

WIDEVINE DRM CDM EME PLAYREADY CENC DASH
AIRPLAY WIDEVINE DRM CDM EME PLAYREADY C
ASH FAIRPLAY WIDEVINE DRM CDM EME PLAYRE
CENC DASH FAIRPLAY WIDEVINE DRM CDM EME DA
LAYREADY CENC DASH FAIRPLAY WIDEVINE DRM
ME PLAYREADY CENC DASH FAIRPLAY WIDEVINE
DM EME PLAYREADY CENC DASH FAIRPLAY WIDE
RM CDM EME PLAYREADY CENC DASH FAIRPLAY
WIDEVINE DRM CDM EME PLAYREADY CENC DASH

<ME/>

```
> cat me/package.json
{
  "private": true,
  "author": "Sebastian Golasch",
  "name": "@asciidisco",
  "description": "Specialist Senior Manager Software Developer",
  "homepage": "asciidisco.com",
  "repository": {
    "type": "job",
    "url": "deutsche-telekom.de"
  },
  "engines": {
    "js": "~13",
    "html": "~13",
    "css": "~13",
    "python": "~2",
    "iot": "~6"
  },
  "devDependencies": {
    "coffee": "^2.5.3"
  }
}
```

WIDEVINE DRM CDM EME PLAYREADY CENC DASH
AIRPLAY WIDEVINE DRM CDM EME PLAYREADY C
ASH FAIRPLAY WIDEVINE DRM CDM EME PLAYRE
CENC DASH FAIRPLAY WIDEVINE DRM CDM EME DA
LAYREADY CENC DASH FAIRPLAY WIDEVINE DRM
ME PLAYREADY CENC DASH FAIRPLAY WIDEVINE
DM EME PLAYREADY CENC DASH FAIRPLAY WIDE
RM CDM EME PLAYREADY CENC DASH FAIRPLAY
WIDEVINE DRM CDM EME PLAYREADY CENC DASH

WHAT?

*Never Heard
Of Those?*

*No Worries,
You've
Probably
Used Them A
Gazillion
Times!*





History

1990

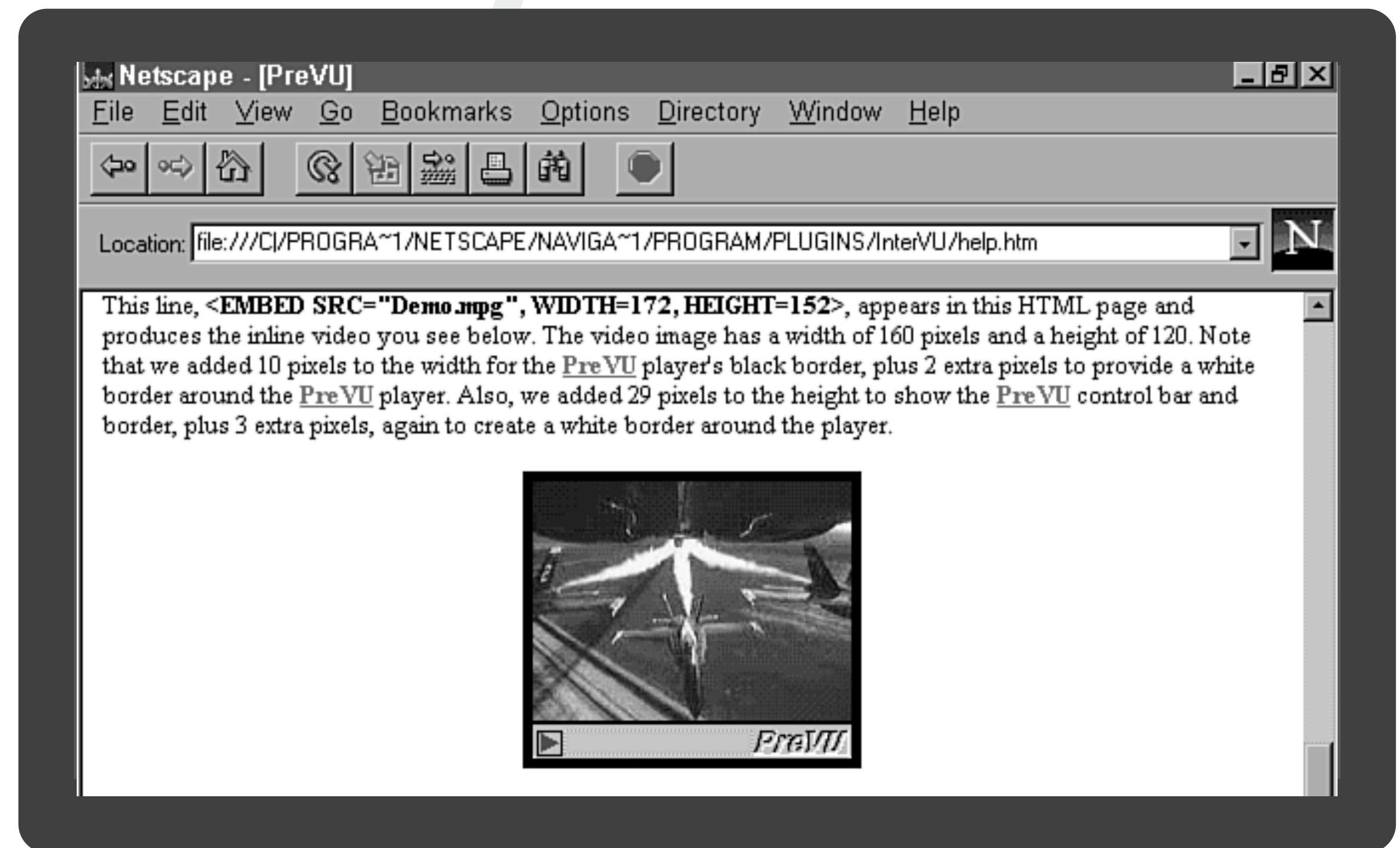
**156 × 116 PX
10 Fps**

**Without HW
Acceleration**

Min. 9600 Baud



Good Old Days



2002

Sorenson Spark
Aka. H.263

Optimised For:

Low Res
Small Filesize

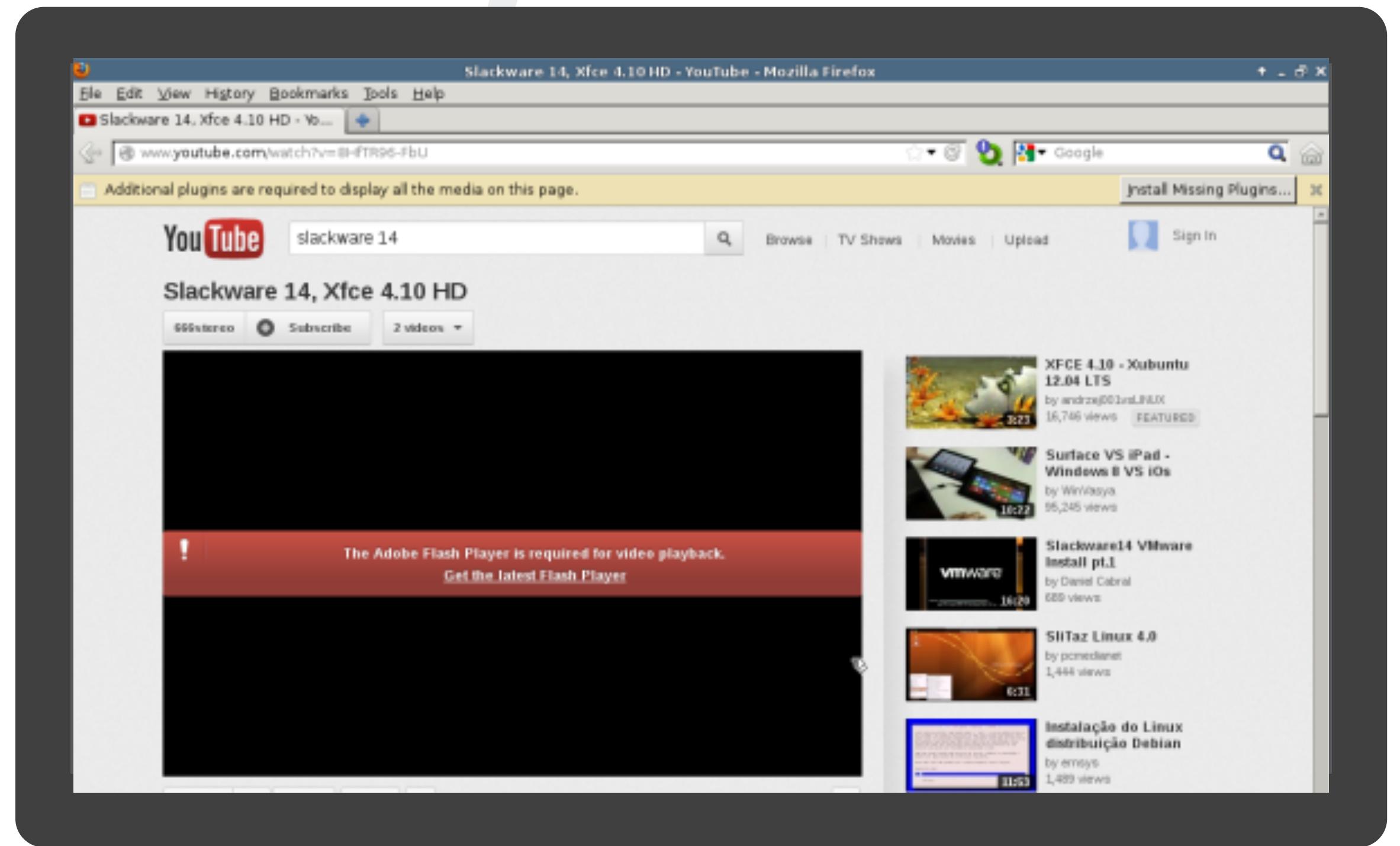


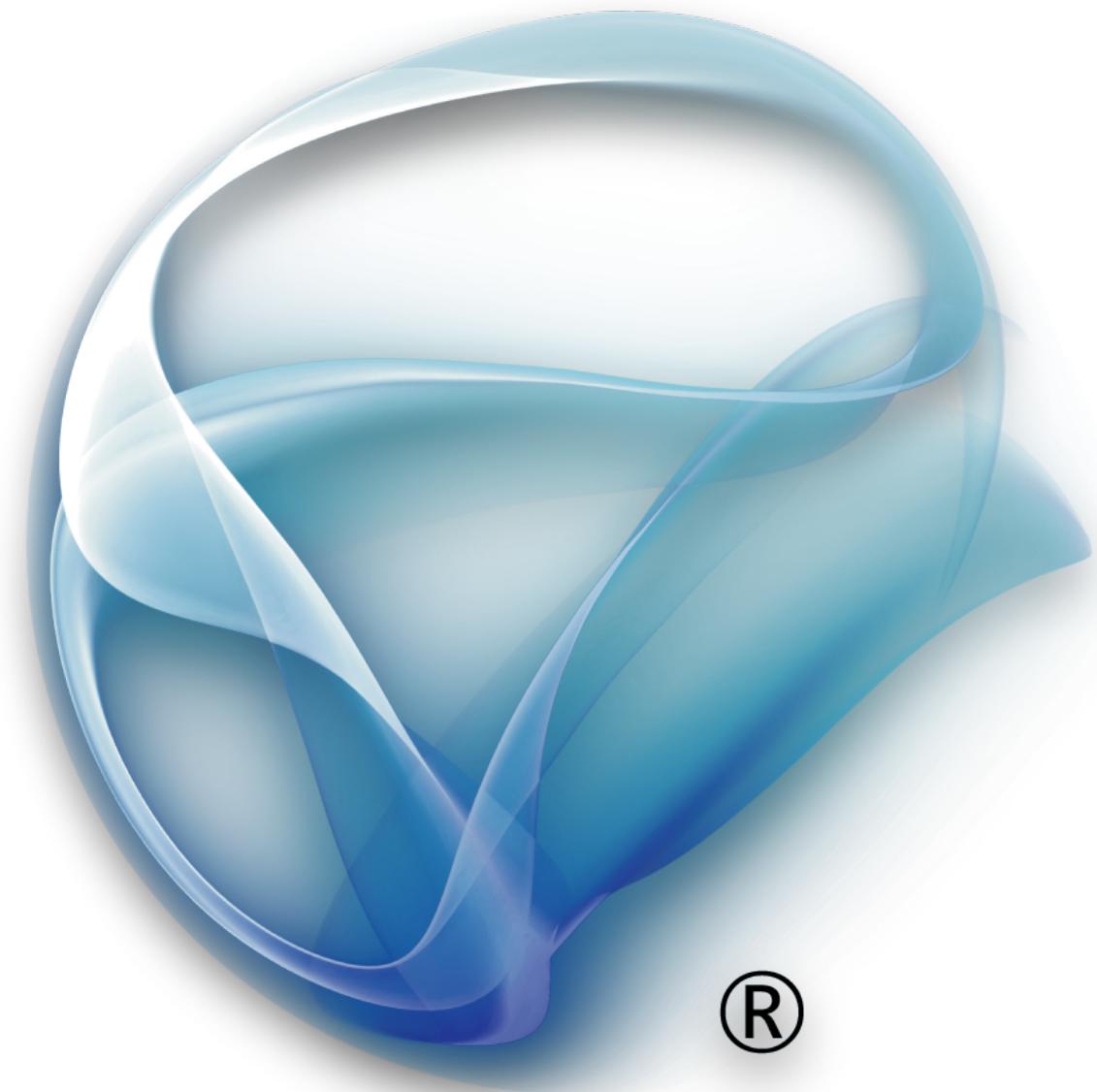
H.263

**560 Kb
43 Sec Dur.**



Good Old Days (Part 2)





Microsoft®
Silverlight®

<EMBED
BLACKBOX=TRUE/>

2007

Opera
Proposed
The
`<video>`
Tag





Present

<VIDEO/>



```
<video src="videofile.webm" autoplay poster="posterimage.jpg">  
Sorry, your browser doesn't support embedded videos,  
but don't worry, you can <a href="videofile.webm">download it</a>  
and watch it with your favorite video player!  
</video>
```

Prevent HTML5 video from being downloaded (right-click saved)?



How can I disable "Save Video As..." from a browser's right-click menu to prevent clients from downloading a video?

119

Are there more complete solutions that prevent the client from accessing a file path directly?



javascript html5 menu html5-video right-click



share edit flag

91

edited Jan 7 '17 at 3:35



Mureinik

149k • 21 • 102 • 163

asked Mar 18 '12 at 7:53



python

808 • 2 • 9 • 13

protected by Tushar Gupta Oct 4 '14 at 6:54

This question is protected to prevent "thanks!", "me too!", or spam answers by new users. To answer it, you must have earned at least 10 reputation on this site (the association bonus does not count).

- 3 I up-voted this question because it only *absolutely* asks for how to "disable the right-click" for an HTML5 video. I am not sure if it is similar to right-click disabling for normal images or if there are other overlay tricks,

15 Answers

active

oldest

votes



In reality, you can't. But you *can make it harder to download*.

...

WHOOPS...





<VIDEO
BLACKBOX=FALSE/>

*So Can I
Just...*



... No!



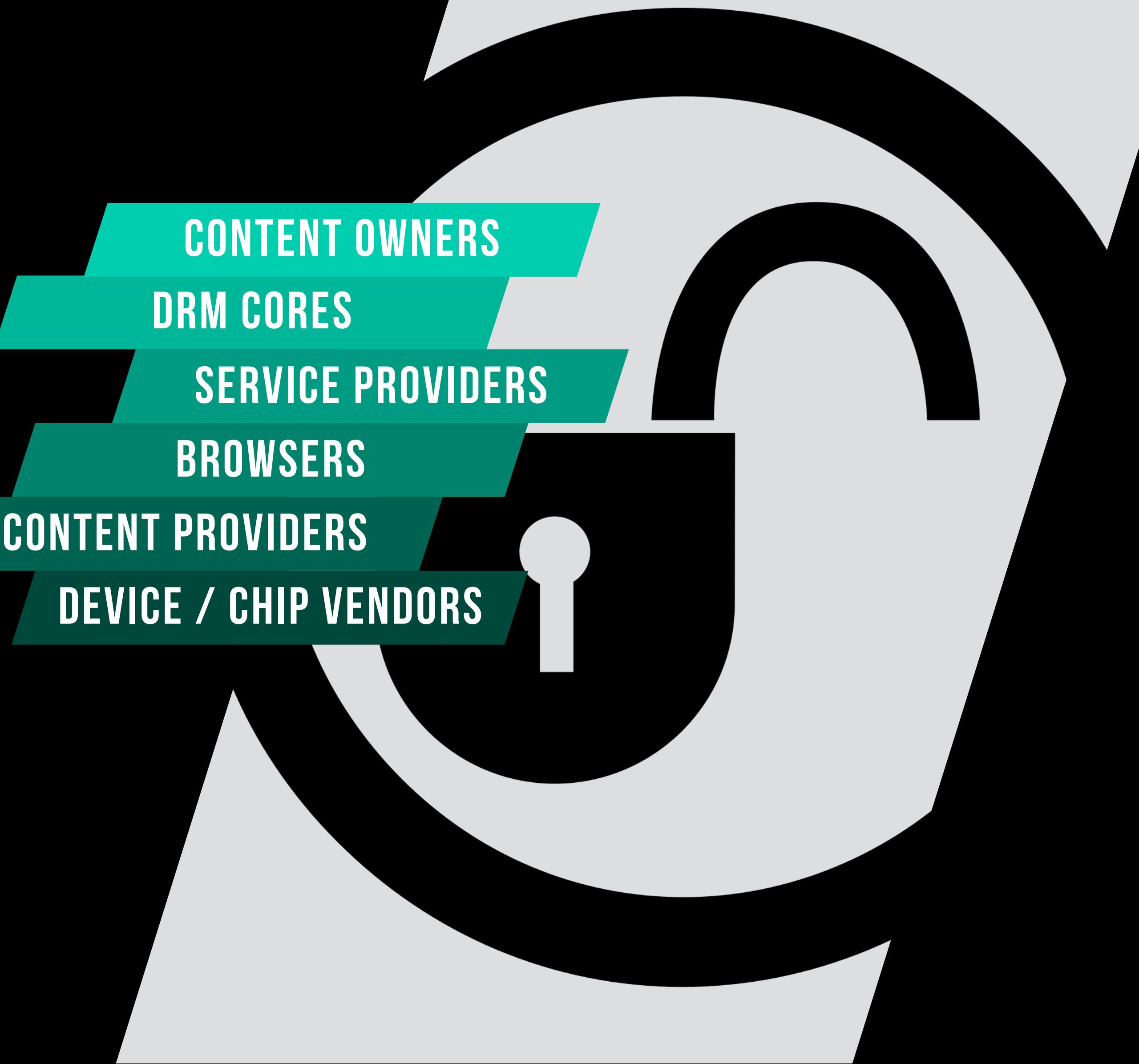


Digital Rights (Restrictions) Management

DRM

- AUTHENTICATION & USER SPECIFIC ENCRYPTION
- CONTENT SPECIFIC ENCRYPTION
- RIGHTS DEFINITION & RESTRICTION ENFORCEMENT
- REVOCATION & RENEWAL
- OUTPUT CONTROL & LINK PROTECTION
- FORENSICS & TRAITOR TRACING
- KEY & LICENSE MANAGEMENT

Ecosystem Stakeholders



CONTENT OWNERS
DRM CORES
SERVICE PROVIDERS
BROWSERS
CONTENT PROVIDERS
DEVICE / CHIP VENDORS

Version: 5.0008.544.011
Esn: NFCDCH-MC-CT75XQ4ETW768HFFJ3F1QE69CHYG8F
PBCID: 6.Hg9vLPeLc6kjGILRAYqWKX-YA4WS6EcNKbyWWhU40_s
UserAgent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36

Movield: 80222897
TrackingId: 14170287
Xid: 15236094717647 (15236094717647)
Position: 1539.446
Duration: 5541.000
Volume: 100%

Player state: Normal
Buffering state: Normal
Rendering state: Playing

Playing bitrate (a/v): 96 / 1470 (853x480)
Playing/Buffering vmaf: 84/84
Buffering bitrate (a/v): 96 / 1470
Buffer size in Bytes (a/v): 2966488 / 38281259
Buffer size in Bytes: 41247747
Buffer size in Seconds (a/v): 236.596 / 226.318
Current CDN (a/v): c070.fra001.ix.netflixvideo.net, Id: 48427 / c070.fra001.ix.netflixvideo.net, Id: 48427

Audio Track: de, Id: A:1:1;2;de;1;, Channels: 2.0, Codec: audio/mp4;codecs=mp4a.40.2
Video Track: Codec: video/mp4;codecs=avc1.640028 (avc)
Timed Text Track: none

Framerate: NaN
Current Dropped Frames: 0

ying bitrate (a/v): 96 / 1470 (853x480)

ying/Buffering vmaf: 84/84

fering bitrate (a/v): 96 / 1470

fer size in Bytes (a/v): 3164118 / 4121

fer size in Bytes: 44383374

fer size in Seconds (a/v): 246.593 / 22

rent CDN (a/v): c070.fra001.ix.netflix.net

CONTENT QUALITY DEPENDING ON TRUSTED ENVIRONMENT ROBUSTNESS

SOFTWARE
CLIENT

HARDWARE
ASSISTED

FULL
HARDWARE

4K

HD+

1080P

720P

SD



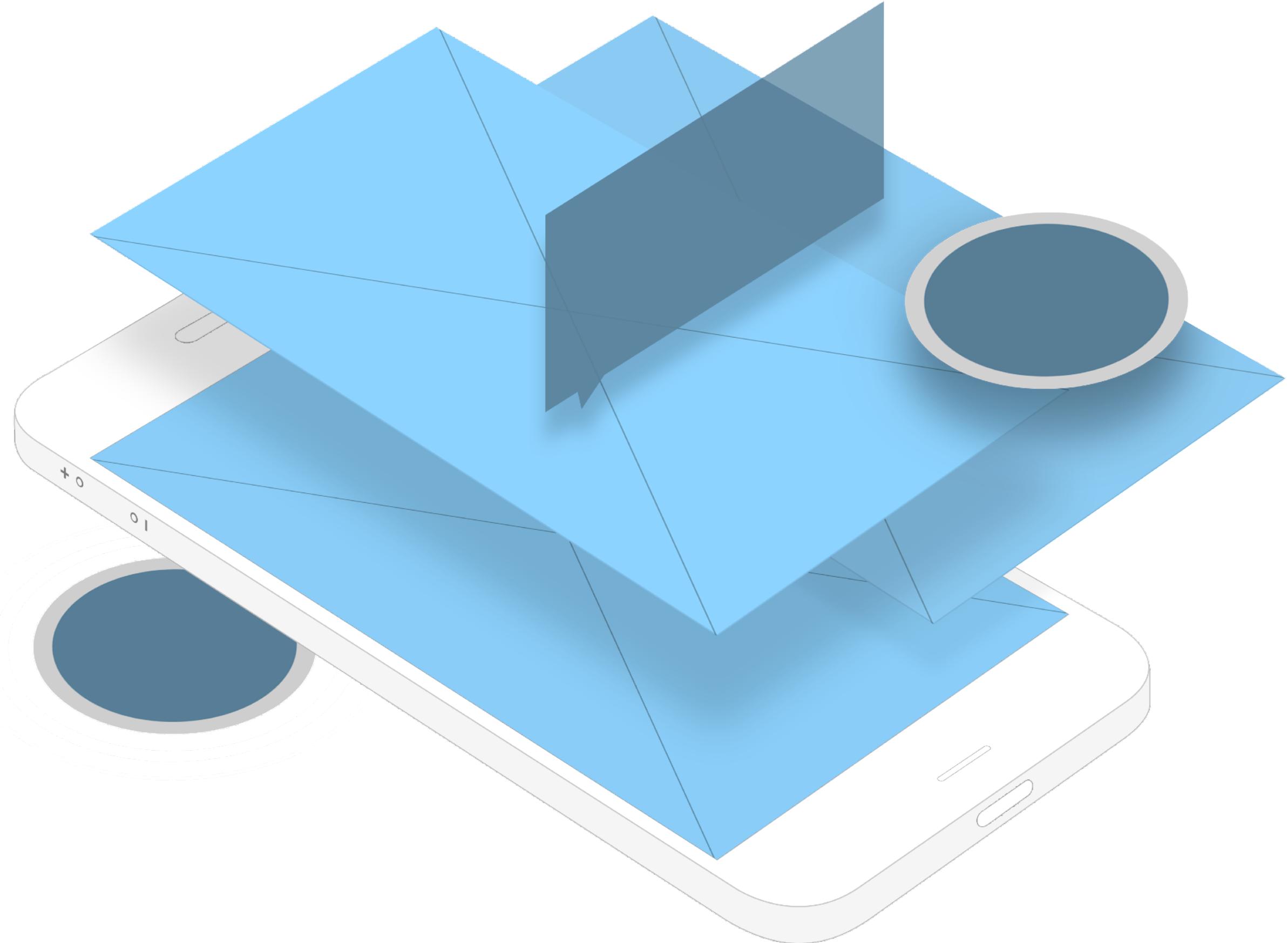


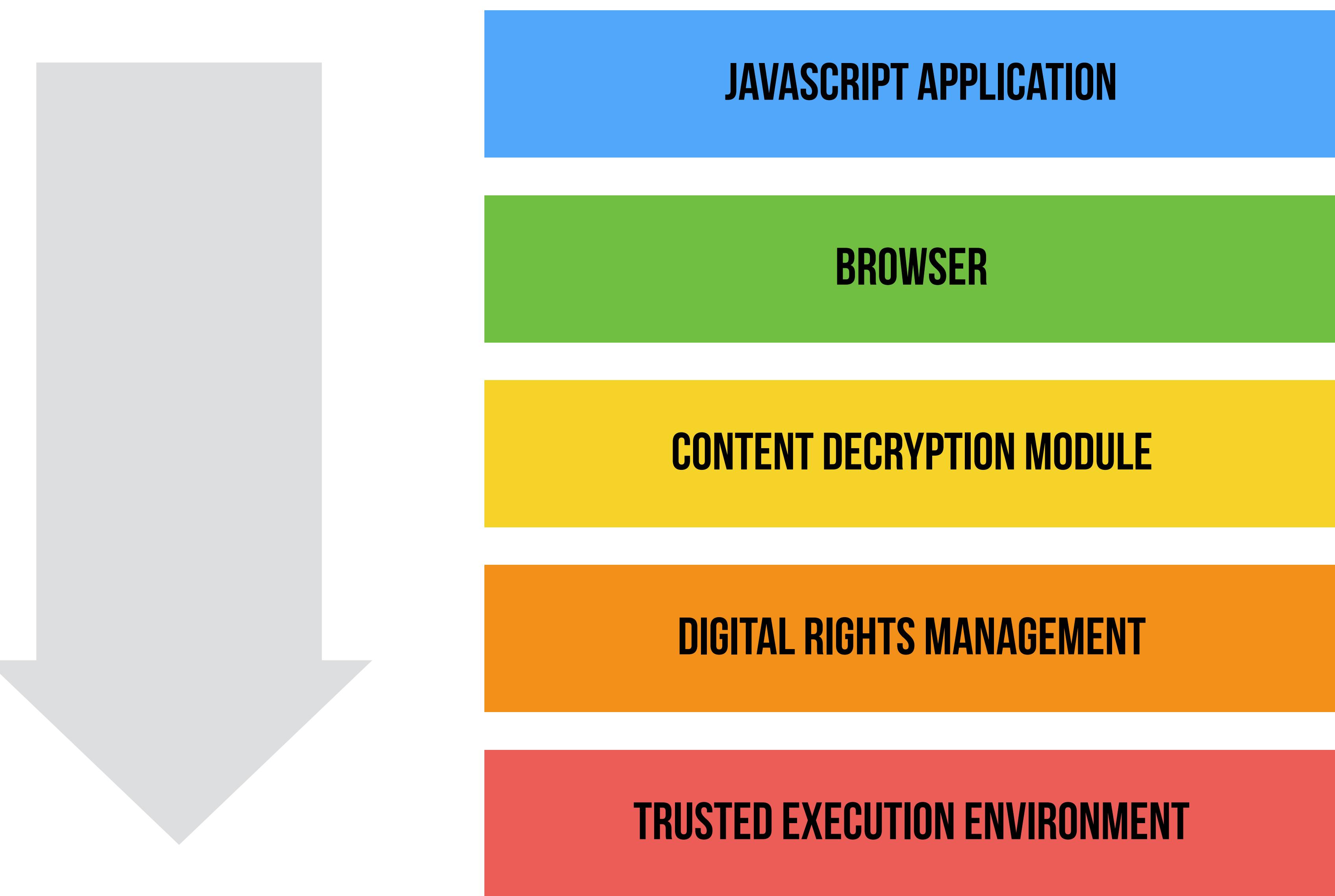
```
{  
  "vendors": {  
    "gmp-widevinecdm": {  
      "platforms": {  
        "WINNT_x86-msvc-x64": {  
          "alias": "WINNT_x86-msvc"  
        },  
        "WINNT_x86-msvc": {  
          "fileUrl": "https://redirector.gvt1.com/edgedl/widevine-cdm/970-win-ia32.zip",  
          "hashValue": "...",  
          "filesize": 2342279  
        },  
        "WINNT_x86-msvc-x86": {  
          "alias": "WINNT_x86-msvc"  
        },  
        "Linux_x86_64-gcc3": {  
          "fileUrl": "https://redirector.gvt1.com/edgedl/widevine-cdm/970-linux-x64.zip",  
          "hashValue": "...",  
          "filesize": 2210161  
        },  
        "Darwin_x86_64-gcc3-u-i386-x86_64": {  
          "fileUrl": "https://redirector.gvt1.com/edgedl/widevine-cdm/970-mac-x64.zip",  
          "hashValue": "...",  
          "filesize": 1681446  
        }  
      }  
    }  
  }  
}
```

CDM

- **CONTENT DECRYPTION MODULE**
- **“SOME PIECE OF SOFT- OR HARDWARE”**
- **DECRYPTION ONLY, LET THE <VIDEO/> RENDER**
- **DECRYPT & DECODE, PASS VIDEO FRAMES -> BROWSER**
- **DECRYPT, DECODE & RENDER ON HARDWARE (GPU)**

Browser Decryption And Decoding Layers





JAVASCRIPT APPLICATION

BROWSER

CONTENT DECRYPTION MODULE

DIGITAL RIGHTS MANAGEMENT

TRUSTED EXECUTION ENVIRONMENT

JAVASCRIPT APPLICATION

BROWSER

CONTENT DECRYPTION MODULE

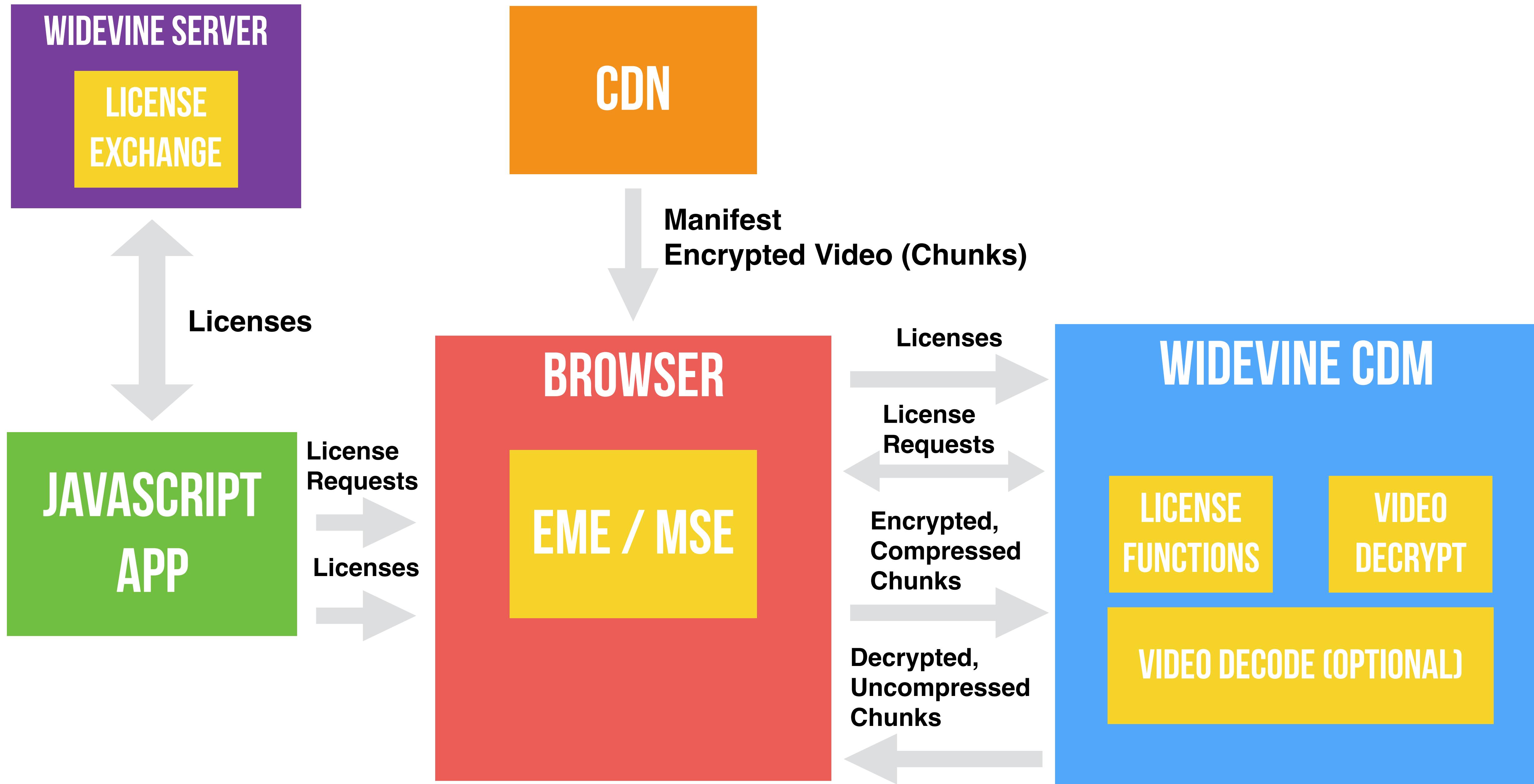
DIGITAL RIGHTS MANAGEMENT

TRUSTED EXECUTION ENVIRONMENT

PLAYER

CLIENT

CORE





Real
World
(Netflix)



Demo

<input type="checkbox"/> pathEvaluator?drmSystem=widevine&isWatchlistEnable...ranchingP...	200	xhr	none:2	2.3 KB	63...
<input type="checkbox"/> pathEvaluator?method=call&drmSystem=widevine&isWat...ranchin...	200	xhr	none:2	1.6 KB	62...
<input type="checkbox"/> metadata?drmSystem=widevine&isWatchlistEnabled=fal...bp&with...	200	xhr	none:2	2.7 KB	55...
<input type="checkbox"/> probe?monotonic=false&iter=0	200	xhr	cadmium-...	1.5 KB	29...
<input type="checkbox"/> debug	(failed)	xhr	cadmium-...	0 B	28...
<input type="checkbox"/> debug	(failed)	xhr	cadmium-...	0 B	14...
<input type="checkbox"/> stop	200	xhr	cadmium-...	7.7 KB	85...
<input type="checkbox"/> release	200	xhr	cadmium-...	7.7 KB	81...
<input type="checkbox"/> manifest	200	xhr	cadmium-...	107 KB	34...
<input type="checkbox"/> 0-11550?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZjm...Te3NX...	200	xhr	cadmium-...	11.8 KB	52...
<input type="checkbox"/> 0-11550?o=AQHxPHPM_TMR7DzcAwFxbwtRkxcB62cZjn...Te3NX.....	200	xhr	cadmium-...	11.8 KB	44...
<input type="checkbox"/> 330559-449351?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZjm...	200	xhr	cadmium-...	117 KB	10...
<input type="checkbox"/> 34236-230191?o=AQHxPHPM_TMR7DzcAwFxbwtRkxcB62cZjn.....	200	xhr	cadmium-...	192 KB	96...
<input type="checkbox"/> 449352-1897496?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZj...	200	xhr	cadmium-...	1.4 MB	27...
<input type="checkbox"/> 1897497-2970949?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZ...	200	xhr	cadmium-...	1.0 MB	24...
<input type="checkbox"/> 230192-425229?o=AQHxPHPM_TMR7DzcAwFxbwtRkxcB62cZjn.....	200	xhr	cadmium-...	191 KB	10...
<input type="checkbox"/> 2970950-3937754?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZ...	200	xhr	cadmium-...	945 KB	31...
<input type="checkbox"/> 3937755-5103453?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZ...	200	xhr	cadmium-...	1.1 MB	44...
<input type="checkbox"/> cl2	202	xhr	none:2	1.1 KB	99...
<input type="checkbox"/> license	200	xhr	cadmium-...	12.0 KB	20...
<input type="checkbox"/> 5103454-6097163?o=AQHxPHPM_TMR7DzcAwFxbwdTkxF62cZ...	200	xhr	cadmium-...	971 KB	30...
<input type="checkbox"/> 0-11550?o=AQHxPHPM_TMR7DzcAwFxbwdQIRID62cZjm...Te3NX...	200	xhr	cadmium-...	11.8 KB	82...
<input type="checkbox"/> 0-11550?o=AQHxPHPM_TMR7DzcAwFxbwdWIRoH62cZjm...Te3NX...	200	xhr	cadmium-...	11.8 KB	58...
<input type="checkbox"/> bind	200	xhr	cadmium-...	11.0 KB	11...
<input type="checkbox"/> 8568320-9750720?o=AQHxPHPM_TMR7DzcAwFxbwdWIRoH62cZ...	200	xhr	cadmium-...	1.1 MB	44...
<input type="checkbox"/> 425230-620638?o=AQHxPHPM_TMR7DzcAwFxbwtRkxcB62cZjn.....	200	xhr	cadmium-...	191 KB	14...



<input type="checkbox"/> manifest	200	xhr	cadmium-...	107 KB	34...
-----------------------------------	-----	-----	-----------------------------	--------	-------

<input type="checkbox"/> license	200	xhr	cadmium-...	12.0 KB	20...
----------------------------------	-----	-----	-----------------------------	---------	-------

<input type="checkbox"/> 0-11550?o=AQHxPHPM_TMR7DzcAwFxbwdWIRoH62cZjmvTe3NX...	200	xhr	cadmium-...	11.8 KB	58...
--	-----	-----	-----------------------------	---------	-------

BOILERPLATE



```
const $video = document.querySelector('video')
const config = getKeySystemConfig()
const mediaKeys = await initMediaKeySystem(config, $video)
const request = await createMediaKeySession(mediaKeys)
```

EME

- “ENCRYPTED MEDIA EXTENSIONS”
- BROWSER API
- INTERFACE TO THE CDM
- INTERFACE TO THE KEY SYSTEM
- INTERFACE TO THE LICENSE (KEY) SERVER
- INTERFACE TO THE PACKAGING SERVICE
- DON’T DO ANY DECODING/DECRYPTING THEMSELVES

BOILERPLATE



```
const $video = document.querySelector('video')
const config = getKeySystemConfig()
const mediaKeys = await initMediaKeySystem(config, $video)
const request = await createMediaKeySession(mediaKeys)
```

KEY SYSTEM CONFIG

```
function getKeySystemConfig() {
  return [
    distinctiveIdentifier: 'not-allowed',
    videoCapabilities: [
      {
        contentType: 'video/mp4; codecs="avc1.640028"' ,
        robustness: 'HW_SECURE_DECODE'
      }, {
        contentType: 'video/mp4; codecs="avc1.640028"' ,
        robustness: 'SW_SECURE_DECODE'
      }],
    audioCapabilities: [
      {
        contentType: 'audio/mp4; codecs="mp4a.40.5"' ,
        robustness: 'SW_SECURE_CRYPTO'
      }]
  }
}
```

KEY SYSTEM CONFIG

```
function getKeySystemConfig() {
  return [
    distinctiveIdentifier: 'not-allowed',
    videoCapabilities: [
      {
        contentType: 'video/mp4; codecs="avc1.640028"' ,
        robustness: 'HW_SECURE_DECODE'
      }, {
        contentType: 'video/mp4; codecs="avc1.640028"' ,
        robustness: 'SW_SECURE_DECODE'
      }],
    audioCapabilities: [
      {
        contentType: 'audio/mp4; codecs="mp4a.40.5"' ,
        robustness: 'SW_SECURE_CRYPTO'
      }]
  ]
}
```

BOILERPLATE



```
const $video = document.querySelector('video')
const config = getKeySystemConfig()
const mediaKeys = await initMediaKeySystem(config, $video)
const request = await createMediaKeySession(mediaKeys)
```

KEY SYSTEM



```
async function initMediaKeySystem (config, $video) {
  let keySystem = await navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
  let mediaKeys = await keySystem.createMediaKeys()
  mediaKeys.setServerCertificate(TextDecoder().decode(SERVER_CERT).buffer)
  $video.setMediaKeys(mediaKeys)
  return mediaKeys
}
```

KEY SYSTEM



```
async function initMediaKeySystem (config, $video) {
  let keySystem = await navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
  let mediaKeys = await keySystem.createMediaKeys()
  mediaKeys.setServerCertificate(TextDecoder().decode(SERVER_CERT).buffer)
  $video.setMediaKeys(mediaKeys)
  return mediaKeys
}
```

KEY SYSTEM



```
async function initMediaKeySystem (config, $video) {
  let keySystem = await navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
  let mediaKeys = await keySystem.createMediaKeys()
  mediaKeys.setServerCertificate(TextDecoder().decode(SERVER_CERT).buffer)
  $video.setMediaKeys(mediaKeys)
  return mediaKeys
}
```

KEY SYSTEM



```
async function initMediaKeySystem (config, $video) {
  let keySystem = await navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
  let mediaKeys = await keySystem.createMediaKeys()
  mediaKeys.setServerCertificate(TextDecoder().decode(SERVER_CERT).buffer)
  $video.setMediaKeys(mediaKeys)
  return mediaKeys
}
```

BOILERPLATE



```
const $video = document.querySelector('video')
const config = getKeySystemConfig()
const mediaKeys = await initMediaKeySystem(config, $video)
const request = await createMediaKeySession(mediaKeys)
```

KEY SESSION



```
async function createMediaKeySession (mediaKeys) {
  // 'AAAANHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAATKB0kAAAAAAAAAAA='
  let initData = new Uint8Array([/**/])
  let keySession = mediaKeys.createSession(
    'video/mp4',
    new Uint8Array(0),
    new Uint8Array(`<ServerCert>Q0hBSQAAAAEAAAUMAAAAAAAJDRVJUAAAAAQ...`))
  keySession.addEventListener('message', handleMessage, false)
  let request = await keySession.generateRequest('cenc', initData)
  return request
}
```

KEY SESSION



```
async function createMediaKeySession (mediaKeys) {
  // 'AAAANHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAATKB0kAAAAAAAAAAA=='
  let initData = new Uint8Array([/**/])
  let keySession = mediaKeys.createSession(
    'video/mp4',
    new Uint8Array(0),
    new Uint8Array(`<ServerCert>Q0hBSQAAAAEAAAUMAAAAAAAJDRVJUAAAAAQ...`))
  keySession.addEventListener('message', handleMessage, false)
  let request = await keySession.generateRequest('cenc', initData)
  return request
}
```

KEY SESSION



```
async function createMediaKeySession (mediaKeys) {
  // 'AAAANHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAATKB0kAAAAAAAAAAA='
  let initData = new Uint8Array([/**/])
  let keySession = mediaKeys.createSession(
    'video/mp4',
    new Uint8Array(0),
    new Uint8Array(`<ServerCert>Q0hBSQAAAAEAAAUMAAAAAAAJDRVJUAAAAAQ...`))
  keySession.addEventListener('message', handleMessage, false)
  let request = await keySession.generateRequest('cenc', initData)
  return request
}
```

KEY SESSION

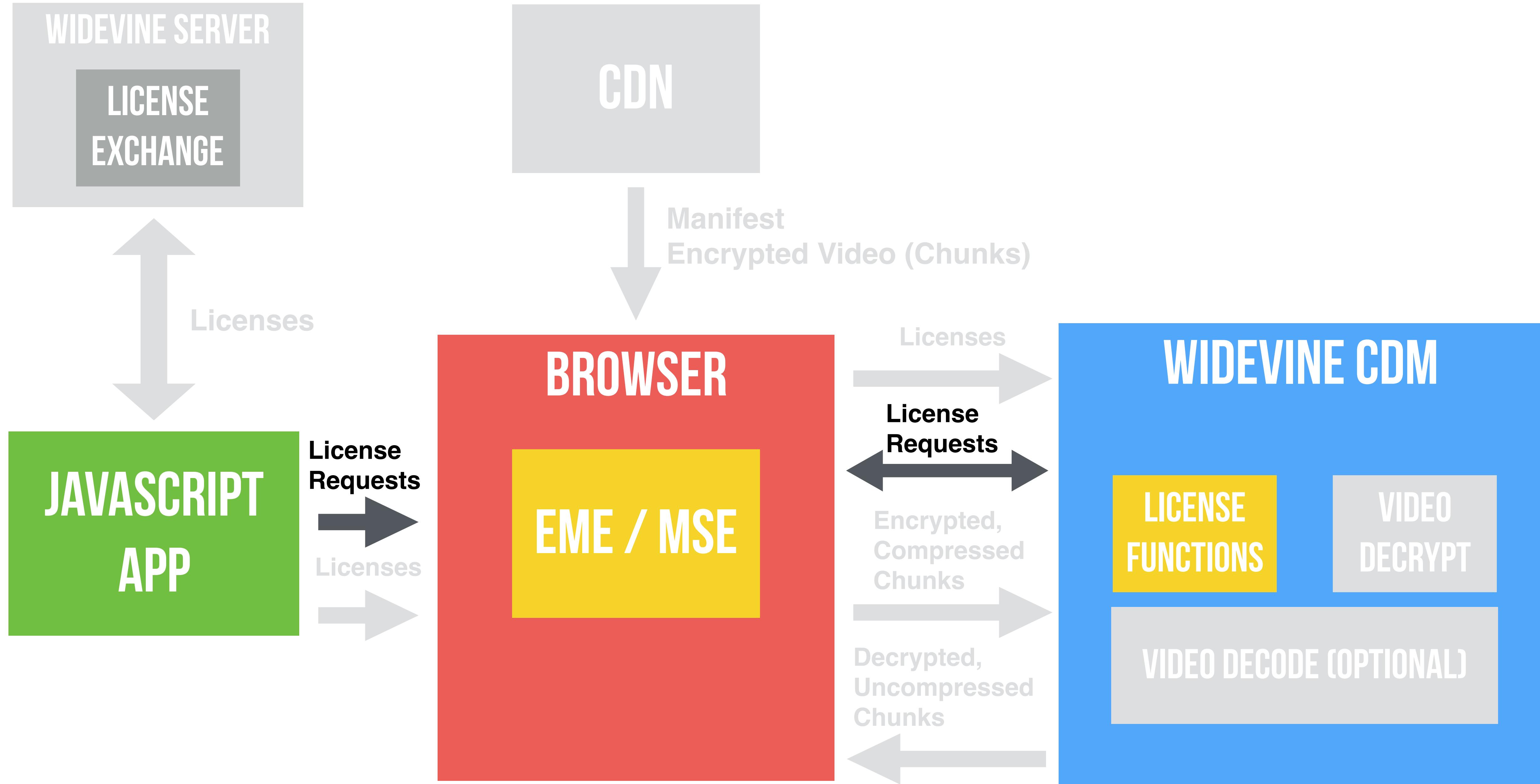


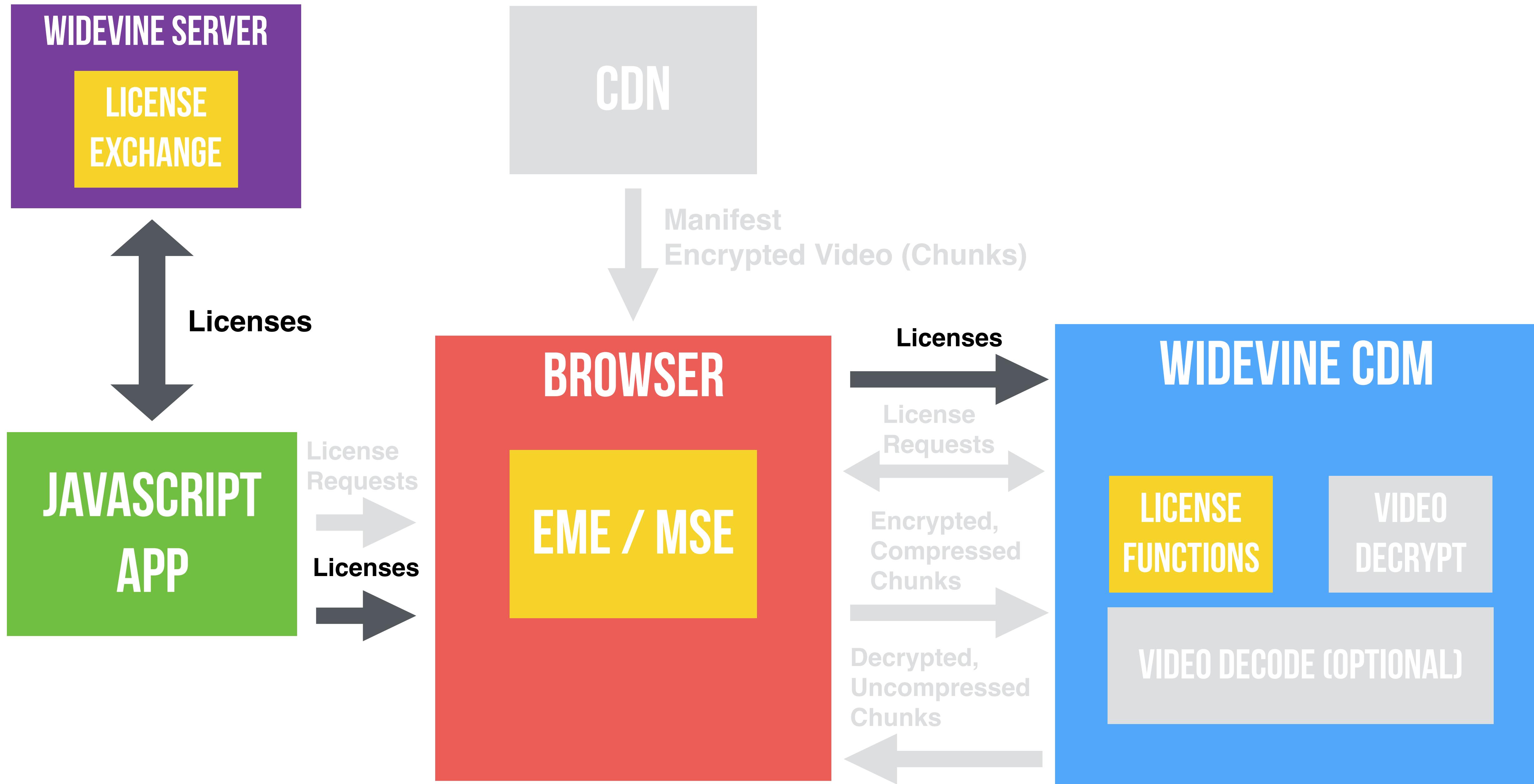
```
async function createMediaKeySession (mediaKeys) {
  // 'AAAANHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAATKB0kAAAAAAAAAA=='
  let initData = new Uint8Array([/**/])
  let keySession = mediaKeys.createSession(
    'video/mp4',
    new Uint8Array(0),
    new Uint8Array(`<ServerCert>Q0hBSQAAAAEAAAUMAAAAAAAJDRVJUAAAAAQ...`))
  keySession.addEventListener('message', handleMessage, false)
  let request = await keySession.generateRequest('cenc', initData)
  return request
}
```

GET LICENSE



```
async function handleMessage(event) {
  const response = await fetch(LICENSE_SERVER, {method: 'POST', body: event.message})
  const license = await response.arrayBuffer()
  const session = event.target
  session.update(license)
}
```





MSE

- **"MEDIA SOURCE EXTENSIONS"**
- **BROWSER API**
- **PROGRAMMATIC INTERFACE TO THE <VIDEO/> SRC**
- **ALLOWS TO CREATE STREAMS FROM JS**
- **APPLY CHUNKS OF MEDIA TO THE <VIDEO/>**
- **TOTALLY UNRELATED TO DRM ITSELF**

MEDIA SOURCE EXTENSIONS

```
let sourceBuffer = null
const mediaSource = new MediaSource()
mediaSource.addEventListener('sourceopen', () => {
  sourceBuffer = mediaSource.addSourceBuffer('video/mp4; codecs="avc1.640028,mp4a.40.5"')
}, false)

const $video = document.querySelector('video')
$video.src = window.URL.createObjectURL(mediaSource)

async function loadChunk (range, sourceBuffer) {
  const response = await fetch(`path/to/chunk/${range}`)
  const rawChunk = await response.arrayBuffer()
  const chunk = new Uint8Array(rawChunk)
  sourceBuffer.append(chunk)
}

['0-150', '150-300', '300-450'].map(loadChunk)
```

MEDIA SOURCE EXTENSIONS

```
let sourceBuffer = null
const mediaSource = new MediaSource()
mediaSource.addEventListener('sourceopen', () => {
  sourceBuffer = mediaSource.addSourceBuffer('video/mp4; codecs="avc1.640028,mp4a.40.5"')
}, false)

const $video = document.querySelector('video')
$video.src = window.URL.createObjectURL(mediaSource)

async function loadChunk (range, sourceBuffer) {
  const response = await fetch(`/path/to/chunk/${range}`)
  const rawChunk = await response.arrayBuffer()
  const chunk = new Uint8Array(rawChunk)
  sourceBuffer.append(chunk)
}

['0-150', '150-300', '300-450'].map(loadChunk)
```

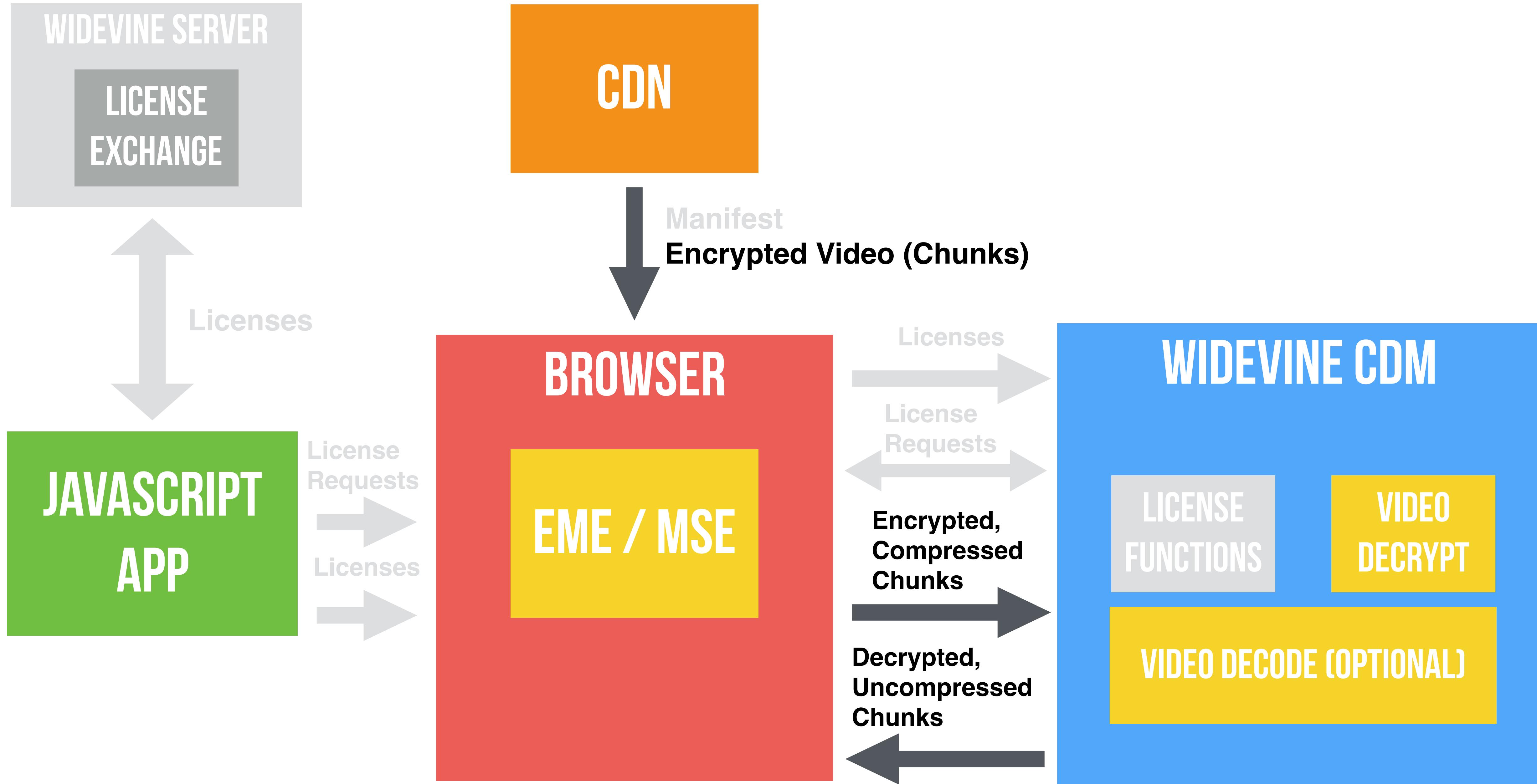
MEDIA SOURCE EXTENSIONS

```
let sourceBuffer = null
const mediaSource = new MediaSource()
mediaSource.addEventListener('sourceopen', () => {
  sourceBuffer = mediaSource.addSourceBuffer('video/mp4; codecs="avc1.640028,mp4a.40.5"')
}, false)

const $video = document.querySelector('video')
$video.src = window.URL.createObjectURL(mediaSource)

async function loadChunk (range, sourceBuffer) {
  const response = await fetch(`/path/to/chunk/${range}`)
  const rawChunk = await response.arrayBuffer()
  const chunk = new Uint8Array(rawChunk)
  sourceBuffer.append(chunk)
}

['0-150', '150-300', '300-450'].map(loadChunk)
```





Manifest

MANIFEST



```
function getManifest() {
    return {
        sessionId: "2362746237491632",
        viewableIds: [70301645],
        drmSystem: "widevine",
        profiles: [
            "playready-h264bpl30-dash",
            "playready-h264mpl30-dash",
            "playready-h264mpl31-dash",
            "playready-h264mpl40-dash",
            "heaac-2-dash",
            "simplesdh",
            "BIF240",
            "BIF320",
            "ddplus-2.0-dash",
            "ddplus-5.1-dash"
        ],
        uiVersion: "akira",
        secureUrls: false,
```

MANIFEST



```
function getManifest( ) {
    return {
        sessionId: "2362746237491632",
        viewableIds: [ 70301645 ],
        drmSystem: "widevine",
        profiles: [
            "playready-h264bpl30-dash",
            "playready-h264mpl30-dash",
            "playready-h264mpl31-dash",
            "playready-h264mpl40-dash",
            "heaac-2-dash",
            "simplesdh",
            "BIF240",
            "BIF320",
            "ddplus-2.0-dash",
            "ddplus-5.1-dash"
        ],
        uiVersion: "akira",
        secureUrls: false,
```

MANIFEST



```
function getManifest( ) {
    return {
        sessionId: "2362746237491632",
        viewableIds: [ 70301645 ],
        drmSystem: "widevine",
        profiles: [
            "playready-h264bpl30-dash",
            "playready-h264mpl30-dash",
            "playready-h264mpl31-dash",
            "playready-h264mpl40-dash",
            "heaac-2-dash",
            "simplesdh",
            "BIF240",
            "BIF320",
            "ddplus-2.0-dash",
            "ddplus-5.1-dash"
        ],
        uiVersion: "akira",
        secureUrls: false,
```

MPEG-DASH

- XML Based
- Defines Duration, Buffer Sizes, Content Type
- Defines Which Chunks To Load When
- Can Define Same Chunks For Different Resolutions
- Known As “Adaptive Bitrate Switching”
- Can Define Different Audio Tracks (Stereo, 5.1, I18N)

MANIFEST

● ● ●

```
<MPD mediaPresentationDuration="PT8152.00S" xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013">
  <Period duration="PT8152.00S" start="PT0S">
    <AdaptationSet contentType="video" mimeType="video/mp4">
      <ContentProtection schemeIdUri="urn:uuid:EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED">
        <widevine:license robustness_level="HW_SECURE_CODECS_REQUIRED" />
        <cenc:pssh>AAAANHBzc2gAAAAA7+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAAATCTgkAAAAAAAAAAA==</cenc:pssh>
      </ContentProtection>
      <Representation bandwidth="215040" codecs="h264" hdp="0.0" height="240" mimeType="video/mp4" width="426">...</Representation>
    </AdaptationSet>
    <AdaptationSet contentType="audio" impaired="false" lang="en" mimeType="audio/mp4">...</Representation>
  </AdaptationSet>
  <AdaptationSet codecs="stpp" contentType="text" lang="en" mimeType="application/ttml+xml">...</AdaptationSet>
  </Period>
</MPD>
```

MANIFEST

● ● ●

```
<MPD mediaPresentationDuration="PT8152.00S" xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013">
  <Period duration="PT8152.00S" start="PT0S">
    <AdaptationSet contentType="video" mimeType="video/mp4">
      <ContentProtection schemeIdUri="urn:uuid:EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED">
        <widevine:license robustness_level="HW_SECURE_CODECS_REQUIRED" />
        <cenc:pssh>AAAANHBzc2gAAAAA7+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAAATCTgkAAAAAAAAAAA==</cenc:pssh>
      </ContentProtection>
      <Representation bandwidth="215040" codecs="h264" hdp="0.0" height="240" mimeType="video/mp4" width="426">...</Representation>
    </AdaptationSet>
    <AdaptationSet contentType="audio" impaired="false" lang="en" mimeType="audio/mp4">...</Representation>
  </AdaptationSet>
  <AdaptationSet codecs="stpp" contentType="text" lang="en" mimeType="application/ttml+xml">...</AdaptationSet>
  </Period>
</MPD>
```

MANIFEST

● ● ●

```
<MPD mediaPresentationDuration="PT8152.00S" xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013">
  <Period duration="PT8152.00S" start="PT0S">
    <AdaptationSet contentType="video" mimeType="video/mp4">
      <ContentProtection schemeIdUri="urn:uuid:EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED">
        <widevine:license robustness_level="HW_SECURE_CODECS_REQUIRED" />
        <cenc:pssh>AAAANHBzc2gAAAAA7+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAAATCTgkAAAAAAAAAAA==</cenc:pssh>
      </ContentProtection>
      <Representation bandwidth="215040" codecs="h264" hdp="0.0" height="240" mimeType="video/mp4" width="426">...</Representation>
    </AdaptationSet>
    <AdaptationSet contentType="audio" impaired="false" lang="en" mimeType="audio/mp4">...</Representation>
  </AdaptationSet>
  <AdaptationSet codecs="stpp" contentType="text" lang="en" mimeType="application/ttml+xml">...</AdaptationSet>
  </Period>
</MPD>
```

MANIFEST

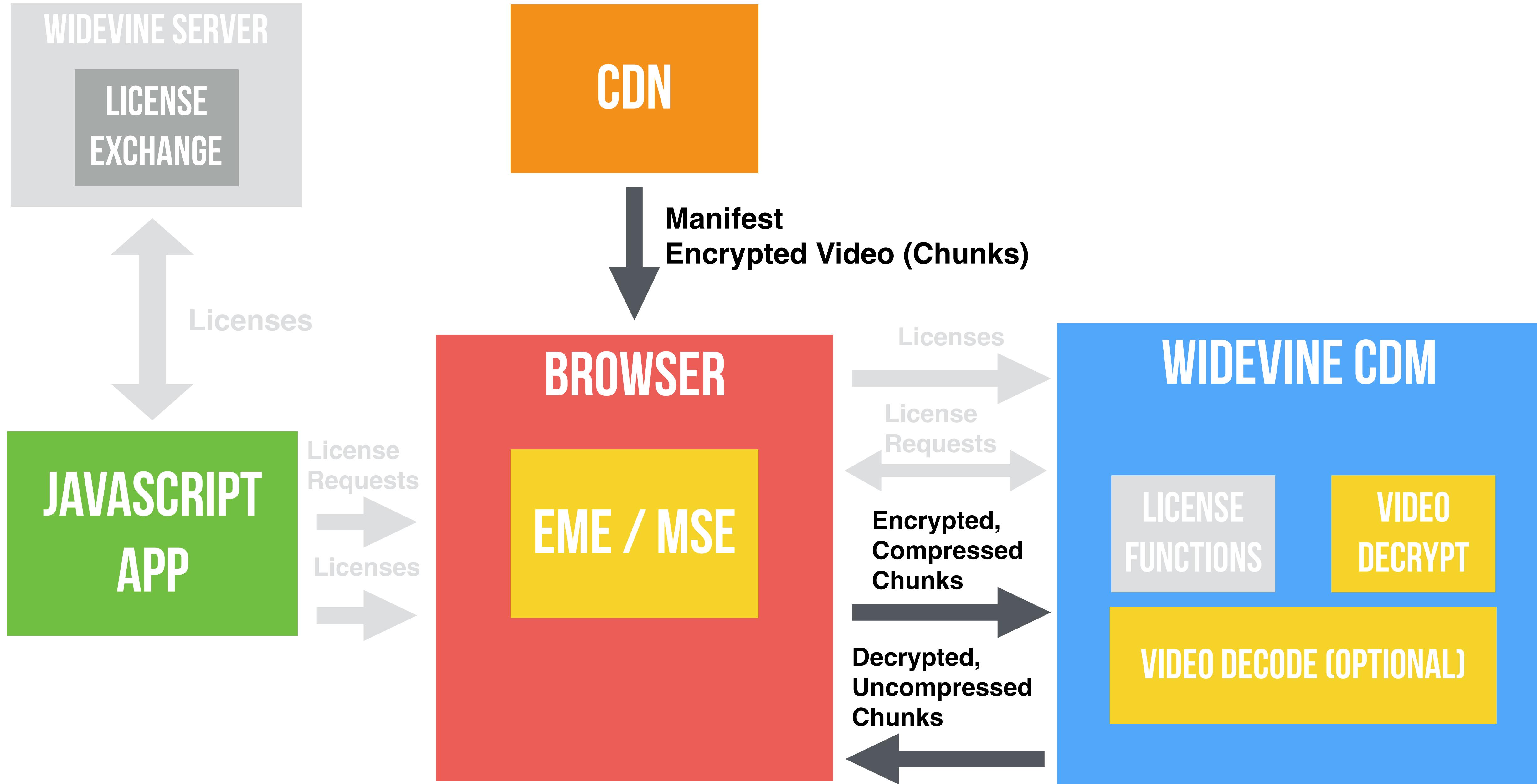
● ● ●

```
<MPD mediaPresentationDuration="PT8152.00S" xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013">
  <Period duration="PT8152.00S" start="PT0S">
    <AdaptationSet contentType="video" mimeType="video/mp4">
      <ContentProtection schemeIdUri="urn:uuid:EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED">
        <widevine:license robustness_level="HW_SECURE_CODECS_REQUIRED" />
        <cenc:pssh>AAAANHBzc2gAAAAA7+LqXnWSs6jyCfc1R0h7QAAABQIARIQAAAAAATCTgkAAAAAAAAAA==</cenc:pssh>
      </ContentProtection>
      <Representation bandwidth="215040" codecs="h264" hdp="0.0" height="240" mimeType="video/mp4" width="426">...</Representation>
    </AdaptationSet>
    <AdaptationSet contentType="audio" impaired="false" lang="en" mimeType="audio/mp4">...</Representation>
  </AdaptationSet>
  <AdaptationSet codecs="stpp" contentType="text" lang="en" mimeType="application/ttml+xml">...</AdaptationSet>
  </Period>
</MPD>
```

MANIFEST



```
<Representation bandwidth="215040" codecs="h264" hdcp="0.0" height="240" mimeType="video/mp4" nflxContentProfile="playready-h264mpl30-dash" width="426">
  <BaseURL>http://ipv6_1.cxl0.c113.sea001.ix.netflixvideo.net/?o=AQHg...</BaseURL>
  <SegmentBase indexRange="0-68912" indexRangeExact="true">
    <Initialization range="0-68912" />
  </SegmentBase>
</Representation>
```





**But Because
Netflix...**



Message Security Layer

MSL

- **Message Security Layer**
- **Is Netflix Specific, but an open sourced protocol**
- **Has nothing to do with Web Media directly**
- **As security layer to encrypt communication**
- **It is used by Netflix to encrypt License, Manifest etc.**



Message Security Layer: A Modern Take on Securing Communication

Netflix serves audio and video to millions of devices and subscribers across the globe. Each device has its own unique hardware and software, and differing security properties and capabilities. The communication between these devices and our servers must be secured to protect both our subscribers and our service.

When we first launched the Netflix streaming service we used a combination of HTTPS and a homegrown security mechanism called NTBA to provide that security. However, over time this combination started exhibiting growing pains. With the advent of HTML5 and the Media Source Extensions and Encrypted Media Extensions we needed something new that would be compatible with that platform. We took this as an opportunity to address many of the shortcomings of the earlier technology. The Message Security Layer (MSL) was born from these dual concerns.

GITHUB/NETFLIX/MSL

The screenshot shows a GitHub repository page for `Netflix/msl`. The top navigation bar includes links for Code, Issues (29), Pull requests (3), Wiki (highlighted in orange), Releases (41), and More. The repository statistics are 318 Watchers, 471 Stars, 56 Forks. The main content area is titled "Home" and shows a message from Paul Adolph edited on 6 Apr 2016. Below this, the "Introduction" section defines Message Security Layer (MSL) as an extensible and flexible secure messaging framework. It notes that MSL does not solve specific use cases but supports a wide variety of applications and leverages external cryptographic resources. The "Overview" section details how MSL messages are associated with entities, use cryptographic keys for authentication and integrity, and support optional user association and replay protection. It also mentions session key establishment through key exchange mechanisms. The "Entity Authentication" section is mentioned as providing the general approach for authentication and key exchange. On the right side, there is a sidebar titled "Pages (59)" containing a hierarchical list of MSL components and concepts.

Netflix / msl ✓

Watch 318 Star 471 Fork 56

Code Issues 29 Pull requests 3 Wiki Releases 41 More

Home

Paul Adolph edited this page on 6 Apr 2016 · 7 revisions

Introduction

Message Security Layer (MSL) is an extensible and flexible secure messaging framework that can be used to transport data between two or more communicating entities. Data may also be associated with specific users, and treated as confidential or non-replayable if so desired.

MSL does not attempt to solve any specific use case or communication scenario. Rather it is capable of supporting a wide variety of applications and of leveraging external cryptographic resources. There is no one-size-fits-all implementation or configuration; proper use of MSL requires the application designer to understand their specific security requirements.

Overview

Each MSL message is associated with an entity. Cryptographic keys associated with the entity are used to authenticate the entity and integrity protect the message. Data, whether confidential or not, is typically encrypted during transport using cryptographic keys associated with the entity, but non-confidential data may also be transmitted without encryption. Messages are optionally associated with a user by including data that can be used to authenticate the user. Messages are optionally protected against replay through the use of a non-replayable ID synchronized between the communicating entities.

Session key establishment is also supported through the use of key exchange mechanisms. It may be necessary to establish session keys before confidential data can be securely transmitted.

The authentication and key exchange schemes are not explicitly defined and must be provided in order to create a secure deployment of the MSL framework. The general approach by which messages are authenticated and key exchange occurs are documented below in Entity Authentication and Key Exchange.

Pages (59)

- [Introduction](#)
 - [Terminology](#)
 - [Environment Assumptions](#)
 - [Security Objectives](#)
 - [Application Objectives](#)
- [Encoding & Normalization](#)
- [Cryptography](#)
 - [Ciphertext Envelopes](#)
 - [Signature Envelopes](#)
 - [Algorithm Names](#)
- [Versioning](#)
- [MSL Networks](#)
 - [Trusted Services](#)
 - [Peer-to-Peer](#)
 - [Multiple Network Participation](#)
- [Entity Authentication](#)
 - [Entity Authentication Data](#)
 - [Master Tokens](#)
- [User Authentication](#)
 - [User Authentication Data](#)
 - [User ID Tokens](#)
- [Key Exchange](#)
- [Service Tokens](#)
- [Messages](#)

GITHUB/ASCIIDISCO/PLUGIN.VIDEO.NETFLIX

The screenshot shows a GitHub repository page for the plugin.video.netflix repository. The page title is "Netflix Plugin for Kodi 18 (plugin.video.netflix)". Below the title, there is a horizontal bar with various status indicators: "bitcoin donate" (green), "build failing" (red), "test coverage 10%" (red), "code climate" (grey), "302 issues" (red), "maintainability D" (orange), "release v0.12.9" (blue), and "docs passing" (green). A "License MIT" badge is also present. The main content area includes sections for "Disclaimer", "Prerequisites", "Installation & Updates", and "Further installations instructions can be found in the Wiki". The GitHub interface includes a header with a star icon, a fork icon, and other navigation links.

Disclaimer

This plugin is not officially commissioned/supported by Netflix. The trademark "Netflix" is registered by "Netflix, Inc."

Prerequisites

- Kodi 18 [nightlybuild](#)
- Inputstream.adaptive $\geq v2.0.0$ (should be included in your Kodi 18 installation)
- Libwidevine $\geq 1.4.8.970$ (for non Android devices)
- Cryptdome python library (for Linux systems, install using `pip install --user pycryptodomex` as the user that will run Kodi)

Note: The link to download the Widevine Library for non ARM Systems can be found in the [Firefox Sources](#) & needs to be placed in the `cdm` folder in [special://home](#).

Please make sure to read the licence agreement that comes with it, so you know what you're getting yourself into.

Installation & Updates

You can use [our repository](#) to install plugin. Using this, you'll immediately receive updates once a new release has been drafted.

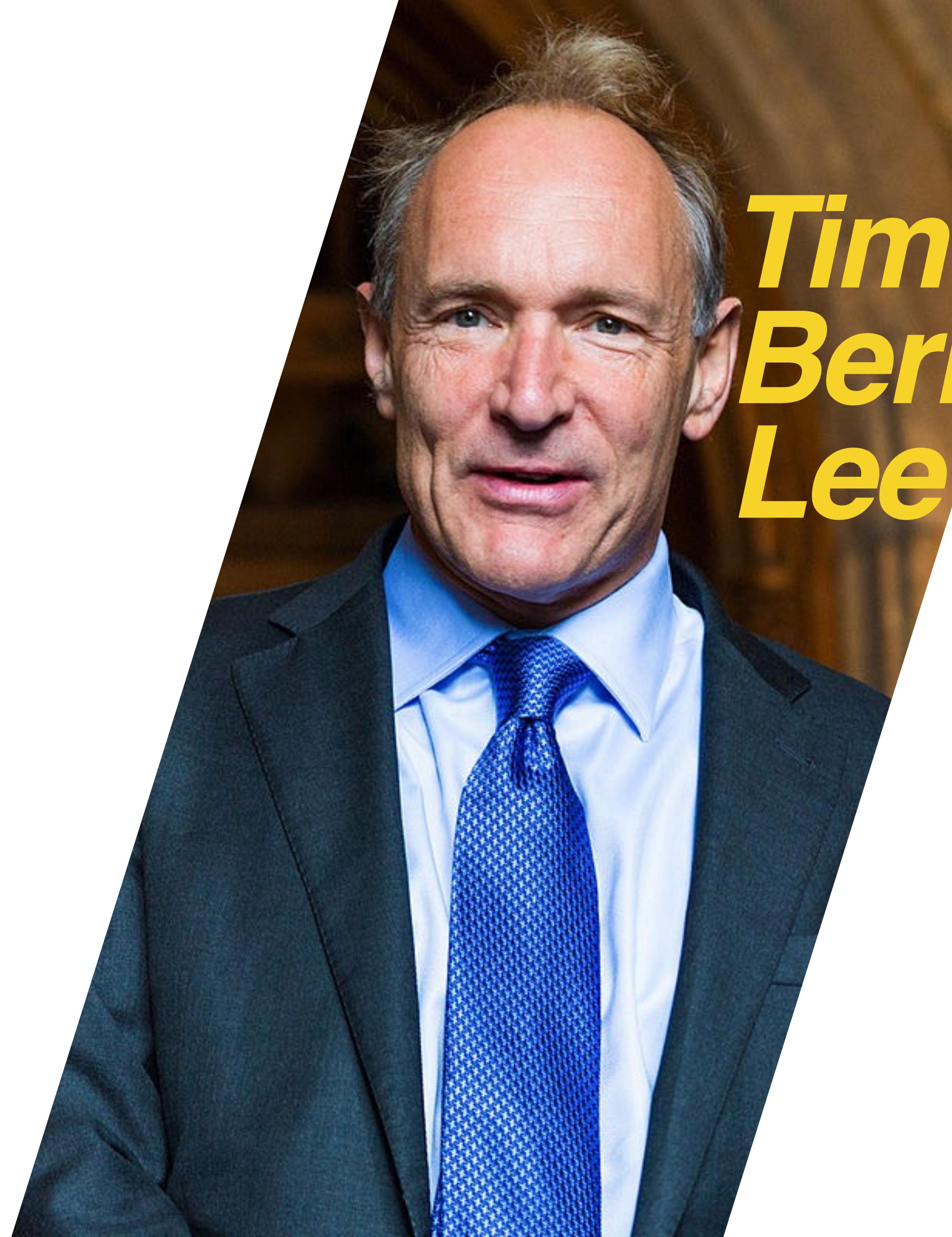
Further installations instructions can be found in the [Wiki](#)



<VIDEO
BLACKBOX=TRUE/>



**Blackbox
Again?!**



**Tim
Berners
Lee**

“So in summary, it is important to support EME as providing a relatively safe online environment in which to watch a movie, as well as the most convenient, and one which makes it a part of the interconnected discourse of humanity”



Electronic Frontiers Foundation

“In 2013, EFF was disappointed to learn that the W3C had taken on the project of standardizing “Encrypted Media Extensions,” an API whose sole function was to provide a first-class role for DRM within the Web browser ecosystem.”



Electronic Frontiers Foundation

“So we'll keep fighting to keep the web free and open. We'll keep suing the US government to overturn the laws that make DRM so toxic, and we'll keep bringing that fight to the world's legislatures.”



Make Up Your
Own Mind

Thank You!

<https://asciidisco.com> / E-mail: public@asciidisco.com / Sebastian Golasch 2018