-SingleKey/Manifest_1080p.mpd';

video = document.querySelector( selectors: 'video');
player = dashjs.MediaPlayer().create();
player.initialize(video, url,  autoPlay: true);
player.setProtectionData(protData);
```

Excursus: Hardware DRM on mobile devices

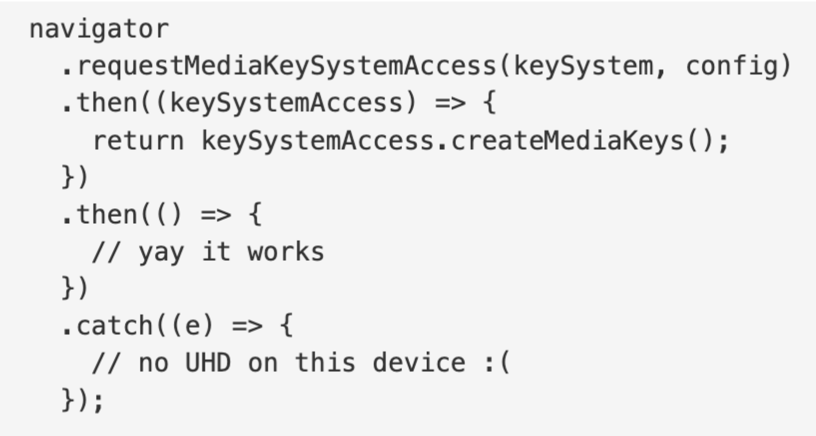
- Some devices successfully resolve the promise returned *requestMediaKeySystemAccess* but fail to create the *MediaKeys* afterwards.

	Samsung Galaxy S9 Android 9 Chrome 75	**HTC OnePlus 5T Android 8.1 Chrome 75**
requestMediaKey SystemAcess	✓	✓
keySystemAccess. createMediaKeys	✓	✗

Tests performed in 2019



```
const config = [
  {
    "initDataTypes": [
      "cenc"
    ],
    "persistentState": "optional",
    "distinctiveIdentifier": "optional",
    "sessionTypes": [
      "temporary"
    ],
    "audioCapabilities": [
      {
        "robustness": "SW_SECURE_CRYPTO",
        "contentType": "audio/mp4;codecs="mp4a.40.2""
      }
    ],
    "videoCapabilities": [
      {
        "robustness": "HW_SECURE_ALL",
        "contentType": "video/mp4;codecs="avc1.42800C""
      }
    ]
  }
]
```



```
navigator
  .requestMediaKeySystemAccess(keySystem, config)
  .then((keySystemAccess) => {
    return keySystemAccess.createMediaKeys();
  })
  .then(() => {
    // yay it works
  })
  .catch((e) => {
    // no UHD on this device :(
  });
```

Additional settings

- License server URLs via API and MPD *<dashif:Laurl>* element
- DRM specific headers: Add custom headers to your license request
- Define promise-based callback functions to modify license request and license response
- Preserve MediaKeys and MediaKeySessions during MediaPlayer lifetime to avoid new license requests
- See DRM sample section and documentation:
  - https://reference.dashif.org/dash.js/nightly/samples/index.html
  - https://github.com/Dash-Industry-Forum/dash.js/wiki/Digital-Rights-Management-(DRM)-and-license-acquisition

Digital Rights Management

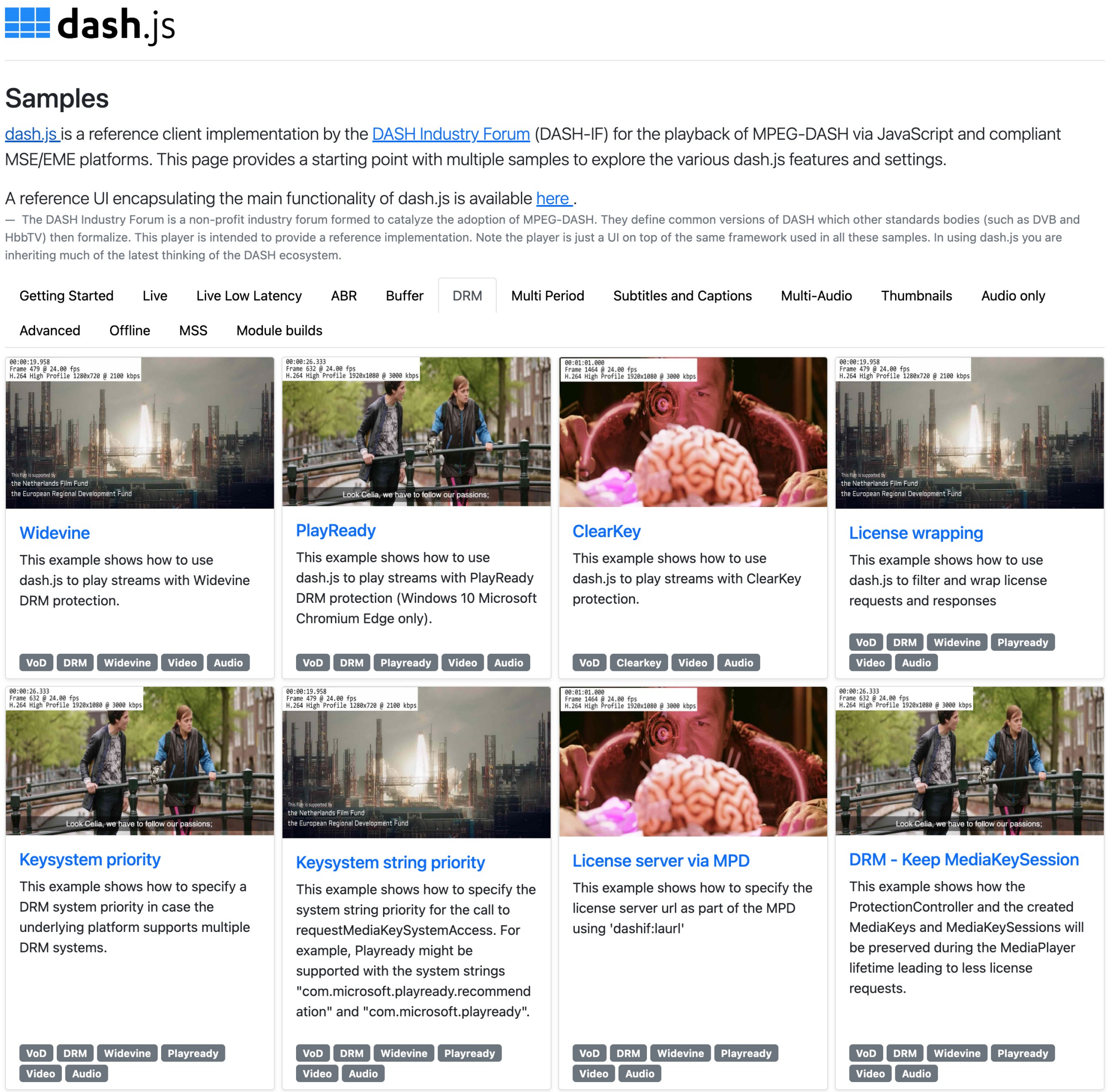
Recommendations / Best practices / Hints

- EME support requires the application to be hosted with https.
- Some platforms required customized (e.g. prefixed) EME implementations.
- Old platforms might not support CBCS encryption for Widevine and Playready, see https://websites.fraunhofer.de/video-dev/is-this-the-end-of-cenc-an-overview-of-drm-codec-support-in-2021/
- Make sure to use correct robustness level and key system string when enforcing Hardware DRM
- Checking for Hardware DRM support might require multiple calls to the EME
- Chrome does not support support HEVC with Widevine

Digital Rights Management

dash.js demo

Try it out yourself:
https://reference.dashif.org/dash.js/nightly/samples/ -> DRM Section/Tab



dash.js

Samples

dash.js is a reference client implementation by the DASH Industry Forum (DASH-IF) for the playback of MPEG-DASH via JavaScript and compliant MSE/EME platforms. This page provides a starting point with multiple samples to explore the various dash.js features and settings.

A reference UI encapsulating the main functionality of dash.js is available here .

— The DASH Industry Forum is a non-profit industry forum formed to catalyze the adoption of MPEG-DASH. They define common versions of DASH which other standards bodies (such as DVB and HbbTV) then formalize. This player is intended to provide a reference implementation. Note the player is just a UI on top of the same framework used in all these samples. In using dash.js you are inheriting much of the latest thinking of the DASH ecosystem.

Widevine

This example shows how to use dash.js to play streams with Widevine DRM protection.

VoD DRM Widevine Video Audio

PlayReady

This example shows how to use dash.js to play streams with PlayReady DRM protection (Windows 10 Microsoft Chromium Edge only).

VoD DRM Playready Video Audio

ClearKey

This example shows how to use dash.js to play streams with ClearKey protection.

VoD Clearkey Video Audio

License wrapping

This example shows how to use dash.js to filter and wrap license requests and responses

VoD DRM Widevine Playready Video Audio

Keysystem priority

This example shows how to specify a DRM system priority in case the underlying platform supports multiple DRM systems.

VoD DRM Widevine Playready Video Audio

Keysystem string priority

This example shows how to specify the system string priority for the call to requestMediaKeySystemAccess. For example, Playready might be supported with the system strings "com.microsoft.playready.recommendation" and "com.microsoft.playready".

VoD DRM Widevine Playready Video Audio

License server via MPD

This example shows how to specify the license server url as part of the MPD using 'dashif:laurl'

VoD DRM Widevine Playready Video Audio

DRM - Keep MediaKeySession

This example shows how the ProtectionController and the created MediaKeys and MediaKeySessions will be preserved during the MediaPlayer lifetime leading to less license requests.

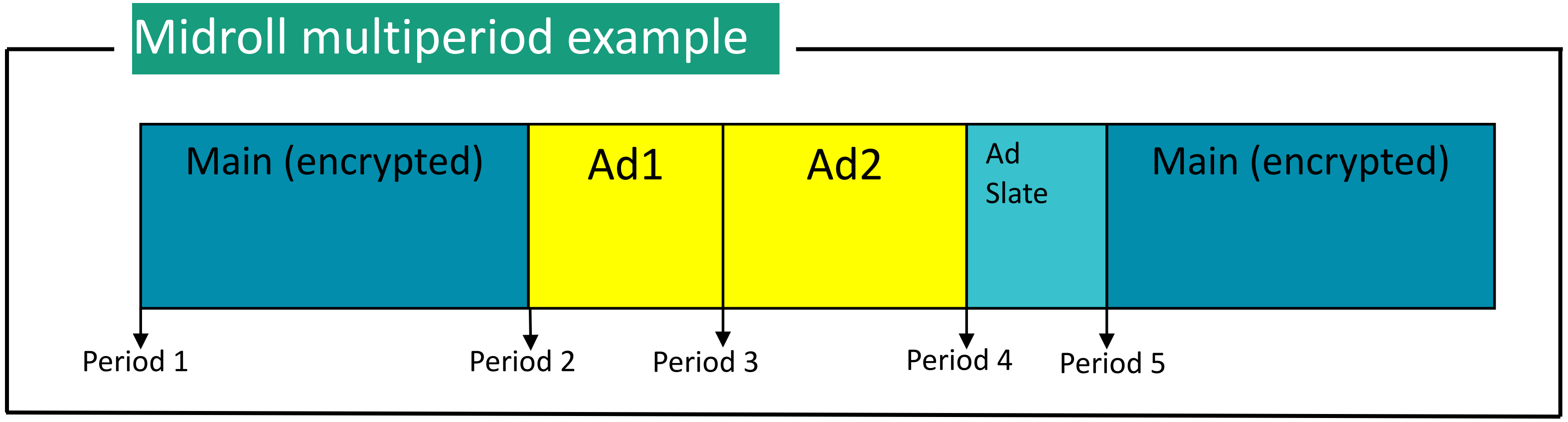
VoD DRM Widevine Playready Video Audio

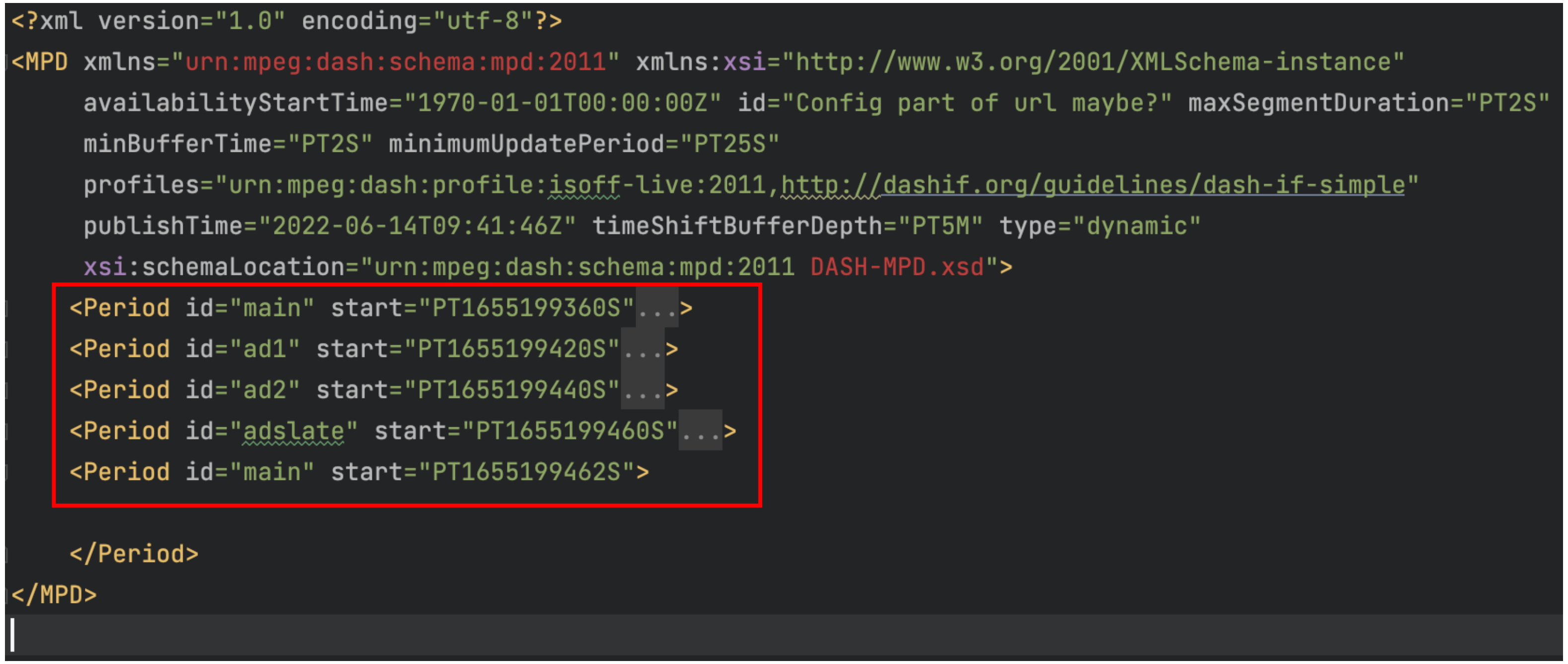
Chapter 06

Multiperiod Playback and Gap Handling

Multiperiod Playback

- Multiperiod playback enables use cases such as
  - server-side ad-insertion
  - transition between encrypted and non-encrypted content
  - A codec change e.g from H.264 to H.265



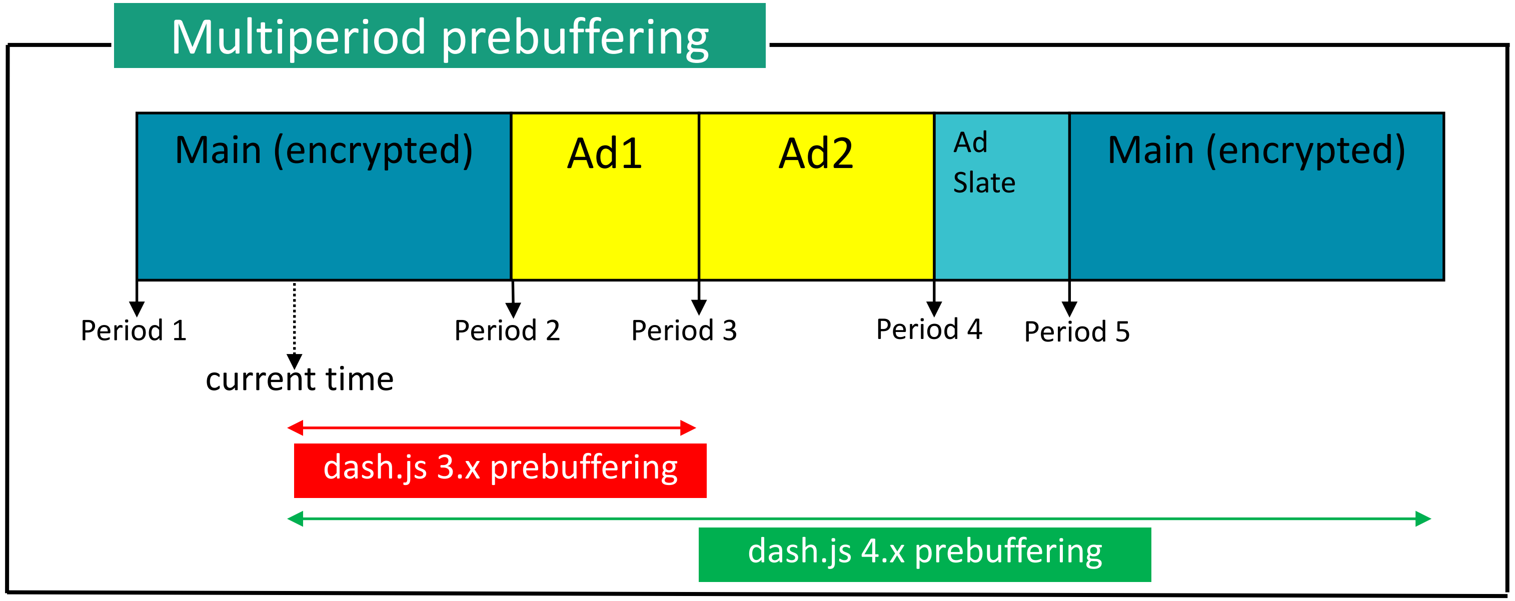


```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     availabilityStartTime="1970-01-01T00:00:00Z" id="Config part of url maybe?" maxSegmentDuration="PT2S"
     minBufferTime="PT2S" minimumUpdatePeriod="PT25S"
     profiles="urn:mpeg:dash:profile:isoff-live:2011,http://dashif.org/guidelines/dash-if-simple"
     publishTime="2022-06-14T09:41:46Z" timeShiftBufferDepth="PT5M" type="dynamic"
     xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd">
    <Period id="main" start="PT1655199360S"...>
    <Period id="ad1" start="PT1655199420S"...>
    <Period id="ad2" start="PT1655199440S"...>
    <Period id="adslate" start="PT1655199460S"...>
    <Period id="main" start="PT1655199462S">

    </Period>
</MPD>
```

Multiperiod Playback

dash.js - Features

- dash.js 4.x supports
  - Prebuffering of multiple upcoming periods to maintain the specified buffer target
  - Support for transition between encrypted and non encrypted periods
  - A DVR window overlapping multiple periods. Seeking within the DVR window is not limited to a single period anymore.
- Support for MSE v.2 *SourceBuffer.changeType()* to enable codec changes without MSE reinitialization



Multiperiod prebuffering

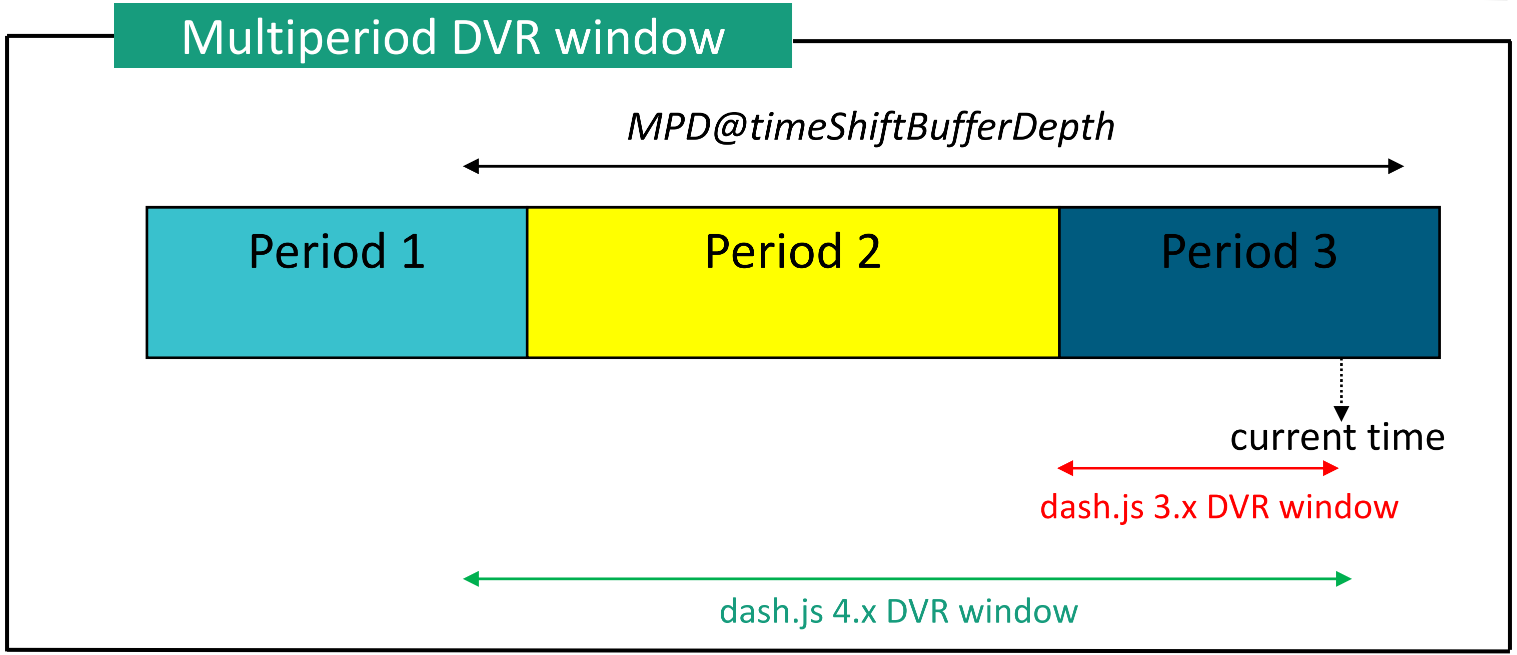
Main (encrypted) | Ad1 | Ad2 | Ad Slate | Main (encrypted)

Period 1, Period 2, Period 3, Period 4, Period 5

current time

dash.js 3.x prebuffering

dash.js 4.x prebuffering



Multiperiod DVR window

MPD@timeShiftBufferDepth

Period 1 | Period 2 | Period 3

current time

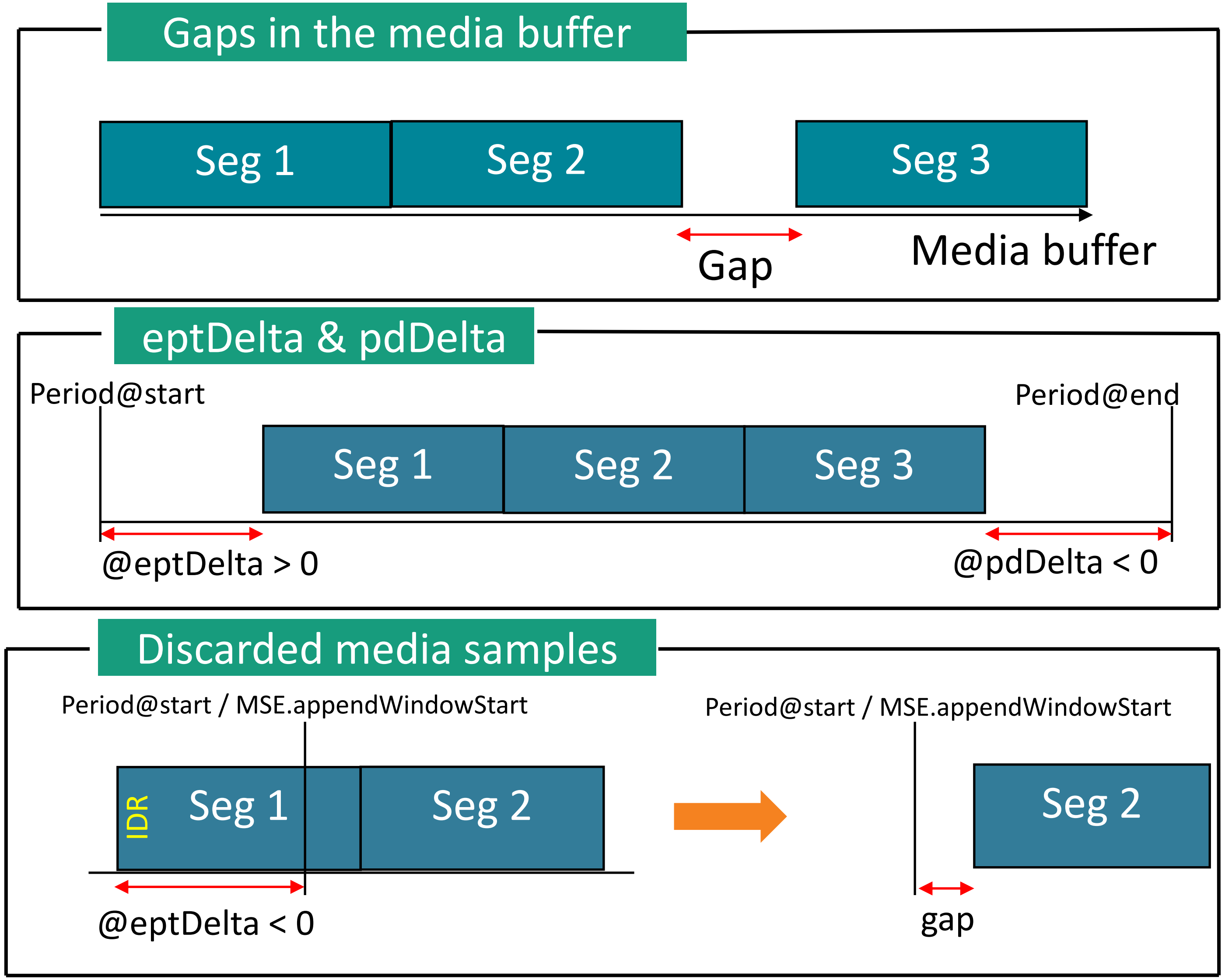
dash.js 3.x DVR window

dash.js 4.x DVR window

Gap Handling

- **MSE implementations stall if the buffer is not continuous.**

- There are various reasons for gaps in the media buffer:
  - Unaligned Periods or segments
  - Sample duration does not match segment duration
  - Positive *@eptDelta* or negative *@pdDelta*
  - Negative *@eptDelta* for video can lead to lost media samples, see
  - Related blog post: https://tinyurl.com/eptdelta
- GapController class in dash.js handles such gaps
  - VoD: Immediate seek
  - Live: Delayed seek, keep consistent live edge



Multiperiod playback & Gap Handling

dash.js - Configuration

Multiperiod in dash.js

Parameter	Description	Default
useAppend Window	Specifies if the *appendWindow* attributes of the MSE SourceBuffers should be set according to the period durations in manifest.	true
Reuse Existing SourceBuffers	Enable reuse of existing MediaSource Sourcebuffers during period transition	true

Gap Handling in dash.js

Parameter	Description	Default
jumpGaps	Defines whether the player should jump small gaps (discontinuities) in the buffer.	true
threshold	• Threshold at which the gap handling is executed. If currentRangeEnd - currentTime < threshold the gap jump will be triggered. • For live stream the jump is delayed to keep a consistent live edge. • Note that the amount of buffer at which platforms automatically stall might differ.	0.3
enableSeekFix	Enables the adjustment of the seek target once no valid segment request could be generated for a specific seek time. This can happen if the user seeks to a position for which there is a gap in the timeline.	true
enableStallFix	If playback stalled in a buffered range this fix will perform a seek by the value defined in *stallSeek* to trigger playback again	false

Multiperiod Playback and Gap Handling

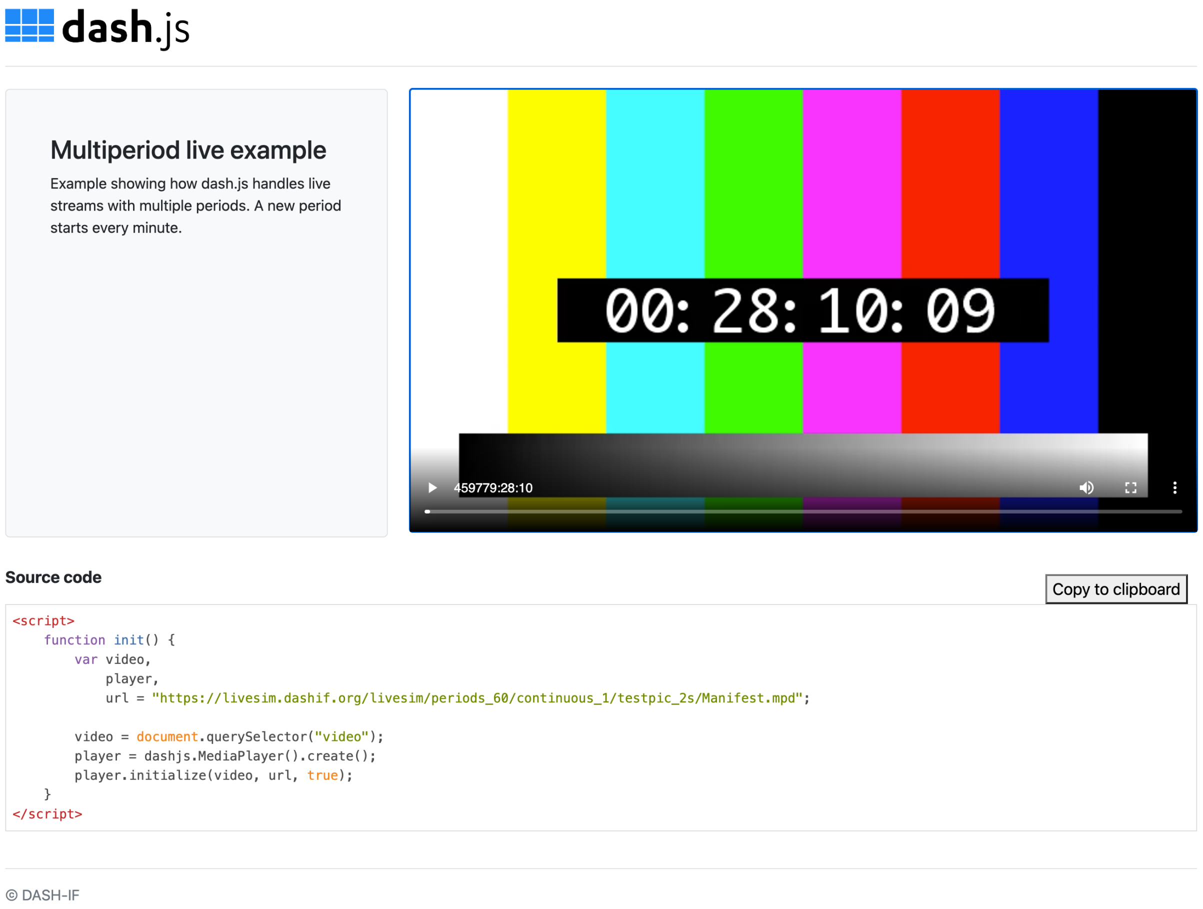
Recommendations / Best practices / Hints

- Don’t remove periods that are still in the DVR window
- Don’t change period IDs
- Avoid segment overlaps at period boundaries
  - A negative @eptDelta can lead to samples being dropped from the buffer
  - A positive @eptDelta leads to a gap at the beginning of a period
  - A negative @pdDelta leads to a gap at the end of a period
- A switch from non-encrypted to encrypted content can cause an MSE reset.

Multiperiod Playback and Gap Handling

dash.js demo

Try it out yourself:
https://reference.dashif.org/dash.js/nightly/samples/multiperiod/live.html



dash.js

Multiperiod live example

Example showing how dash.js handles live streams with multiple periods. A new period starts every minute.

Source code

Copy to clipboard

```
<script>
    function init() {
        var video,
            player,
            url = "https://livesim.dashif.org/livesim/periods_60/continuous_1/testpic_2s/Manifest.mpd";

        video = document.querySelector("video");
        player = dashjs.MediaPlayer().create();
        player.initialize(video, url, true);
    }
</script>
```

© DASH-IF

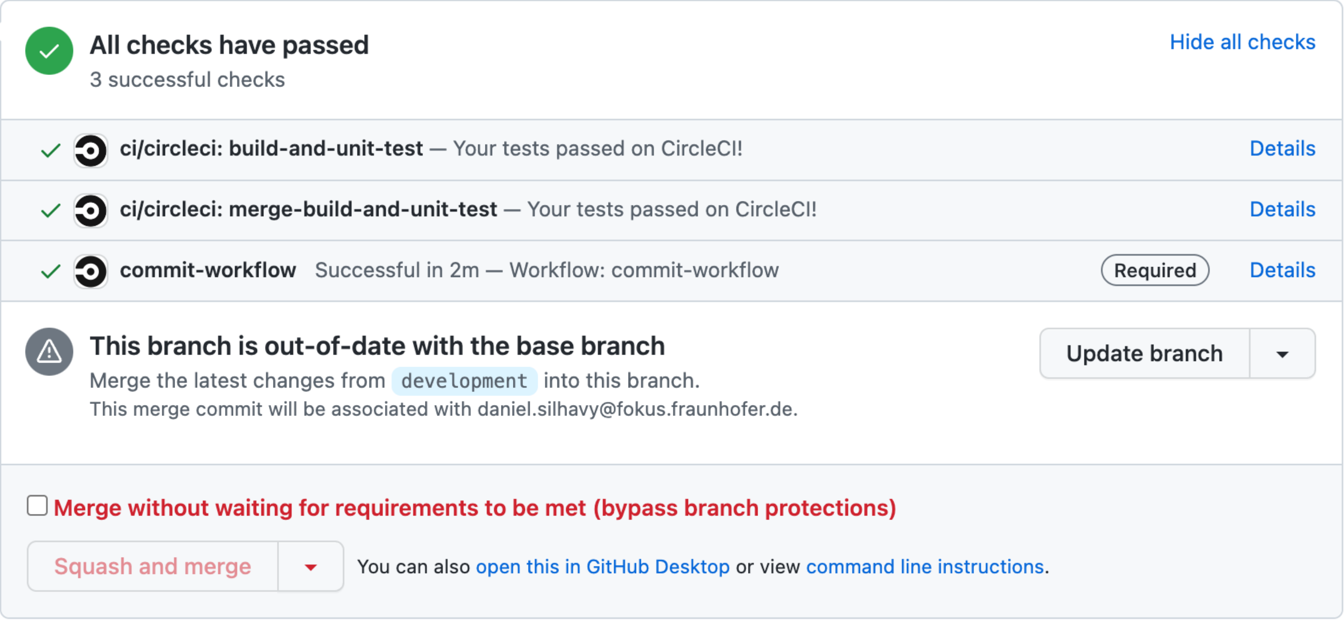
Chapter 08

dash.js - Unit and Functional Testing

dash.js - Unit and Functional Testing

Unit Tests

- Test individual functions or methods (units)
- Located in „test/unit“, can be executed via „npm run test“
- Automatically triggered for each pull request
- Note: Until dash.js 4.6.0 the unit tests were executed in a node.js context. Some missing objects like the „window“ were only present as a mocked implementation. dash.js 4.6.0 introduces unit test execution via the Karma testrunner in “real“ browsers. Typically the execution is performed in headless mode.



All checks have passed
3 successful checks

✓ **ci/circleci: build-and-unit-test** — Your tests passed on CircleCI!

✓ **ci/circleci: merge-build-and-unit-test** — Your tests passed on CircleCI!

✓ **commit-workflow** Successful in 2m — Workflow: commit-workflow

This branch is out-of-date with the base branch
Merge the latest changes from `development` into this branch.
This merge commit will be associated with daniel.silhavy@fokus.fraunhofer.de.

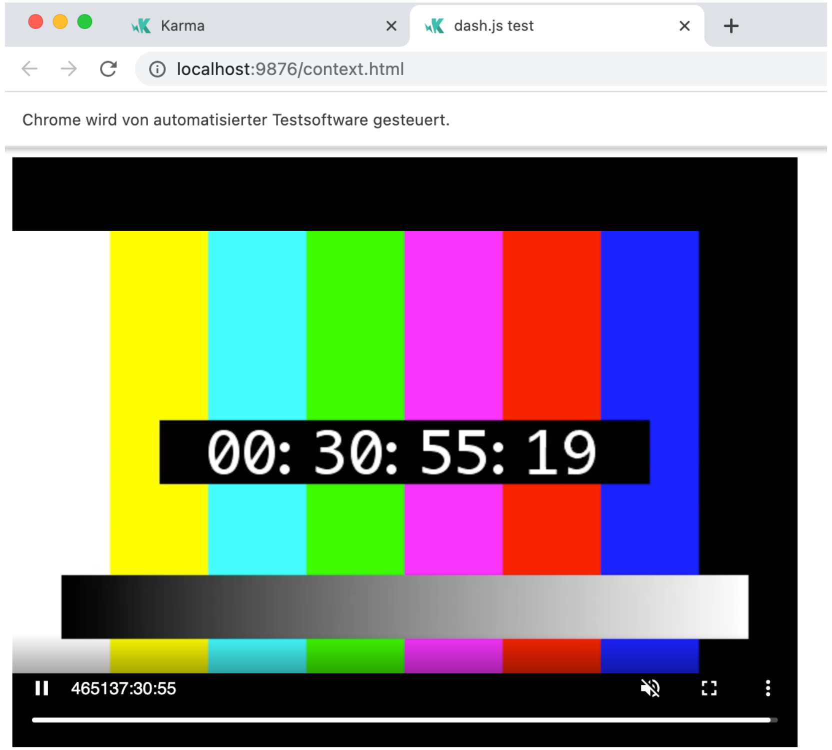
□ Merge without waiting for requirements to be met (bypass branch protections)

You can also open this in GitHub Desktop or view command line instructions.

dash.js - Unit and Functional Testing

Functional Tests

- Checks the functionality of the player, for example play, pause and seek
- Automated execution of certain steps and verification of the playback state afterwards
- Based on Selenium Grid and Intern framework
- Located in „test/functional“, documentation can be found here: https://github.com/Dash-Industry-Forum/dash.js/blob/development/test/functional/readme.md
- Daniel working on a new testsuite based on Karma Testrunner, Selenium Grid and Appium. Allows execution on devices such as Samsung SmartTVs and Android phones.



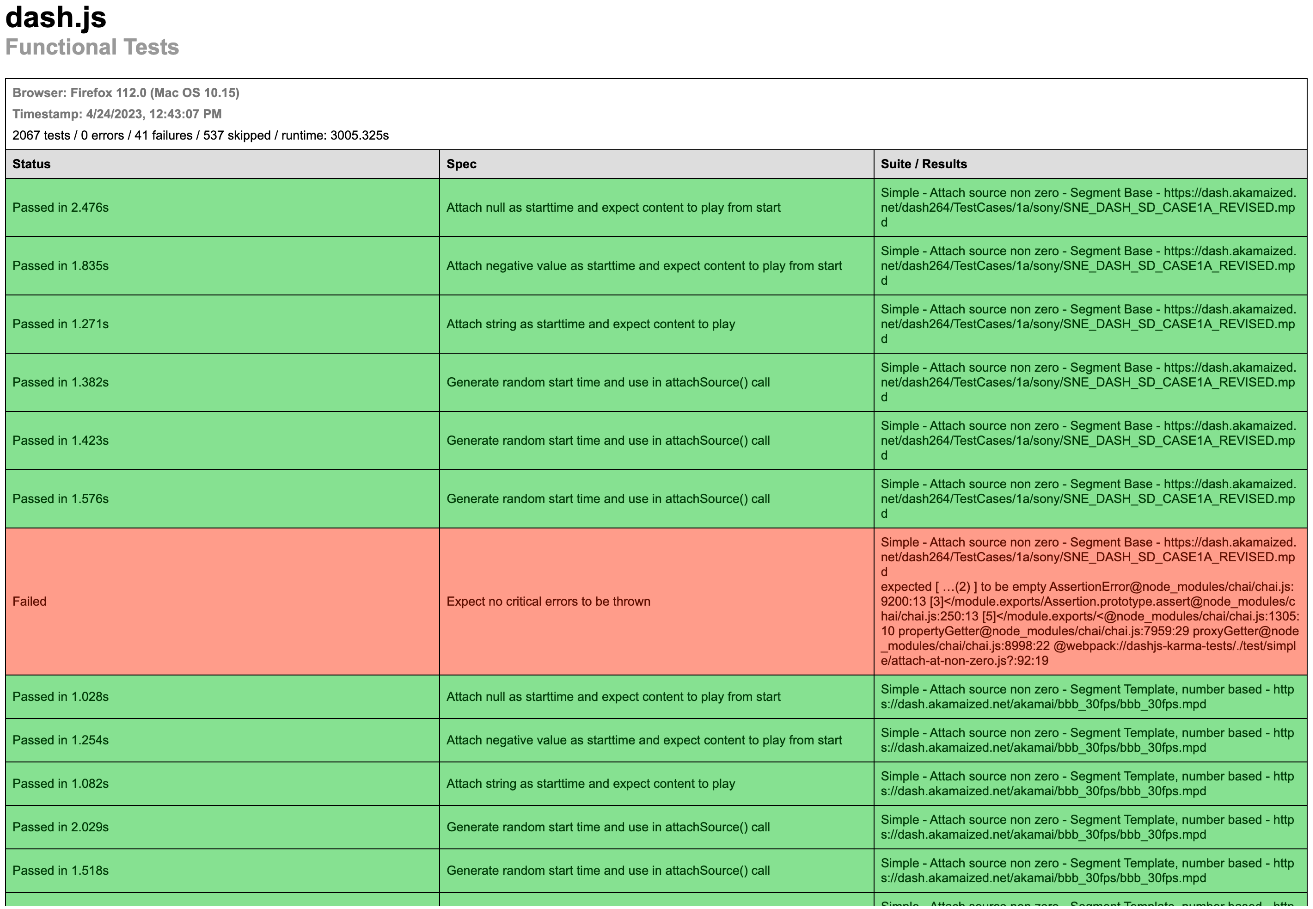
Chrome wird von automatisierter Testsoftware gesteuert.

dash.js - Unit and Functional Testing

Demo - Functional Tests

dash.js - Unit and Functional Testing

Demo Report - Functional Tests



dash.js

Functional Tests

Browser: Firefox 112.0 (Mac OS 10.15)

Timestamp: 4/24/2023, 12:43:07 PM

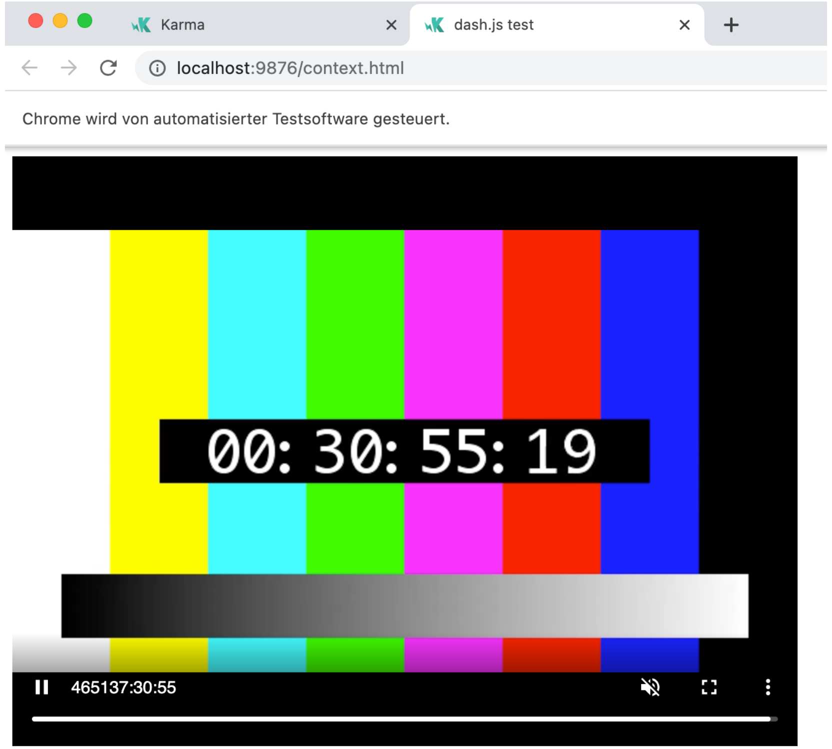
2067 tests / 0 errors / 41 failures / 537 skipped / runtime: 3005.325s

Status	Spec	Suite / Results
Passed in 2.476s	Attach null as starttime and expect content to play from start	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Passed in 1.835s	Attach negative value as starttime and expect content to play from start	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Passed in 1.271s	Attach string as starttime and expect content to play	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Passed in 1.382s	Generate random start time and use in attachSource() call	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Passed in 1.423s	Generate random start time and use in attachSource() call	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Passed in 1.576s	Generate random start time and use in attachSource() call	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd
Failed	Expect no critical errors to be thrown	Simple - Attach source non zero - Segment Base - https://dash.akamaized.net/dash264/TestCases/1a/sony/SNE_DASH_SD_CASE1A_REVISED.mpd expected […(2)] to be empty AssertionError@node_modules/chai/chai.js:9200:13 [3]</module.exports/Assertion.prototype.assert@node_modules/chai/chai.js:250:13 [5]</module.exports/<@node_modules/chai/chai.js:1305:10 propertyGetter@node_modules/chai/chai.js:7959:29 proxyGetter@node_modules/chai/chai.js:8998:22 @webpack://dashjs-karma-tests/./test/simple/attach-at-non-zero.js?:92:19
Passed in 1.028s	Attach null as starttime and expect content to play from start	Simple - Attach source non zero - Segment Template, number based - https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd
Passed in 1.254s	Attach negative value as starttime and expect content to play from start	Simple - Attach source non zero - Segment Template, number based - https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd
Passed in 1.082s	Attach string as starttime and expect content to play	Simple - Attach source non zero - Segment Template, number based - https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd
Passed in 2.029s	Generate random start time and use in attachSource() call	Simple - Attach source non zero - Segment Template, number based - https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd
Passed in 1.518s	Generate random start time and use in attachSource() call	Simple - Attach source non zero - Segment Template, number based - https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd
		Simple - Attach source non zero - Segment Template, number based - http

dash.js - Unit and Functional Testing

dash.js demo

Functional Tests



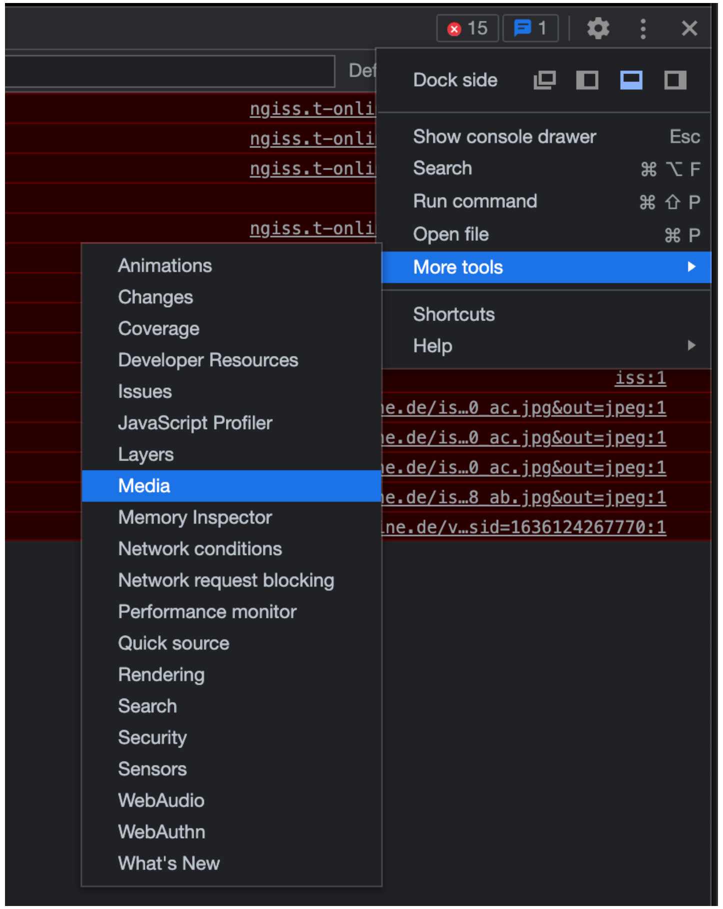
Chrome wird von automatisierter Testsoftware gesteuert.

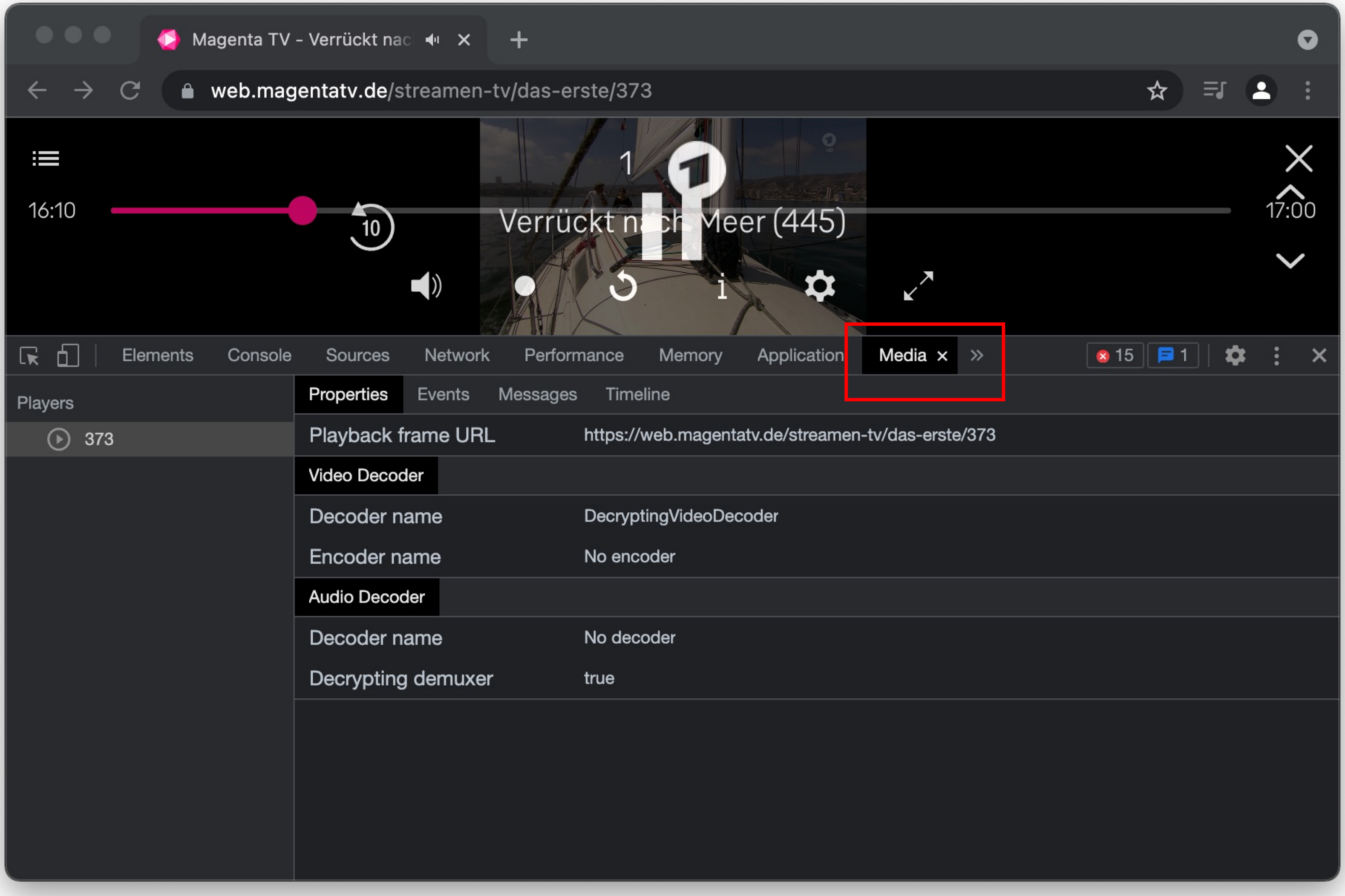
Chapter 09

How to debug your streams

How to debug your streams

Browser based debugging





Playback frame URL https://web.magentatv.de/streamen-tv/das-erste/373

Video Decoder

Decoder name DecryptingVideoDecoder

Encoder name No encoder

Audio Decoder

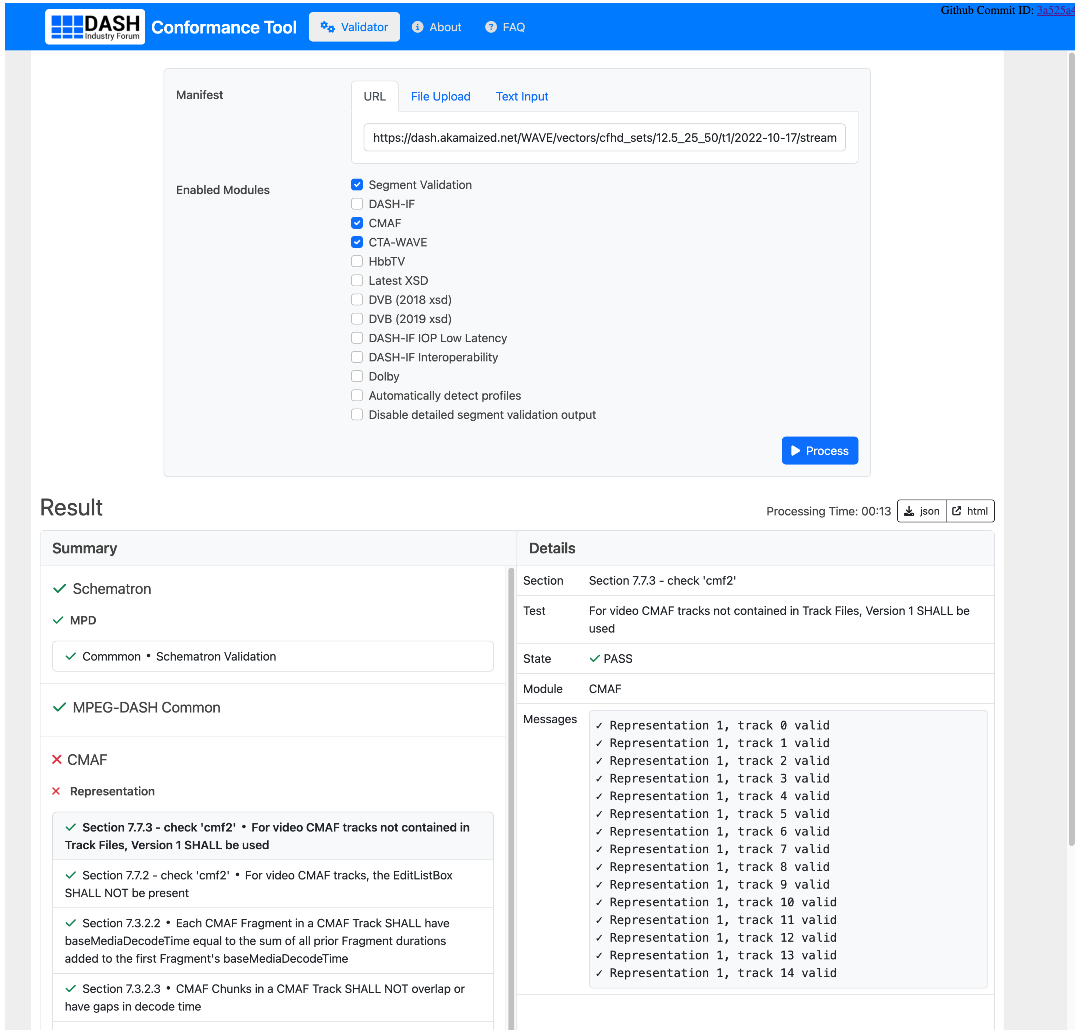
Decoder name No decoder

Decrypting demuxer true

How to debug your streams

DASH validation

- DASH-IF Conformance Validator supports multiple profiles such as DASH-IF, DVB, HbbTV, CMAF and CTA-WAVE
- The Conformance Validator was refactored and optimized as part of the Joint Conformance Software Project (JCCP). Join us on Github, Slack and Google Groups
  - https://github.com/Dash-Industry-Forum/DASH-IF-Conformance
  - https://groups.google.com/g/joint-conformance-software-project-jccp/members
  - https://join.slack.com/t/dashif/shared_invite/zt-191r8cjva-4bu_5_SJ1U~d_oltjqWkEQ in #jccp



Github Commit ID: 3a525a

Manifest

URL | File Upload | Text Input

https://dash.akamaized.net/WAVE/vectors/cfhd_sets/12.5_25_50/t1/2022-10-17/stream

Enabled Modules

☑ Segment Validation
□ DASH-IF
☑ CMAF
☑ CTA-WAVE
□ HbbTV
□ Latest XSD
□ DVB (2018 xsd)
□ DVB (2019 xsd)
□ DASH-IF IOP Low Latency
□ DASH-IF Interoperability
□ Dolby
□ Automatically detect profiles
□ Disable detailed segment validation output

Result

Processing Time: 00:13

Summary

✓ Schematron

✓ MPD

✓ Commmon • Schematron Validation

✓ MPEG-DASH Common

✗ CMAF

✗ Representation

✓ **Section 7.7.3 - check 'cmf2' • For video CMAF tracks not contained in Track Files, Version 1 SHALL be used**

✓ Section 7.7.2 - check 'cmf2' • For video CMAF tracks, the EditListBox SHALL NOT be present

✓ Section 7.3.2.2 • Each CMAF Fragment in a CMAF Track SHALL have baseMediaDecodeTime equal to the sum of all prior Fragment durations added to the first Fragment's baseMediaDecodeTime

✓ Section 7.3.2.3 • CMAF Chunks in a CMAF Track SHALL NOT overlap or have gaps in decode time

Details

Section	Section 7.7.3 - check 'cmf2'
Test	For video CMAF tracks not contained in Track Files, Version 1 SHALL be used
State	✓ PASS
Module	CMAF
Messages	

```
✓ Representation 1, track 0 valid
✓ Representation 1, track 1 valid
✓ Representation 1, track 2 valid
✓ Representation 1, track 3 valid
✓ Representation 1, track 4 valid
✓ Representation 1, track 5 valid
✓ Representation 1, track 6 valid
✓ Representation 1, track 7 valid
✓ Representation 1, track 8 valid
✓ Representation 1, track 9 valid
✓ Representation 1, track 10 valid
✓ Representation 1, track 11 valid
✓ Representation 1, track 12 valid
✓ Representation 1, track 13 valid
✓ Representation 1, track 14 valid
```

Conformance Validator: https://conformance.dashif.org/

ISOBMFF Segment Inspection

Bento4

- “A fast, modern, open-source C++ toolkit for all your MP4 and DASH/HLS/CMAF media format needs.“
- MPEG DASH & HLS packager
- MP4 parsing and modification
- Encryption and Decryption
- See https://www.bento4.com/

ISOViewer

- “GUI application to have closer look ISO 14496-12 and other MP4 files”
- Read only
- https://github.com/sannies/isoviewer/releases

MP4 Inspector

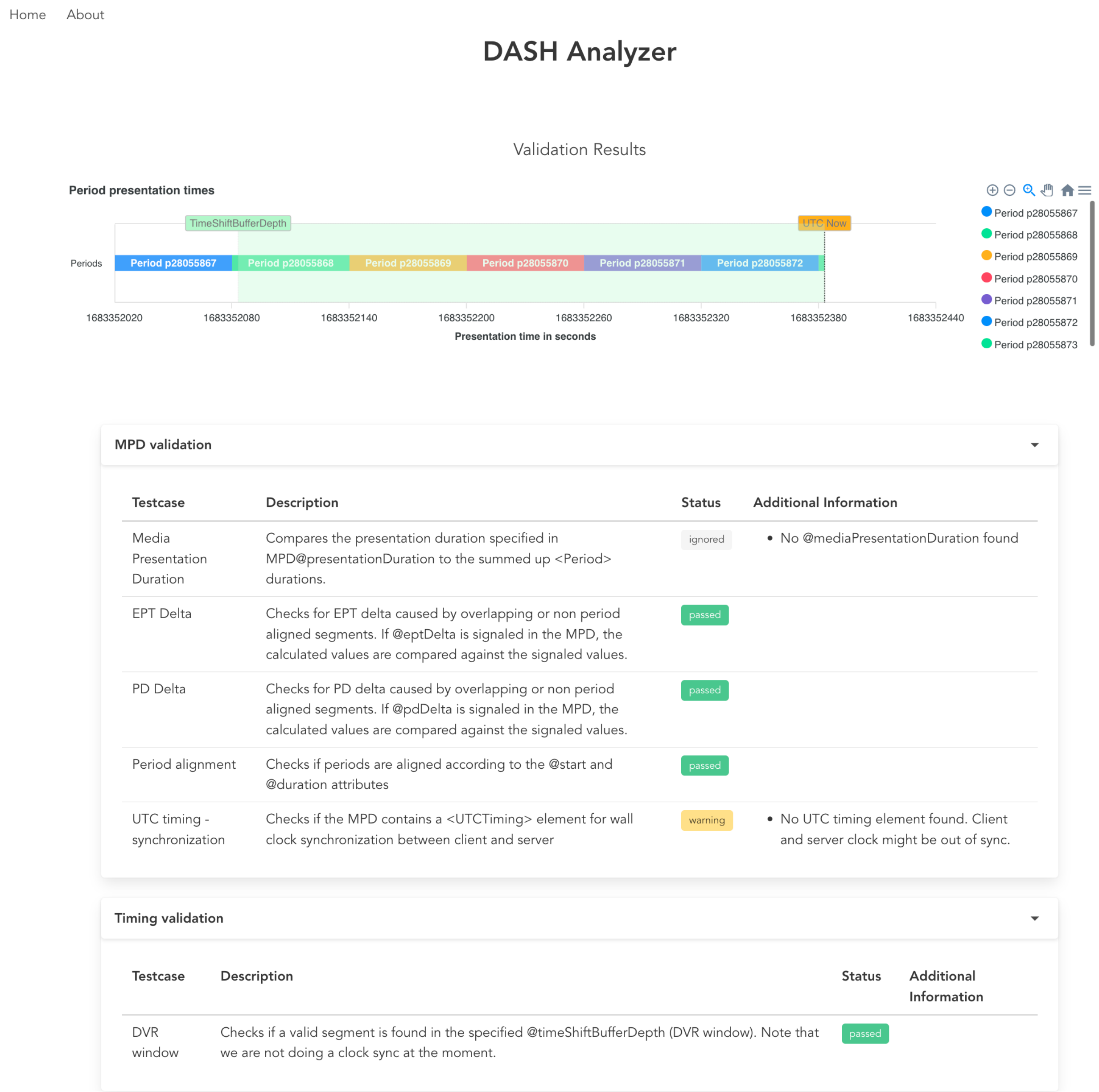
- “Chrome extension that can render mp4 boxes in the Network tab”
- Features
  - Render mp4 boxes
  - Side-by-side mp4 box comparison
  - Download and concatenate segments
- https://github.com/bitmovin/MP4Inspector

- Another tool: https://dev.to/video/mp4ff-beyond-mp4-boxes-2bee

How to debug your streams

Timing validation

- Small internal tool that checks for overlaps of segments in periods
- Can be useful to find the reason for gaps in the content



Home About

DASH Analyzer

Validation Results

Period presentation times

MPD validation

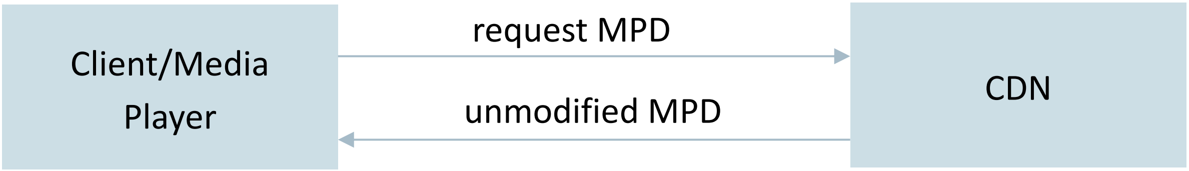
Testcase	Description	Status	Additional Information
Media Presentation Duration	Compares the presentation duration specified in MPD@presentationDuration to the summed up <Period> durations.	ignored	• No @mediaPresentationDuration found
EPT Delta	Checks for EPT delta caused by overlapping or non period aligned segments. If @eptDelta is signaled in the MPD, the calculated values are compared against the signaled values.	passed	
PD Delta	Checks for PD delta caused by overlapping or non period aligned segments. If @pdDelta is signaled in the MPD, the calculated values are compared against the signaled values.	passed	
Period alignment	Checks if periods are aligned according to the @start and @duration attributes	passed	
UTC timing - synchronization	Checks if the MPD contains a <UTCTiming> element for wall clock synchronization between client and server	warning	• No UTC timing element found. Client and server clock might be out of sync.

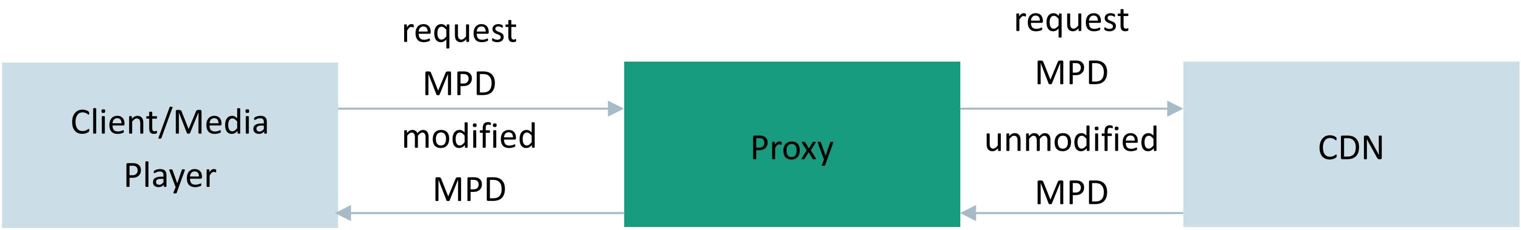
Timing validation

Testcase	Description	Status	Additional Information
DVR window	Checks if a valid segment is found in the specified @timeShiftBufferDepth (DVR window). Note that we are not doing a clock sync at the moment.	passed	

MPD proxy

- Idea: We can adjust the <BaseURL> and the <Location> element to point to a local proxy
- On the proxy we can modify the MPD, for instance remove a specific AdaptationSet or a specific attribute
- Allows us to break down a problem into smaller pieces. For instance, play only the video AS and check if removing audio changes anything

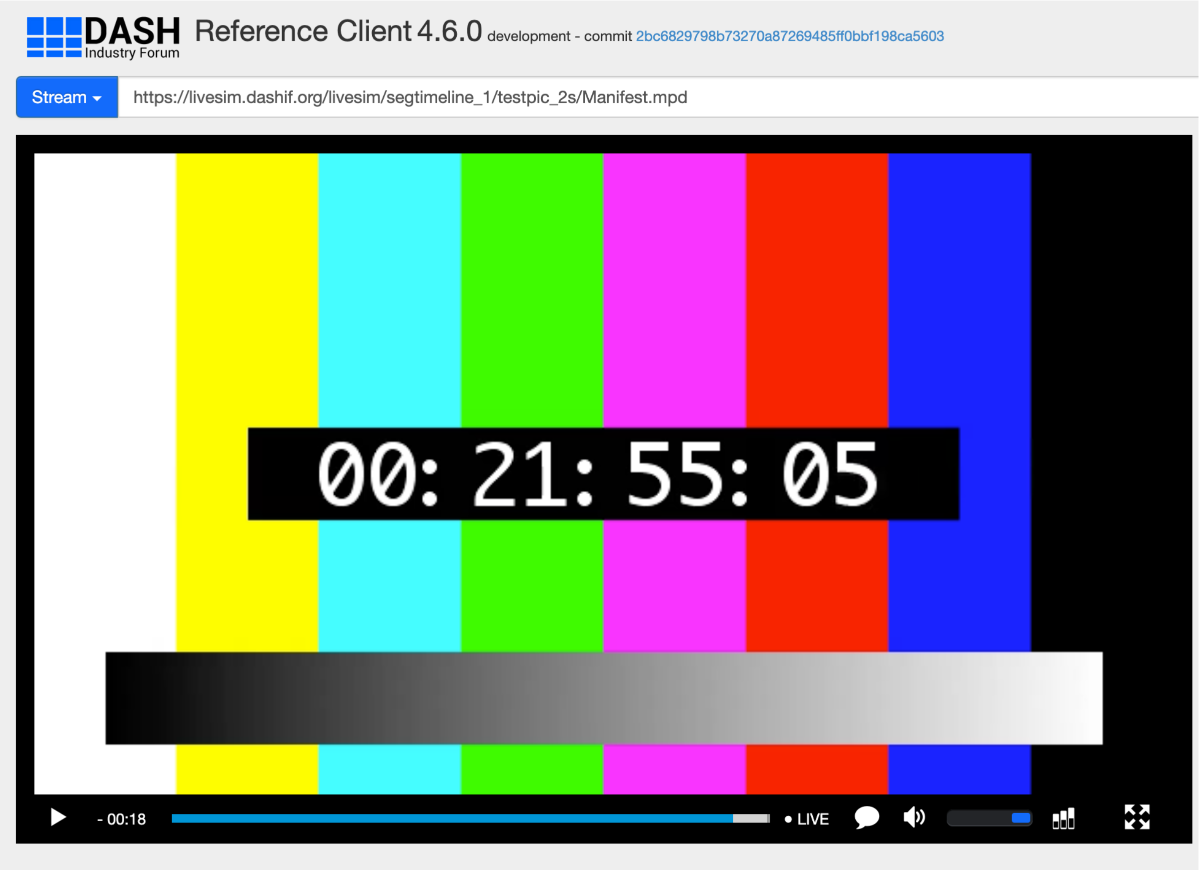




How to debug your streams

DASH-IF Live Simulator

- Creates a reference stream that can be customized for several use cases
- Useful tool to quickly setup a reference stream for comparison and testing
- Used VoD content and modified the MPD and the media segments to provide a live source
- Various configuration options: https://github.com/Dash-Industry-Forum/dash-live-source-simulator/wiki#complete-list-of-options
- Version 1: https://github.com/Dash-Industry-Forum/dash-live-source-simulator
- Work on version 2 has started: https://github.com/Dash-Industry-Forum/livesim2
- Sample streams are hosted on https://livesim.dashif.org/



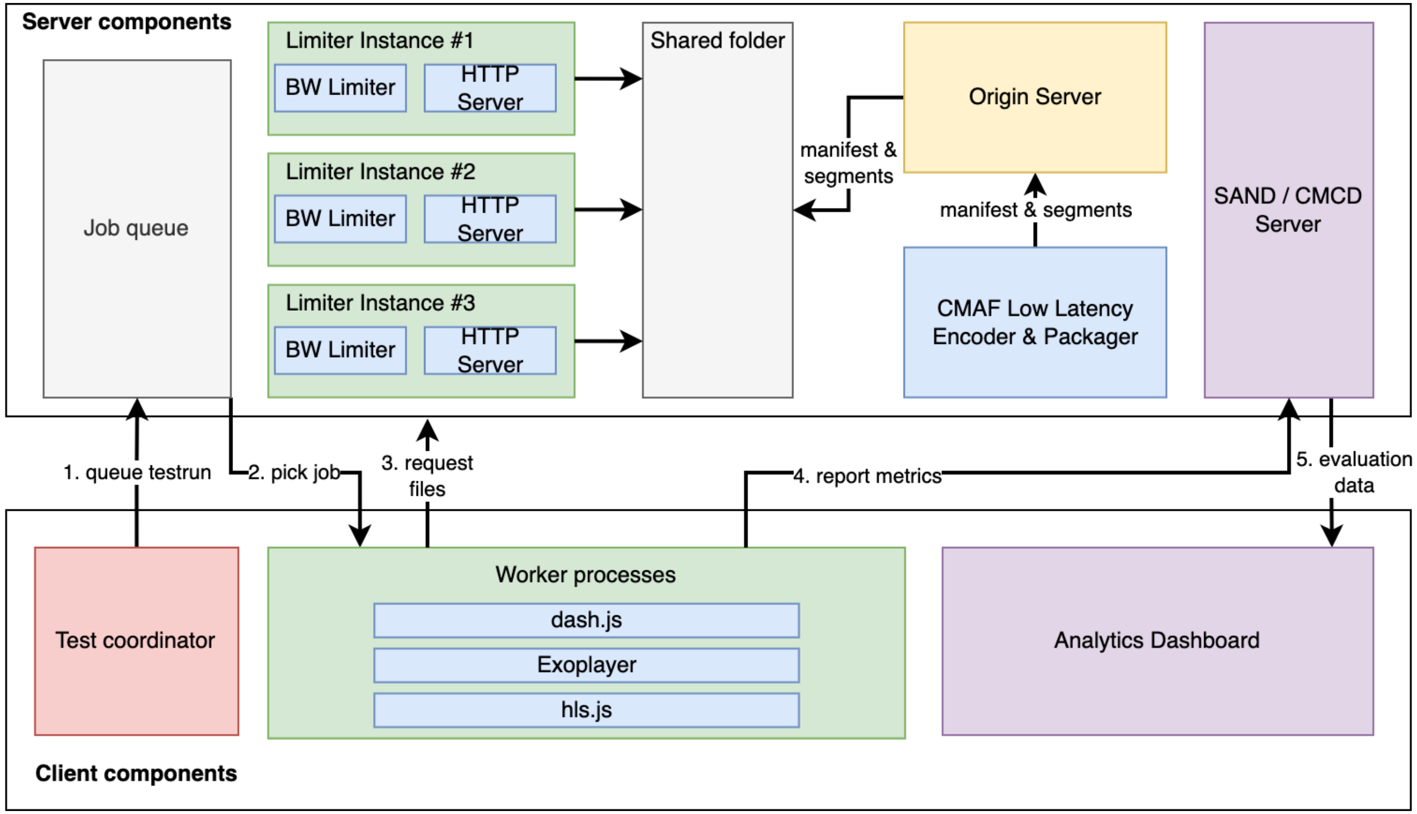
DASH Industry Forum Reference Client 4.6.0 development - commit 2bc6829798b73270a87269485ff0bbf198ca5603

Stream ▾ https://livesim.dashif.org/livesim/segtimeline_1/testpic_2s/Manifest.mpd

How to debug your streams

ABR Testbed

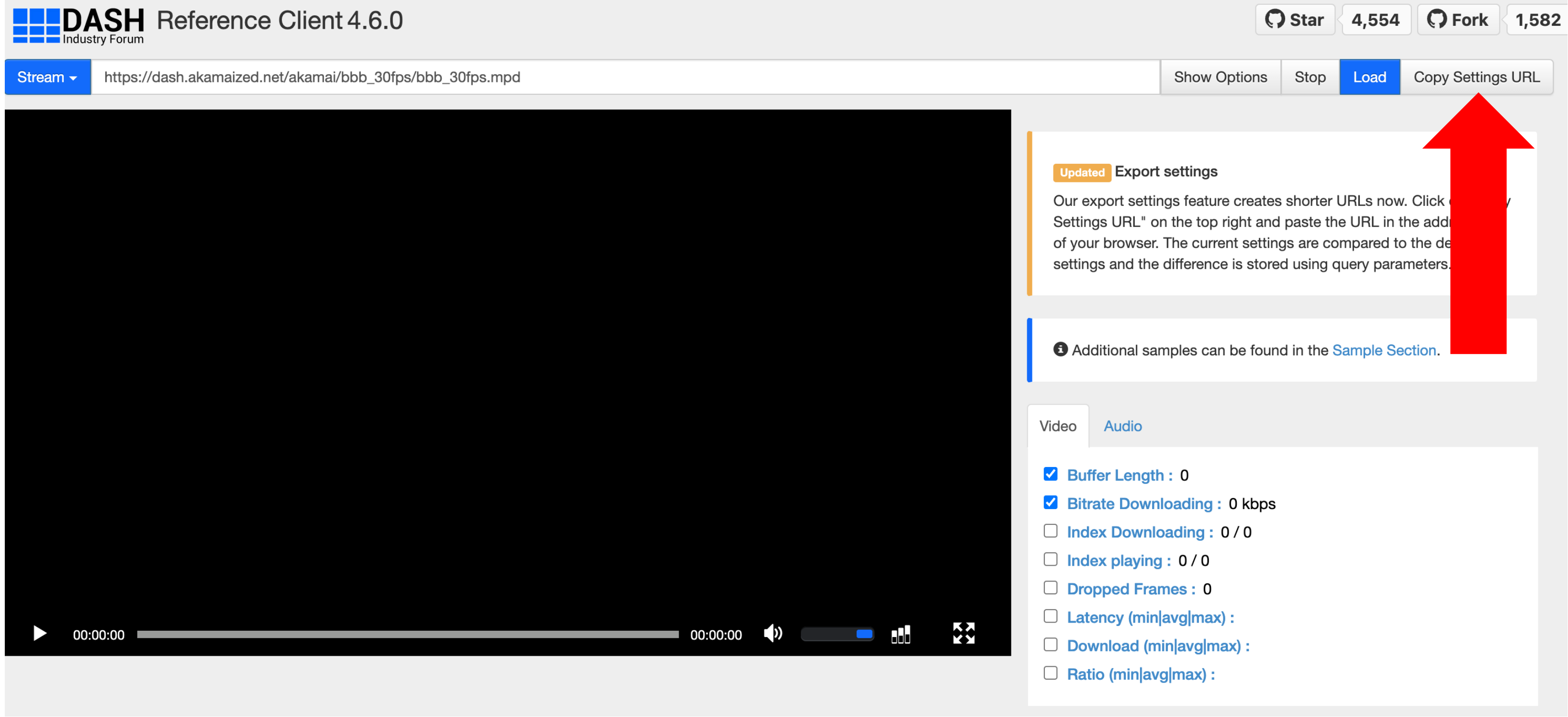
- Test different streams and different players under various network conditions
- Make sure that the ABR algorithms behave in an optimal way



How to debug your streams

Recommendations / Best practices / Hints

- The Reference UI of dash.js allows exporting the settings to be shared via a URL. Enables configuration of the player and sharing the configuration with other developers.



https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd

Updated Export settings

Our export settings feature creates shorter URLs now. Click o Settings URL" on the top right and paste the URL in the add of your browser. The current settings are compared to the de settings and the difference is stored using query parameters.

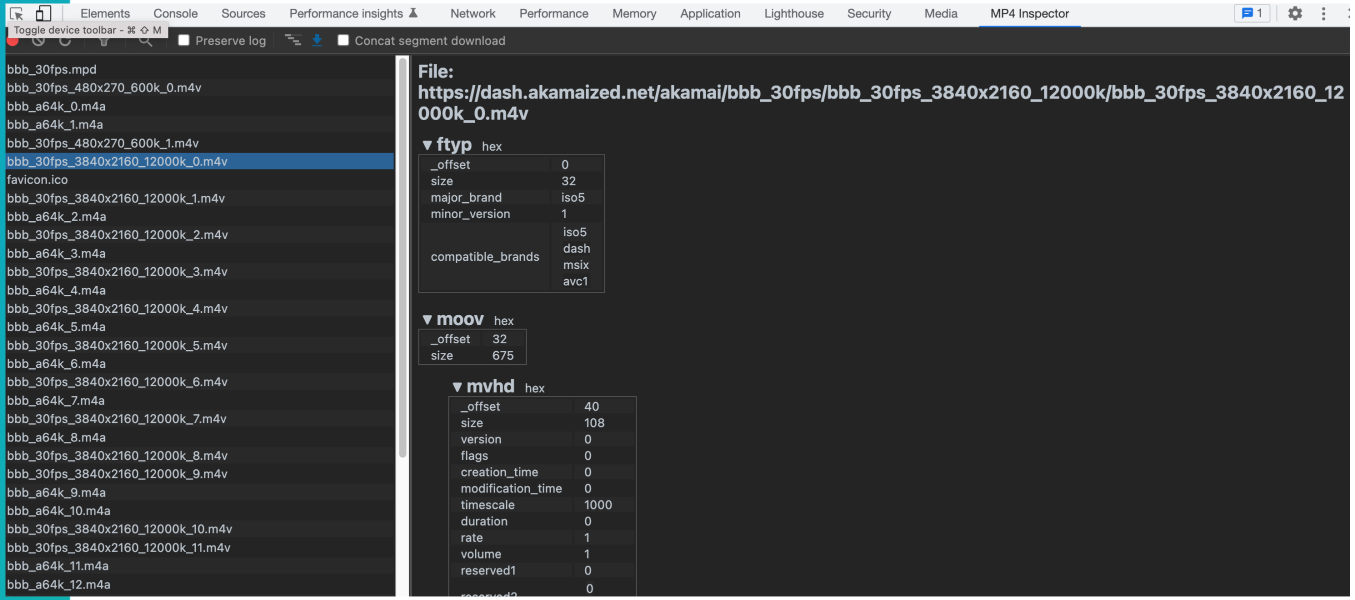
Additional samples can be found in the Sample Section.

☑ Buffer Length : 0
☑ Bitrate Downloading : 0 kbps
☐ Index Downloading : 0 / 0
☐ Index playing : 0 / 0
☐ Dropped Frames : 0
☐ Latency (min|avg|max) :
☐ Download (min|avg|max) :
☐ Ratio (min|avg|max) :

Segment Inspection

Bitmovin’s mp4 inspector

Try it out yourself:
https://bitmovin.com/mp4inspector/



```
bbb_30fps.mpd
bbb_30fps_480x270_600k_0.m4v
bbb_a64k_0.m4a
bbb_a64k_1.m4a
bbb_30fps_480x270_600k_1.m4v
bbb_30fps_3840x2160_12000k_0.m4v
favicon.ico
bbb_30fps_3840x2160_12000k_1.m4v
bbb_a64k_2.m4a
bbb_30fps_3840x2160_12000k_2.m4v
bbb_a64k_3.m4a
bbb_30fps_3840x2160_12000k_3.m4v
bbb_a64k_4.m4a
bbb_30fps_3840x2160_12000k_4.m4v
bbb_a64k_5.m4a
bbb_30fps_3840x2160_12000k_5.m4v
bbb_a64k_6.m4a
bbb_30fps_3840x2160_12000k_6.m4v
bbb_a64k_7.m4a
bbb_30fps_3840x2160_12000k_7.m4v
bbb_a64k_8.m4a
bbb_30fps_3840x2160_12000k_8.m4v
bbb_30fps_3840x2160_12000k_9.m4v
bbb_a64k_9.m4a
bbb_a64k_10.m4a
bbb_30fps_3840x2160_12000k_10.m4v
bbb_30fps_3840x2160_12000k_11.m4v
bbb_a64k_11.m4a
bbb_a64k_12.m4a
```

File:
https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps_3840x2160_12000k/bbb_30fps_3840x2160_12000k_0.m4v

▾ ftyp hex

_offset	0
size	32
major_brand	iso5
minor_version	1
compatible_brands	iso5 dash msix avc1

▾ moov hex

_offset	32
size	675

▾ mvhd hex

_offset	40
size	108
version	0
flags	0
creation_time	0
modification_time	0
timescale	1000
duration	0
rate	1
volume	1
reserved1	0
reserved2	0

Chapter 09

dash.js - What‘s next?

What’s next

CMSD

- Common Media Server Data
- Developed within CTA-WAVE
- Standard by which every media server can communicate data with each media object response
- Processing of the data can be done by intermediate server and players
- Example: Server provides estimated throughput to be used by the client for ABR decisions
- https://github.com/cta-wave/common-media-server-data

ABR rework

- Improve throughput calculation by offering additional configuration options (sample size, weights, mean calculation)
- Add support for “urn:mpeg:dash:adaptation-set-switching:2016”
- Refactor whole ABR decisioning logic

Other items

- Improved XML parsing (speed improvements on low end devices)
- New reference UI
- Support for forced-subtitles
- MSE in webworkers

- Feedback thread: https://github.com/Dash-Industry-Forum/dash.js/discussions/4111

10th FOKUS Media Web Symposium

MARK YOUR CALENDAR!

Fraunhofer
FOKUS

June 13 – 14, 2023, Berlin

Advanced Streaming Technologies: DASH, HLS, SAND, Low Latency Streaming, Content Steering, Media Delivery in 5G/6G, HbbTV, Video Player Tech, DRM, Quality of Experience, Edge and Cloud processing, Remote Rendering, Green Streaming

Artificial Intelligence for Media: Generative AI, AI-based-Media-Encoding, Streaming Analytics, Content Analytics and Metadata, AI based Media Solutions, Content Provenance and Authenticity

Media Applications and Services: Metaverse, Addressable TV, Dynamic Ad Insertion/Substitution, Audience Measurement, Programmatic Advertisement, Holo Conferencing, XR

www.fokus.fraunhofer.de/go/mws

Contact

Daniel Silhavy

- Email: daniel.silhavy@fokus.fraunhofer.de
- LinkedIn: https://www.linkedin.com/in/daniel-silhavy-21650a129/

Fraunhofer FOKUS
Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
info@fokus.fraunhofer.de
www.fokus.fraunhofer.de