Fraunhofer FOKUS

Fraunhofer FOKUS Institute for Open Communication Systems

# Media Players and Tools

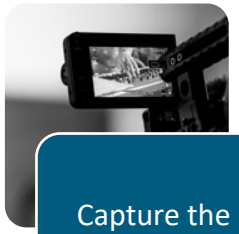
Daniel Silhavy

# Agenda

1. Foundations
2. dash.js Overview
3. dash.js Features
4. Native Media Players
5. Media Player Testing
6. How to debug media streams

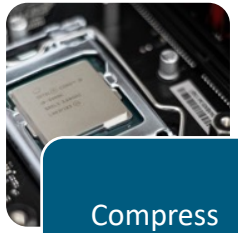
# Overview - Media Streaming Chain

**Media Input**



Capture the media input (video, audio)

**Encoding**



Compress media using codecs

**Packaging**

Create media segments and manifest files

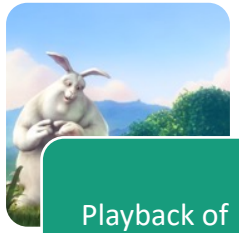
**DRM Protection**

Protect the content against piracy

**Hosting**

Host media segments and the manifest files on a CDN

**Playback**



Playback of the content on the client side

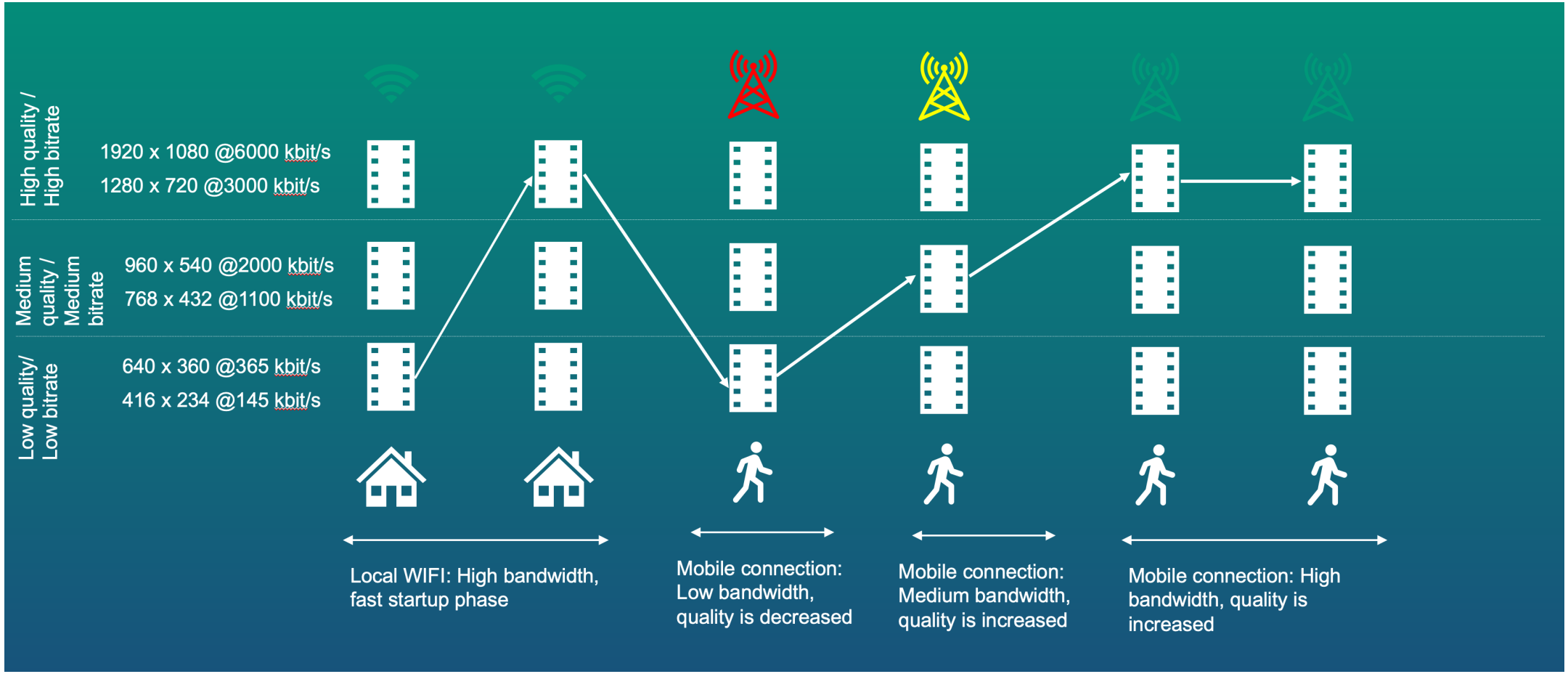
**Monitoring**

Monitor streaming performance and detect errors and failures

Chapter 01

# Foundations

# Adaptive Bitrate (ABR) Streaming



# Types of browser-based media playback

HbbTV

Shaka Player

dash.js

## Type 1 Playback

- 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 Playback

- HTML5 video element but ABR API to control ABR logic of the player
- Not really used by the industry

## Type 3 Playback

- 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

# MSE and EME for Type 3 Playback

## W3C Media Source Extensions API

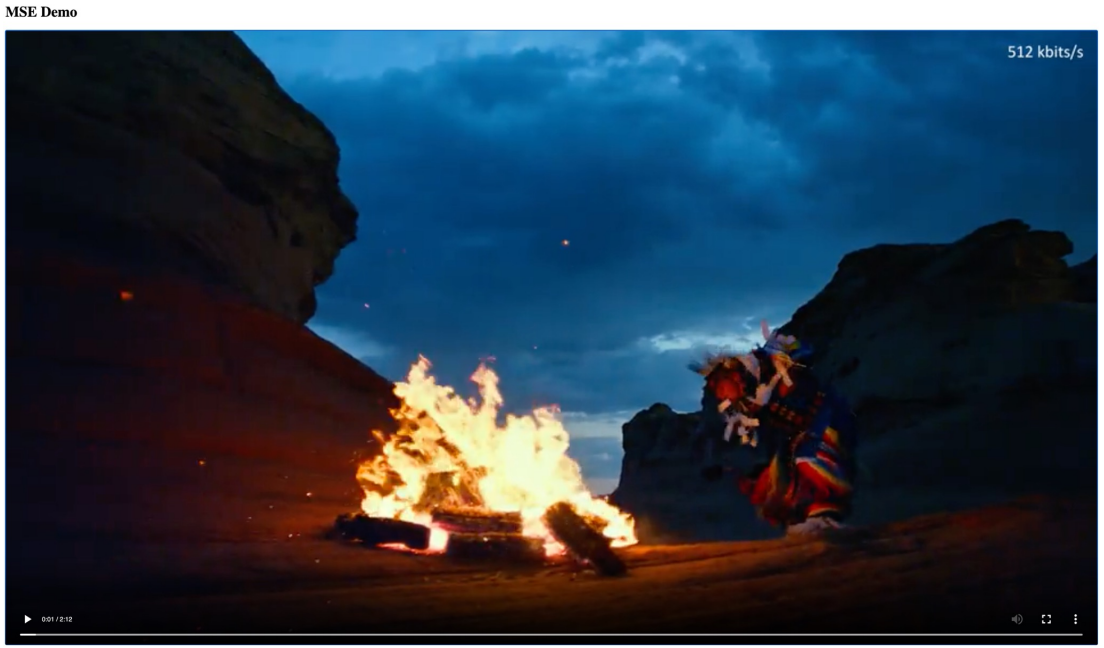
- https://w3c.github.io/media-source/
- Extends the HTML5 Media Element to allow JavaScript applications to generate media streams for playback of adaptive streaming content.
- Provides the necessary APIs to manage the media buffer (append and remove)

## W3C Encrypted Media Extensions API

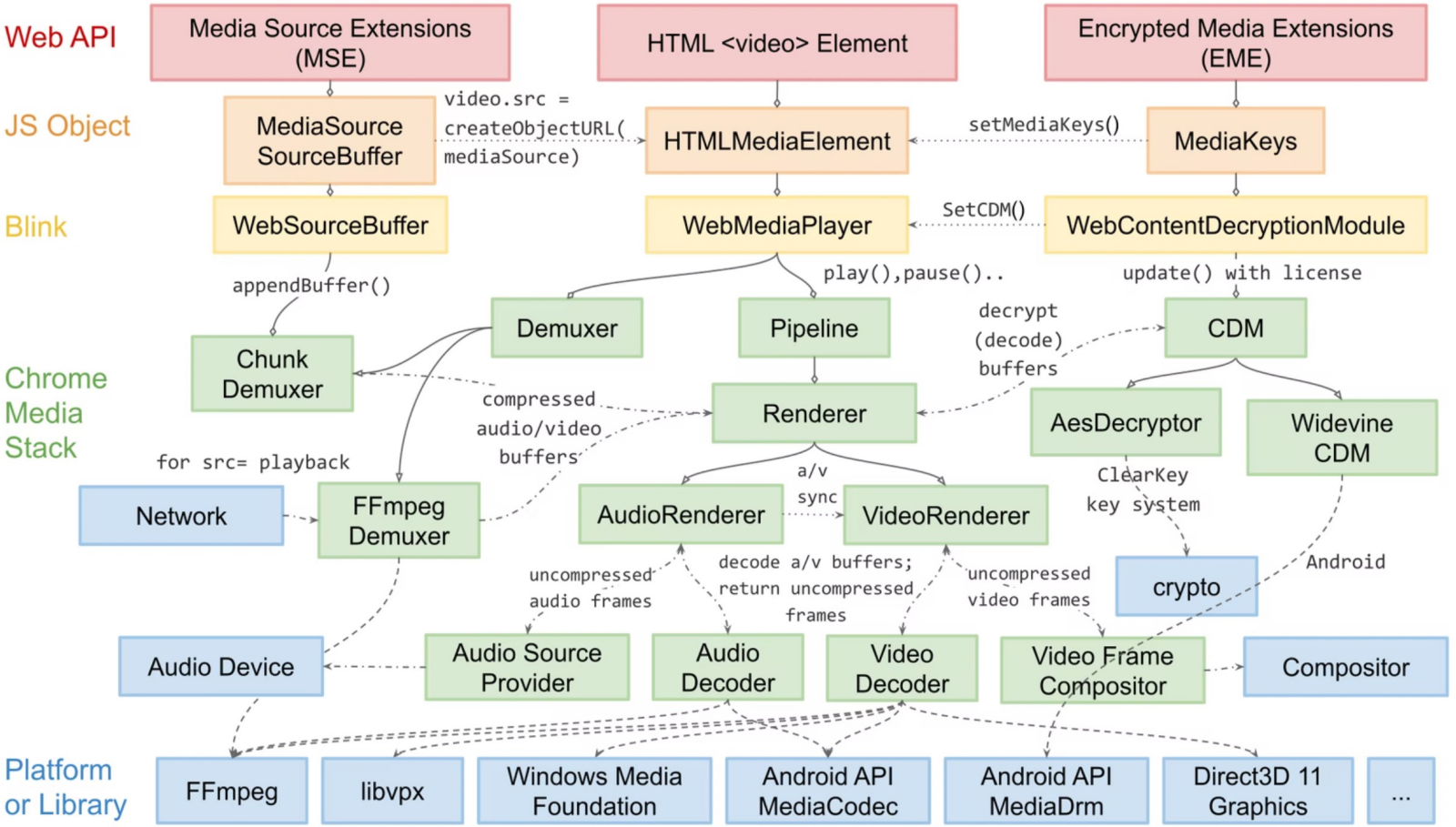
- https://www.w3.org/TR/encrypted-media/
- Extends the HTML5 Media Element providing APIs to control the playback of DRM protected content
- Complements the MSE for Type-3 playback

# W3C Media Source Extensions – Live Demo

```
// 1. Check if MSE is supported
if (!window.MediaSource) {
    console.error('No Media Source API available');
    return;
}

// 2. Setup the MSE
var ms = new MediaSource();
video.src = window.URL.createObjectURL(ms);

// 3. Wait for the setup to be finished
ms.addEventListener('sourceopen', onMediaSourceOpen);

// 4. When MSE is setup, create a Sourcebuffer for the video content
function onMediaSourceOpen() {
    sourceBuffer = ms.addSourceBuffer('video/mp4; codecs="avc1.4d401f"');

    // 5a. As soon as a segment has been appended we can download and append the next segment
    sourceBuffer.addEventListener('updateend', nextSegment);

    // 5b. Start download: Download the init segment and append it to the buffer
    GET(initUrl, appendToBuffer);

    // 6. Start playback
    video.play();
}

// 7. Handle the download of new segments here
function nextSegment() {
    var url = templateUrl.replace('$Number$', index);
    GET(url, appendToBuffer);
    index++;
    if (index > numberOfChunks) {
        sourceBuffer.removeEventListener('updateend', nextSegment);
    }
}
```



# Browser based media streaming – The different layers



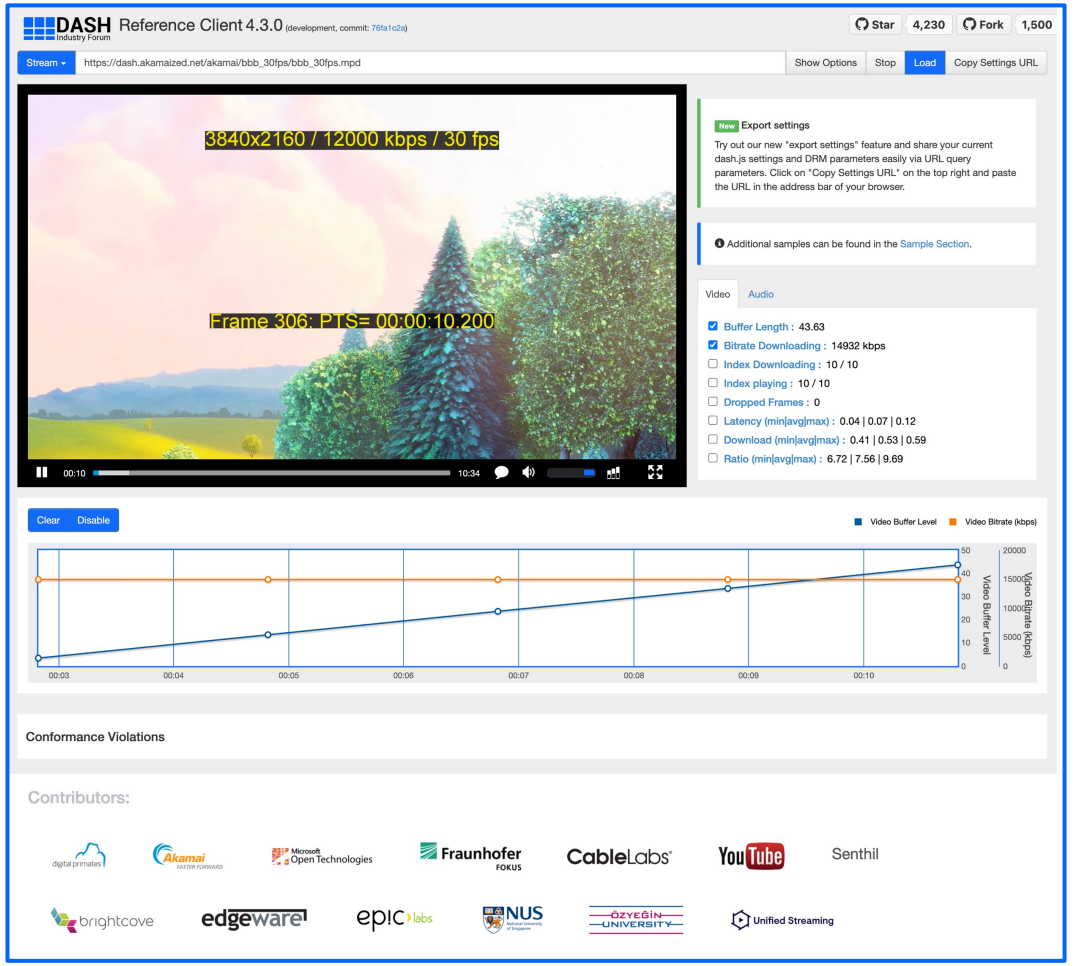

Source: https://developer.chrome.com/articles/videong/

Chapter 02

# 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 4.7.2
- 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://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

# Hands-On – Getting started

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

```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 03

# dash.js Features

# Types of ABR algorithms

## Buffer based

- Use the client‘s current buffer level to decide which quality to download
- Example: BolaRule

## Throughput based

- Use the throughput achieved in prior downloads for decision-making
- Introduces questions of mean calculation mode(arithmetic, harmonic), sample size and putting weights on the derived measurements (higher weight on most recent download)
- Example: ThroughputRule

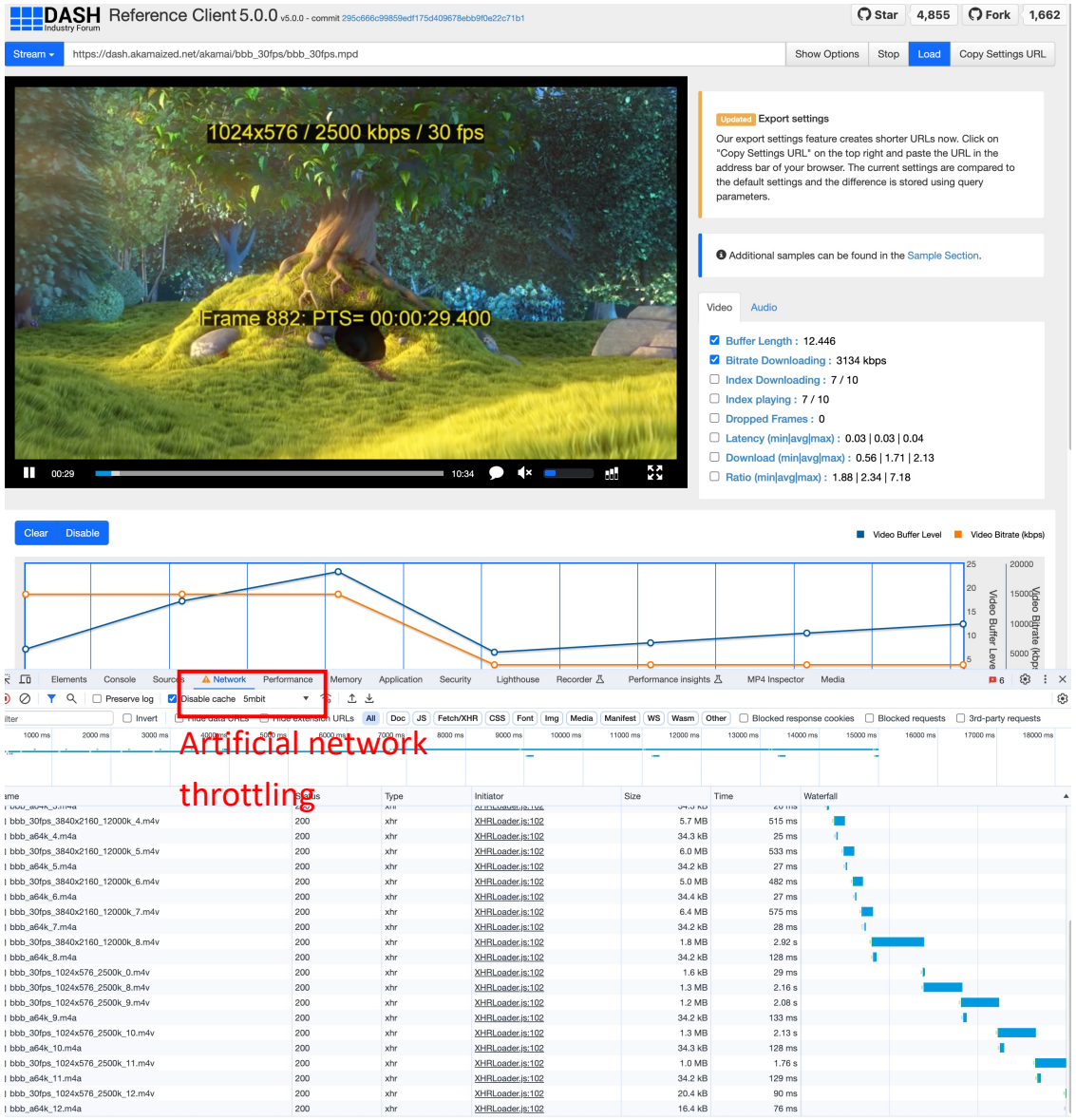
## Hybrid

- Combine both buffer and throughput information
- Example: Dynamic mode in dash.js

# Adaptive Bitrate Switching

## dash.js demo

**Try it out yourself:**
https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html



# 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
  - working on custom parameters
- 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

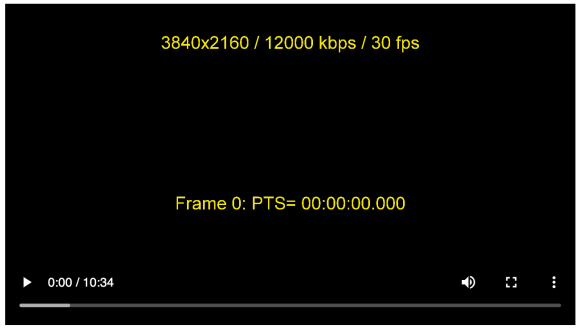
## dash.js demo

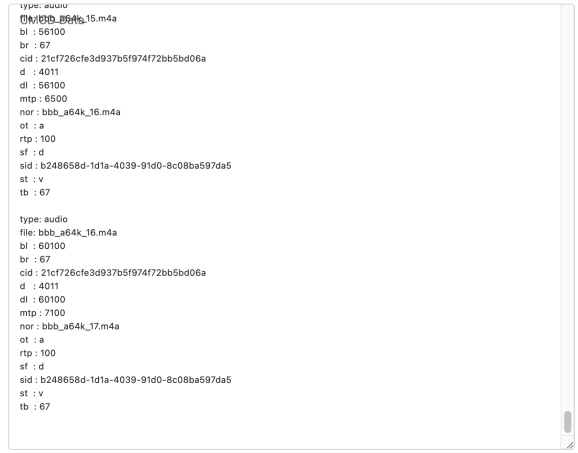
**Try it out yourself:**
https://reference.dashif.org/dash.js/nightly/samples/advanced/cmcd.html

dash.js

CMCD Reporting

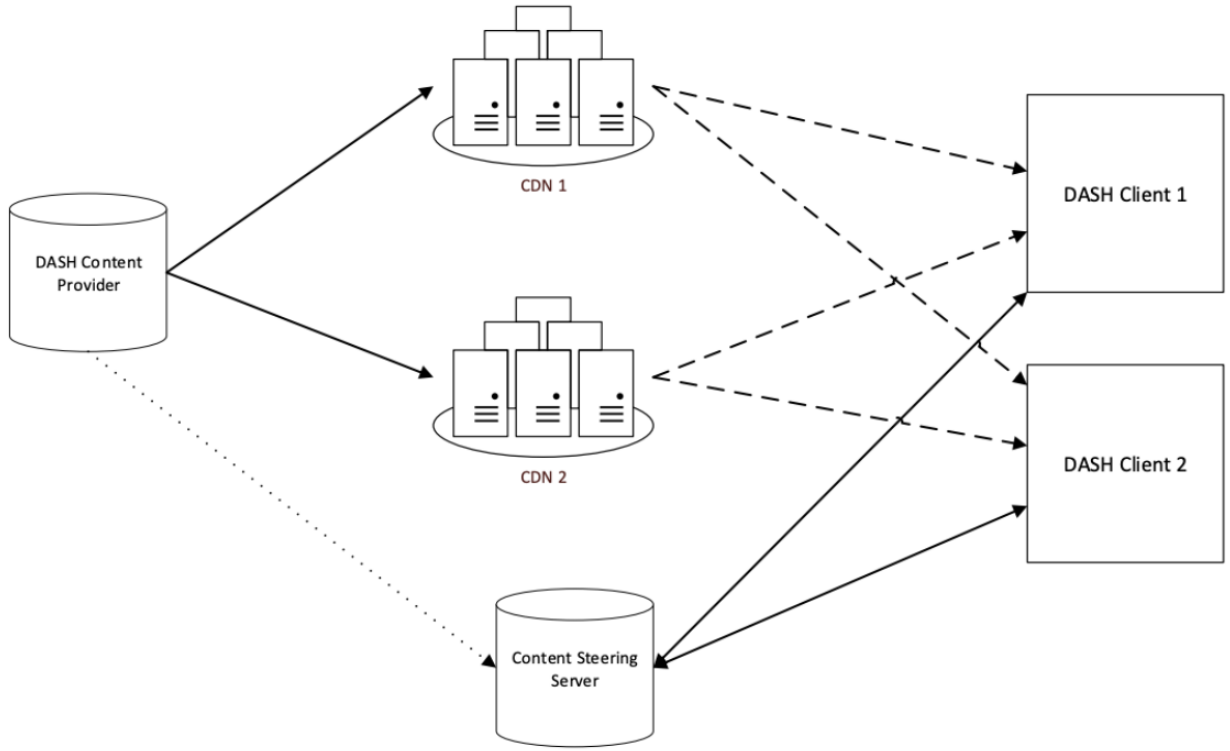
This sample shows how to use dash.js in order to enhance requests to the CDN with Common Media Client Data (CMCD - CTA 5004).



# 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

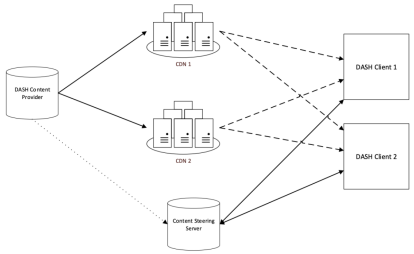
### Content Steering

**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 Load MPD


**CDN Selection**

alpha beta gamma

**Location Selection**

beta_mpd alpha_mpd delta_mpd

**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**

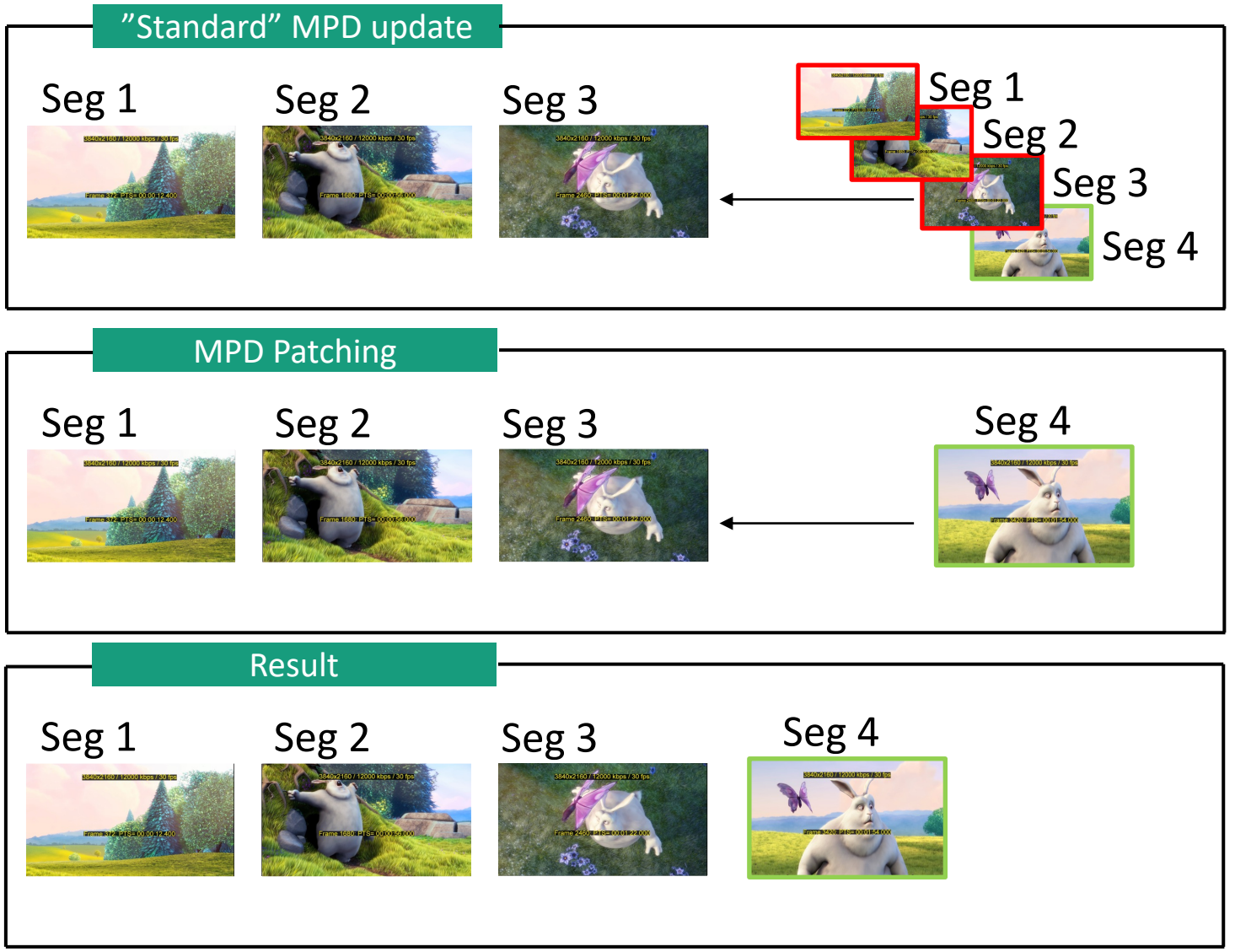
| Service Location | Request URL |
|---|---|
| alpha_mpd | http://localhost:3333/steering-content/bbb/alpha/dash.mpd |

# MPD Patching

- 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



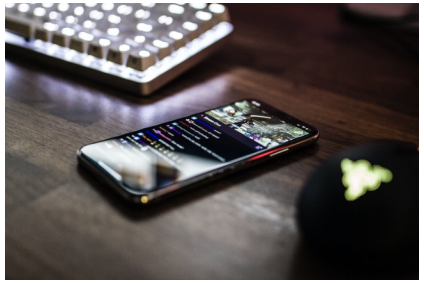
# Use cases

## Sport streaming

- Some viewers may be using a provider which distributes the content using DVB-T or DVB-S services whilst others may be using DASH ABR.
- Viewers with a high latency will have their viewing spoiled as they will hear cheers for the goal before it occurs on their local screen

## Sports betting

- The lower the latency the more opportunities for “live in play betting”
- Legal Considerations: Content cannot be shown if it is more than X seconds behind live

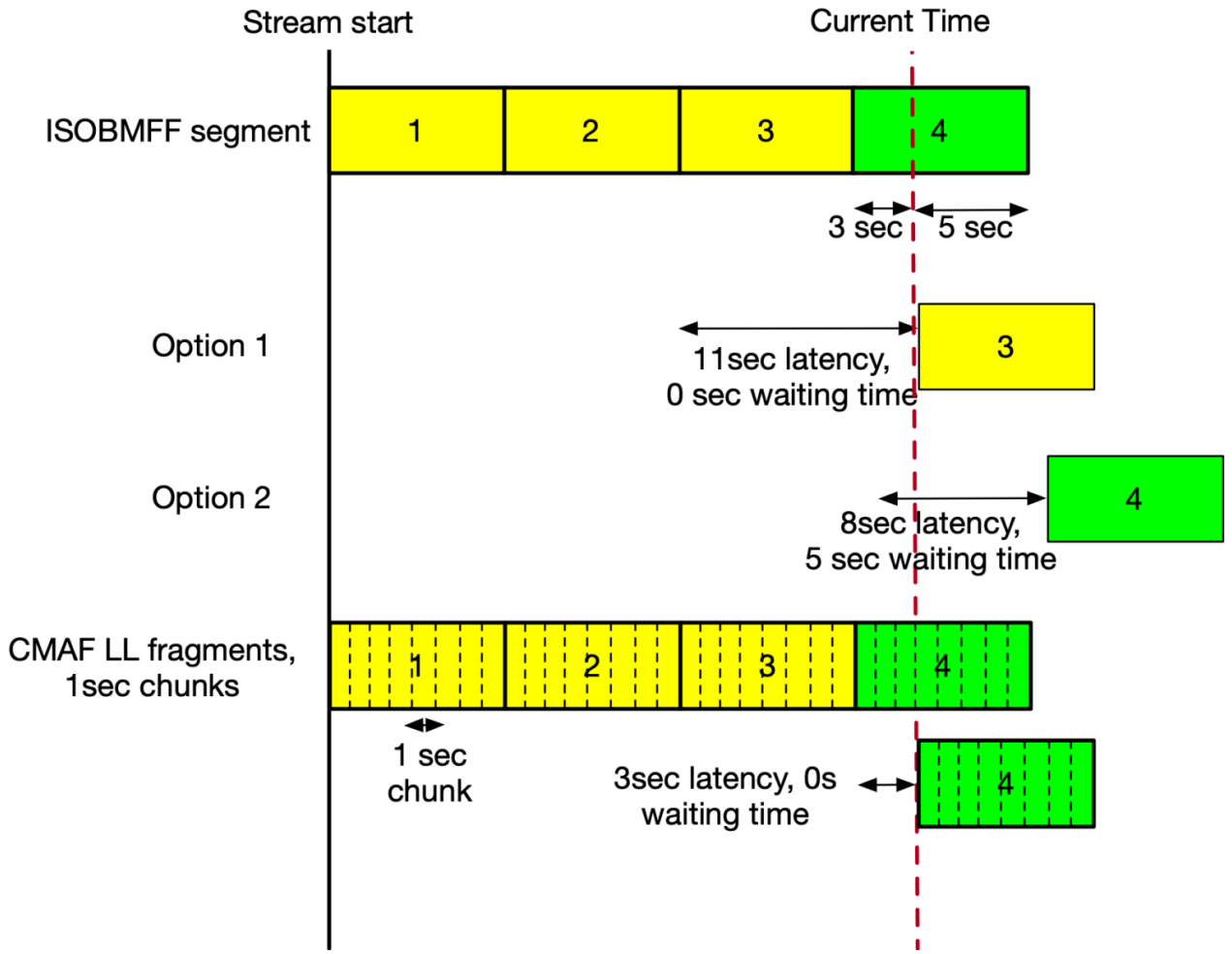


## eSports/Professional streamer

- Streamers interacting with chat directly.
- A high degree of interactivity between the streamer and the audience is required to enable engagement
- A low latency enables direct feedback from the viewers e.g applause after a song, cheering during a song.
- Example: Gamers, Musicians

# DASH Concepts

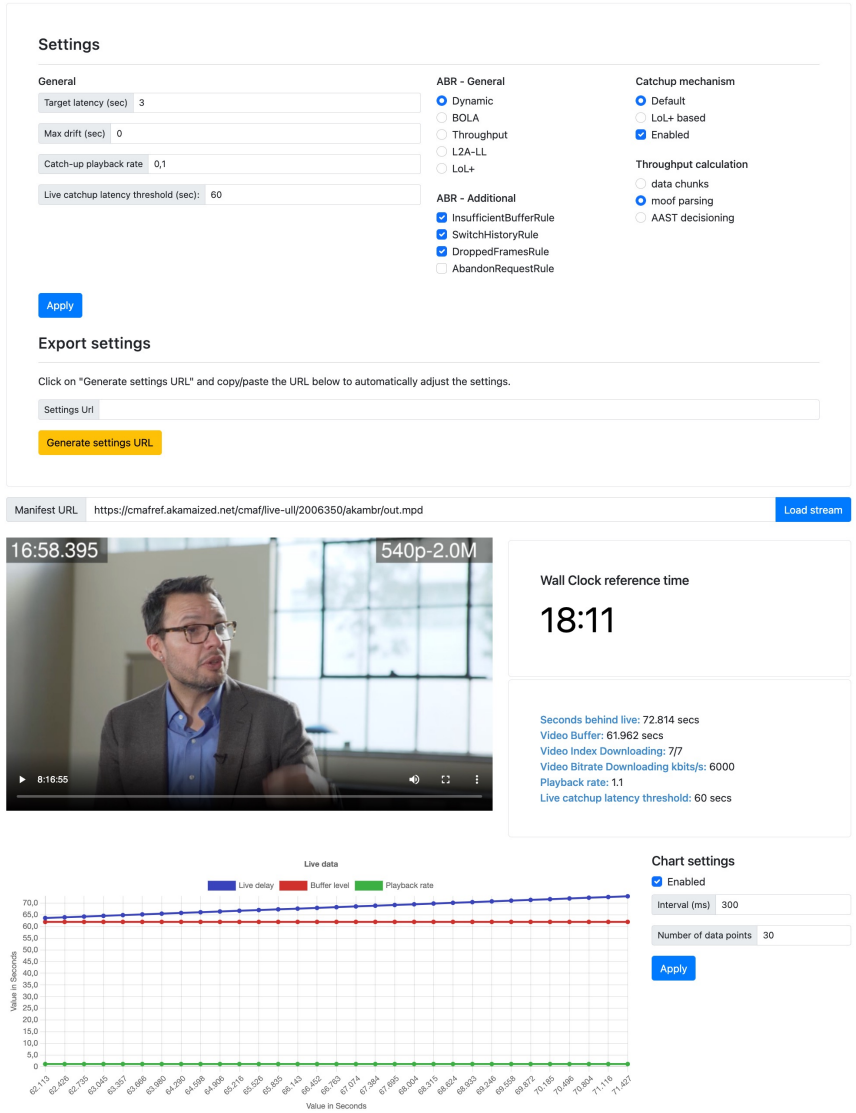
- Key concepts:
  - **CMAF chunks**
    - A CMAF fragment consists of multiple CMAF chunks
    - The CMAF chunks can be fed into the media buffer as soon as they are transferred from the server to the client
    - No need to wait for the whole CMAF Fragment to be delivered before adding data to the buffer
  - **HTTP/1.1 chunked transfer encoding (CTE)**
    - Connection is kept open and individual CMAF chunks are transferred as soon as they are ready
  - Adjustment of playback rate to maintain consistent live edge



# Low Latency Streaming

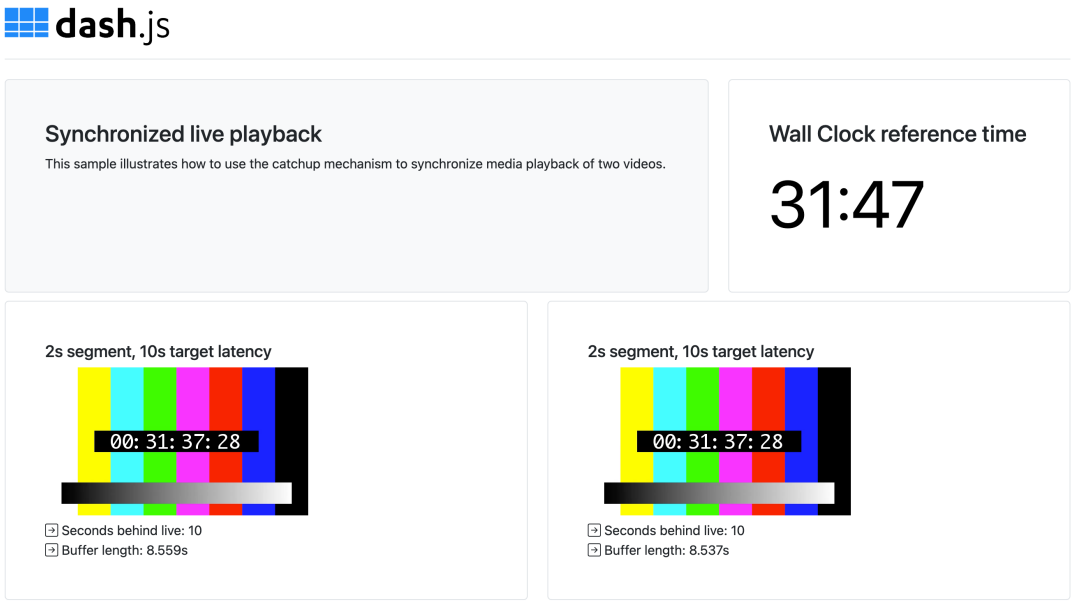
## dash.js demo

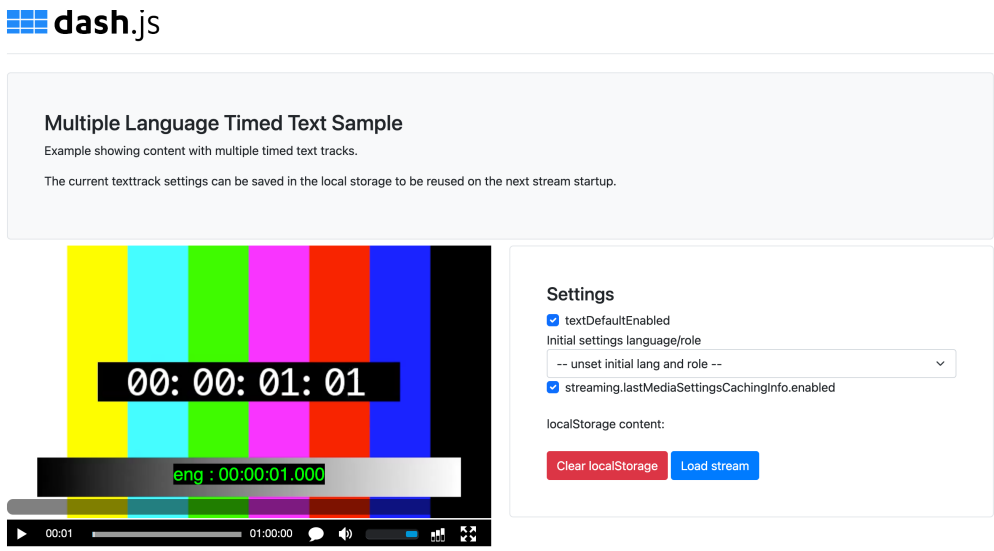
**Try it out yourself:**
https://reference.dashif.org/dash.js/nightly/samples/low-latency/testplayer/testplayer.html



# Other Demos

**Try it out yourself:**
**https://reference.dashif.org/dash.js/nightly/samples/**



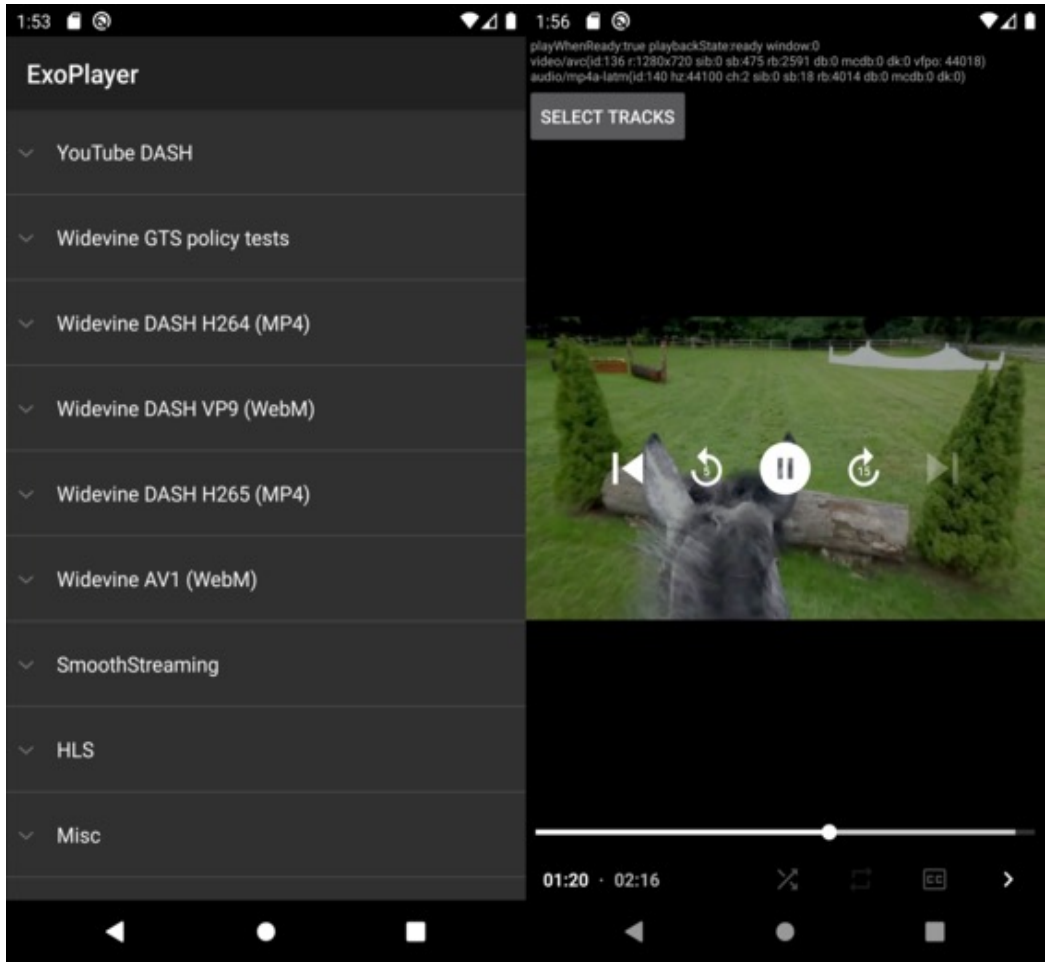



Chapter 04

# Native Media Players

# ExoPlayer

- An application-level media-player for Android.
- Created by Google
- Supports different adaptive streaming formats: DASH, HLS, SmoothStreaming
- Supports different DRM schemes: Widevine, PlayReady and ClearKey.
- Open-Source project on Github - https://github.com/google/ExoPlayer



# AVFoundation

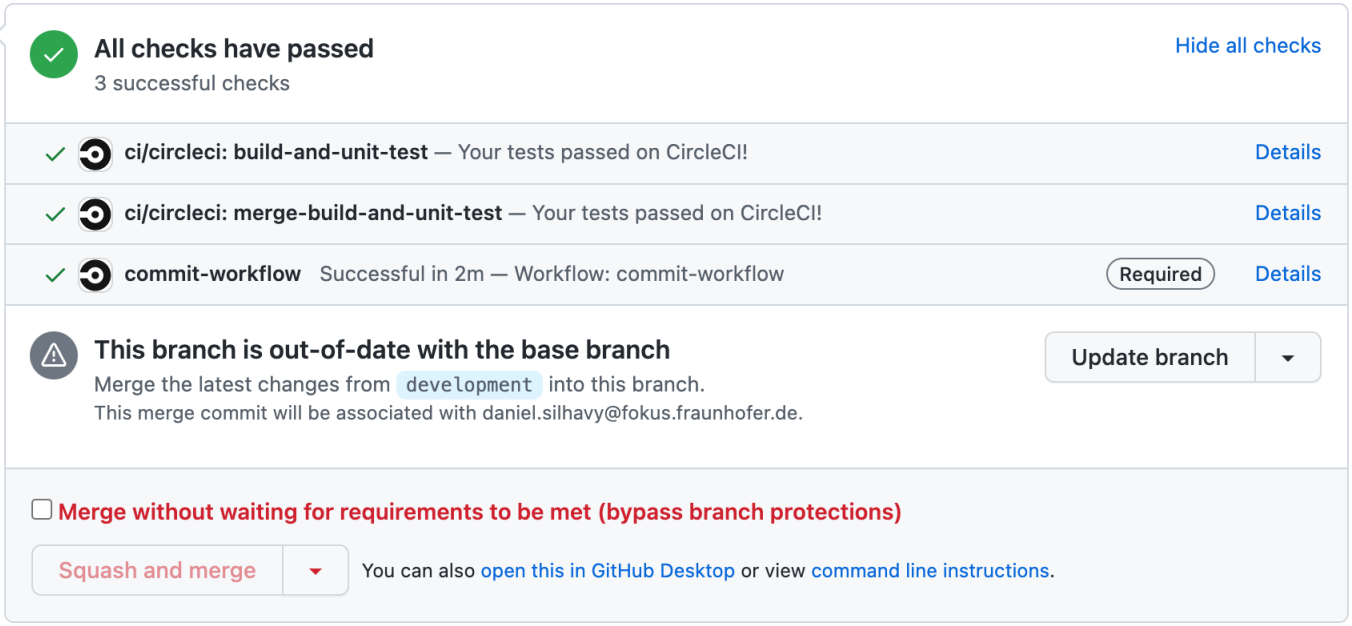
- AVFoundation is the full featured framework for working with time-based audiovisual media on iOS, macOS, watchOS and tvOS.
- Features
  - Full featured Framework for audiovisual media
  - Play, create, edit and export media files
  - Play HLS Streams easily
- AVKit
  - Integrated media player UI components

Chapter 05

# Media Player Testing

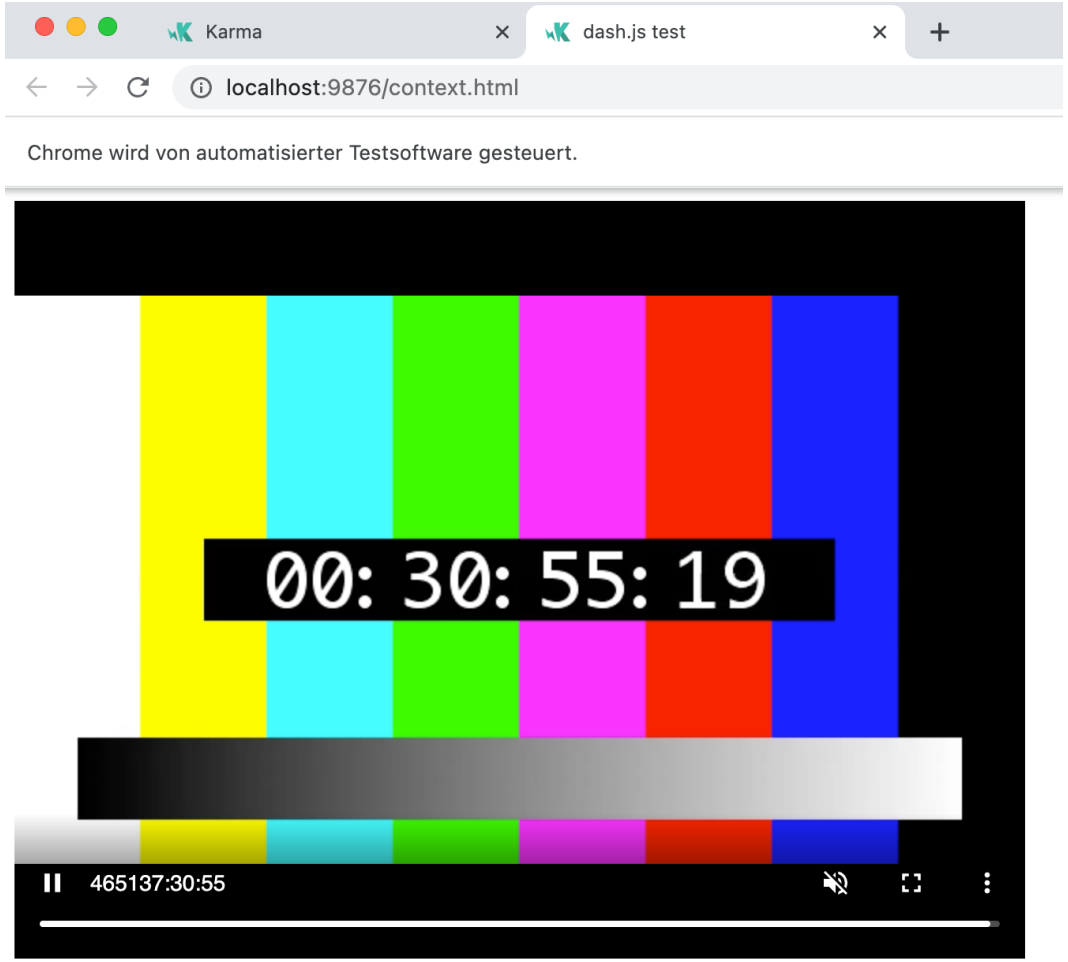
# Unit Tests

- Test individual functions or methods (units) of the code.
- Automatically triggered for each pull request, typically integrated into a CI/CD workflow.



# 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.
- Ideally replaces manual regression tests on various platforms



# Demo - Functional Tests

# Demo Report - Functional Tests

## dash.js

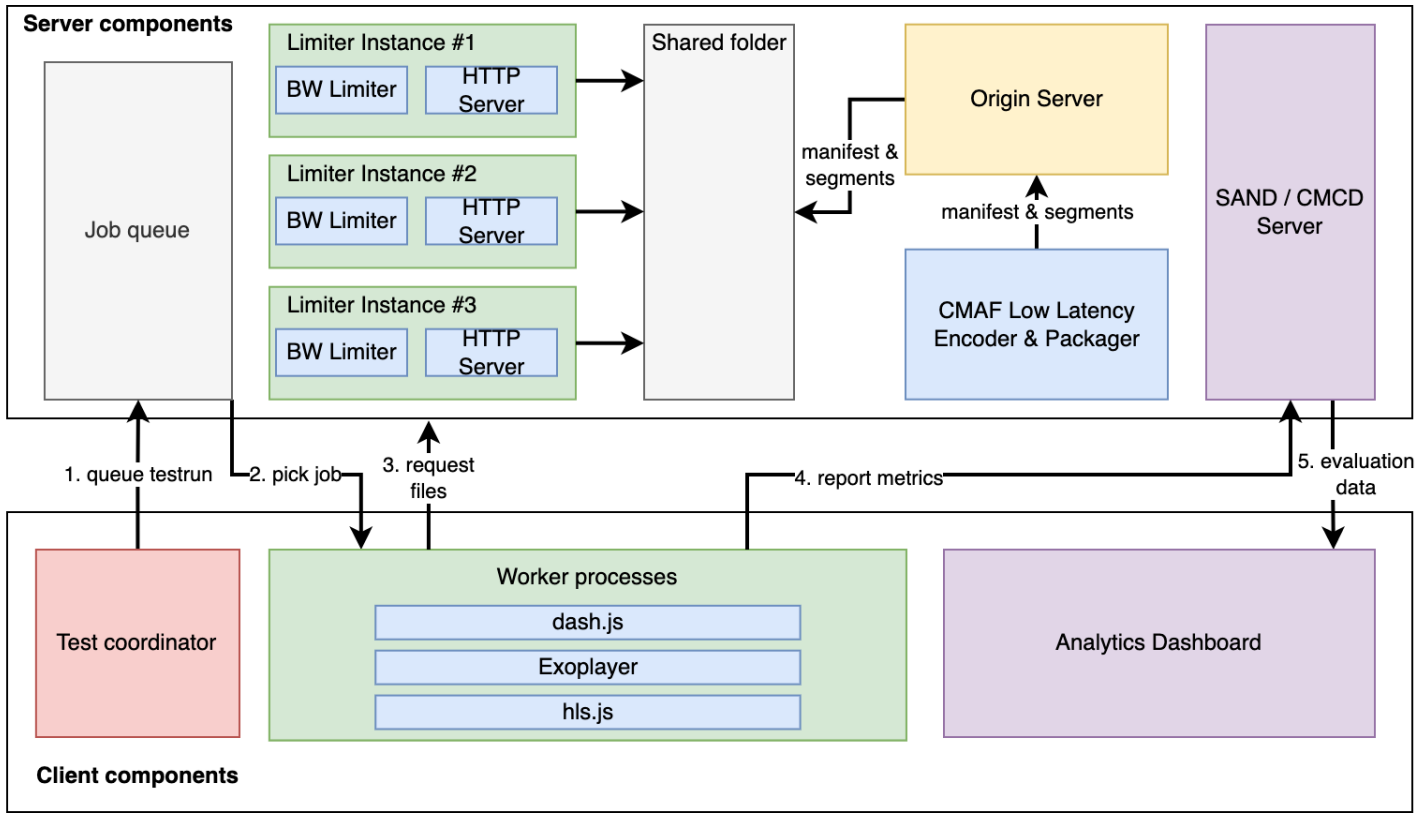
### Functional Tests

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<br>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 |

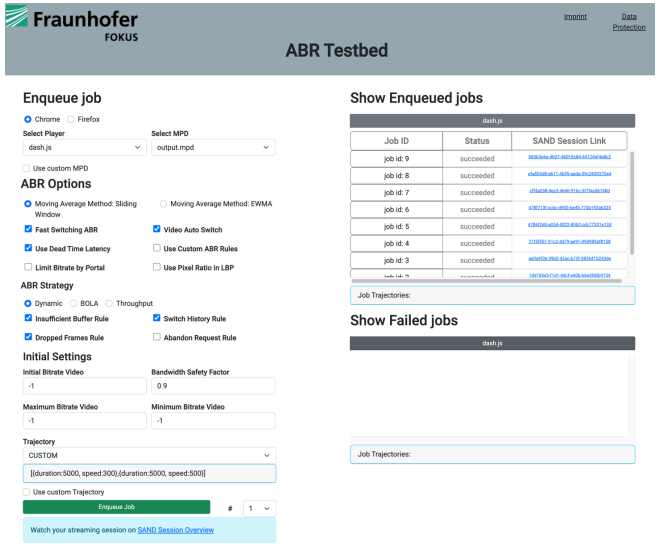
# ABR Testbed

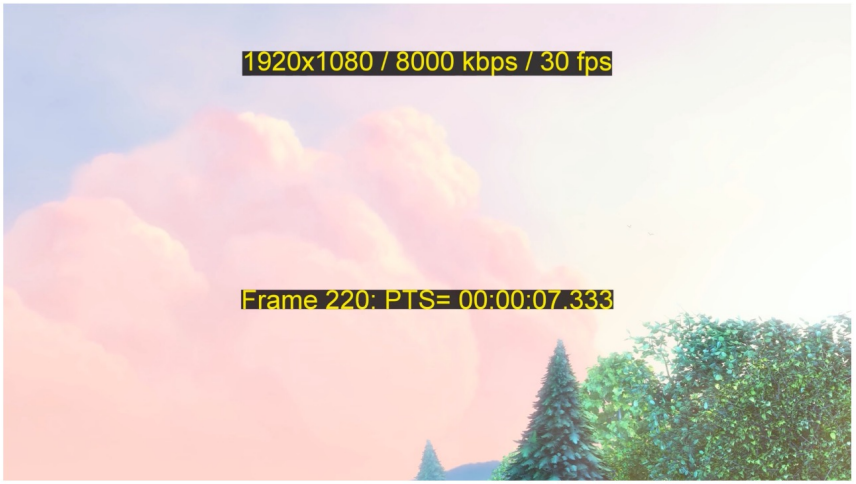
- Test different streams and different players under various network conditions
- Make sure that the ABR algorithms behave in an optimal way

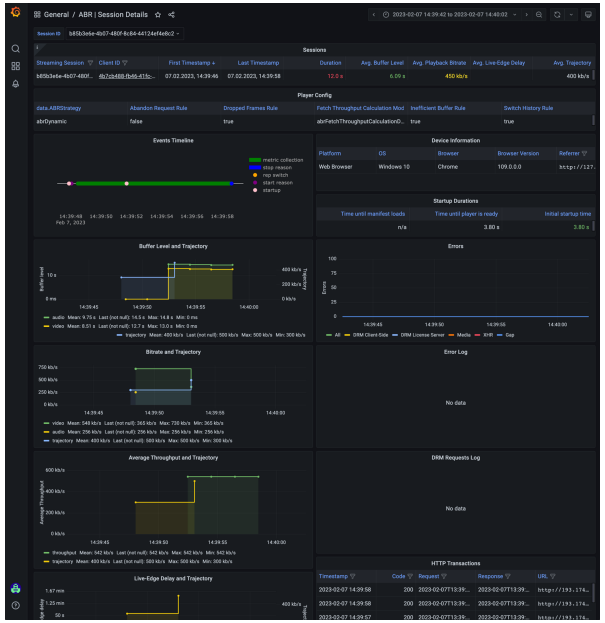


# Media Player Testing

## ABR Testbed




Define test parameters (Test coordinator)

Run test (worker process)

Evaluate result (SAND & CMCD)

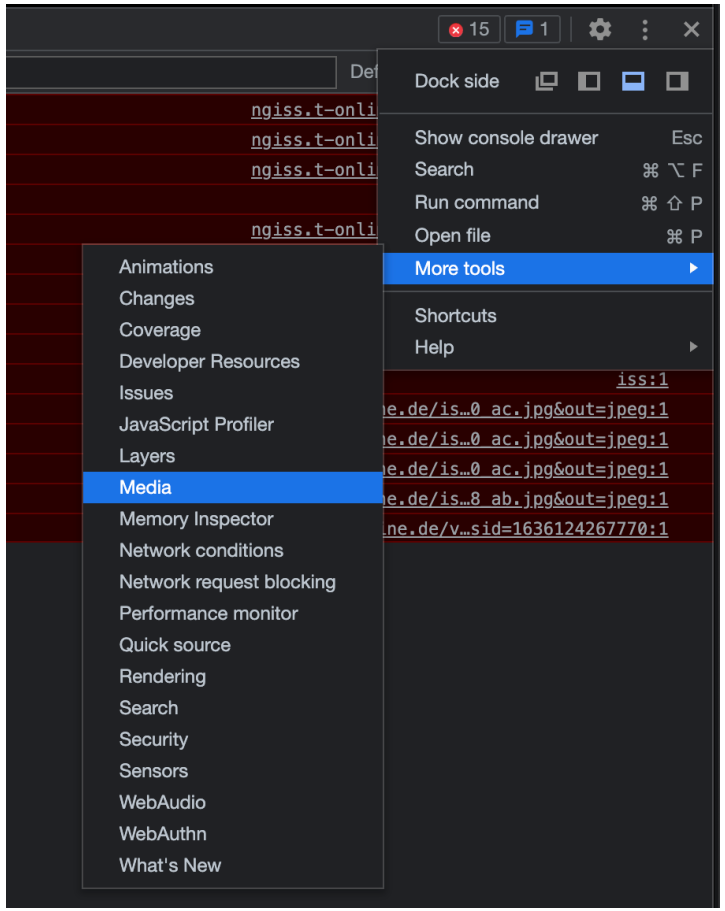
# ABR Testbed Demo Video

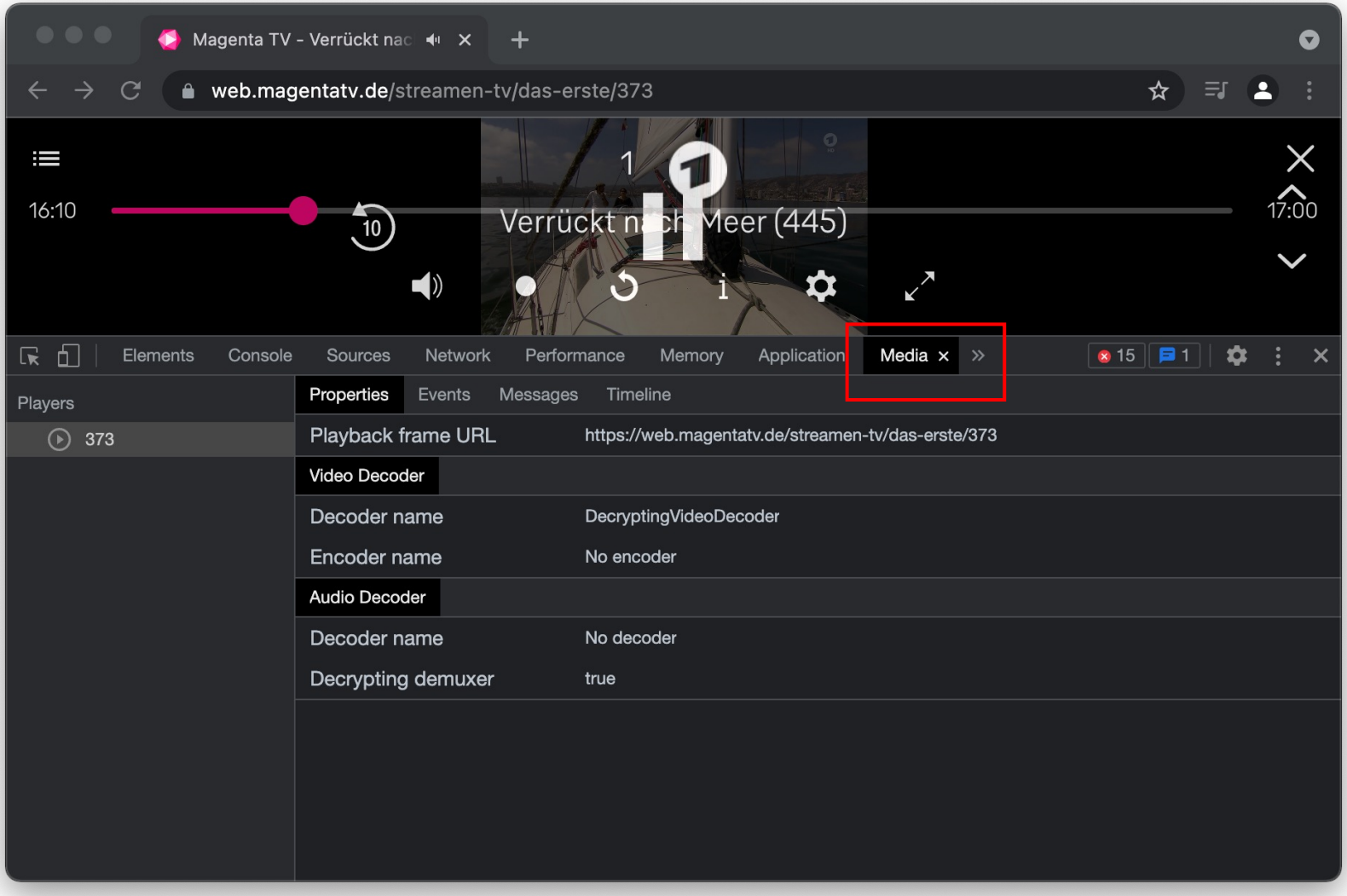
Chapter 06

# How to debug media streams

# How to debug your streams

## Browser based debugging



# Apple HLS – Media Stream Validator

- Tool to validate HLS streams
- Can be downloaded directly from Apple developer space: https://developer.apple.com/download/all/?q=media
- Typical steps
  1. Run mediastreamvalidator against the HLS primary playlist e.g. *mediastreamvalidator -t 60 https://mcdn.daserste.de/daserste/de/master.m3u8*
  2. Create an HTML5 report from the JSON report e.g. *hlsreport validation_data.json*


**HLS Validation Report** (General, tvOS, iOS, macOS, AirPlay 2)

Stream type: LIVE

https://mcdn.daserste.de/daserste/de/master.m3u8

**Variant Overview**

| Audio ID | # | Max Rate | % Diff. | PL Avg Rate | Avg Rate | % Diff | Resolution | IDR Int. | Framerate | Codec | Profile | Level | Encryption | Avg Seg Count | Avg PL Duration | Audio Track Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AUDIO | 1 | 794 | 11.1% | 636 | 573 | -9.9% | 480 x 270 | 2.000 | 50.000 | AVC | Main | 3 | - | 1800.00 | 2:00:00 | AAC-LC, 2-CH |
| ↳ | 2 | 1427 | 10.7% | 1131 | 906 | -19.9% | 640 x 360 | 2.000 | 50.000 | AVC | Main | 3.1 | - | 1800.00 | 2:00:00 | AAC-LC, 2-CH |
| ↳ | 3 | 2593 | 6.4% | 2121 | 1620 | -23.6% | 960 x 540 | 2.000 | 50.000 | AVC | Main | 3.1 | - | 1800.00 | 2:00:00 | AAC-LC, 2-CH |
| ↳ | 4 | 4485 | -2.7% | 3991 | 2608 | -34.7% | 1280 x 720 | 2.000 | 50.000 | AVC | High | 3.2 | - | 1800.00 | 2:00:00 | AAC-LC, 2-CH |

Average duration processed: 2:01:25

**Checked against *HLS Authoring Specification for Apple Devices***

**General requirements**

**Must Fix Issues**

**1. I-frame playlists ( EXT-X-I-FRAME-STREAM-INF ) MUST be provided to support scrubbing and scanning UI**

- Master Playlist

**2. The server MUST deliver playlists using gzip content-encoding**

- All Variants
- Master Playlist

**Should Fix Issues**

**3. Captions SHOULD be provided with your streams to make content accessible to the deaf or hard of hearing.**

- Master Playlist

**4. For H.264 you SHOULD use High Profile in preference to Main or Baseline Profile**

- Variant #1
- Variant #2
- Variant #3

**5. The default video variant(s) SHOULD be the 2000 kb/s variant.**

# 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