Opera TV

Application Certification Program

Apps Requirements 2017

Version: 4.10

Date: 2017-03-16

CONTENTS

1. REVISION HISTORY

Version Number	Chromium	Date	Comment
4.9.0	53.0	2016-08-18	Publication release
4.9.0-r1	53.0	2016-11-04	Changes: ● Removed text about terminal-defined behaviour in the exit key. The application may now manage this key event freely. ● The Application MUST handle both VK_BACK_SPACE and VK_BACK. ● The Application MUST support HD, Full HD or both resolutions ● Removed mention of viewport META tag ● The Application MUST explicitly set colors ● Restructured content ● Improved clarity and grammar
4.10	56.0	2017-03-16	Publication release Changes: ● Added “Performance acceptance criteria” ● Added “Annex C: Supported media by HTML5 MediaElement” table

2. INTRODUCTION

2.1 Scope

This document specifies the requirements for Opera TV Applications and provides guidelines on how to develop Opera TV-compliant apps. It is primarily aimed at app developers and owners.

This document is closely connected to the Opera TV Device Specification [1] which defines the platform that Opera TV Applications run on.

2.3 Abstract

To have an app certified for Opera TV, the owner or developer must submit it for review. This document defines the requirements and criteria for certification and also gives guidelines for developers on how to make apps function well on devices integrating the Opera Devices SDK.

While applications may use any available web technologies outside these specifications to provide an enhanced experience for devices with a richer feature set, the application must provide graceful fallbacks outlined within these specifications to provide the same basic functionality to all certified Opera TV Devices.

2.4 Definitions

Chromium and **Google Chrome** - Chromium is the open-source project that forms the basis for the Google Chrome browser.

Opera Devices SDK - An embeddable browser and streaming engine with an extensible API, based on the Chromium open-source project, which implements a set of international and industry standards to download and render webpages, execute web apps, and stream video and audio content.

Opera TV - A set of products and solutions developed by Opera TV AS for TV and Set-Top Box (STB) manufacturers to enable HTML5 rendering and adaptive streaming in their devices.

Opera TV Application - A web app that is Opera TV-compliant and is certified to run on Opera TV Devices. Also referred to as App or Application in this document.

Opera TV Device - A TV or STB device running software based on the Opera Devices SDK, and certified to meet the requirements defined in the Opera TV Device Specification [1].

Opera TV Device Specification or **Device Specification** specifies platform requirements that an Opera TV Device must meet to be officially certified, and, therefore specifies the platform that an app can expect when running on an Opera TV Device.

See also: ABBREVIATIONS

2.5 Compliance terminology used in this document

The following keywords used in this specification to indicate the level of compliance needed for the requirements are sourced from RFC2119. In essence:

- MUST, REQUIRE or SHALL indicates that you need to comply with the requirement absolutely.
- SHOULD or RECOMMENDED indicates that while sometimes there could be valid reasons to ignore the requirement, you need to fully understand and accept the implications and risks to optimal end-user experience.
- MAY or OPTIONAL indicates that you can decide to implement an item at your discretion.

2.6 Changes and Versioning

When the Application Requirements or the Device Specification [1] are changed, the version number of both documents will be updated to match the corresponding Opera Devices SDK version.

2.6.1 Backward compatibility

New versions of these requirements may not be backward compatible with previous versions. Web standards and technologies are evolving fast; some older specifications may be replaced with more modern features, and their support may be removed. The deprecation process is relatively slow, but cannot be totally avoided. This specification and the Device Specification list features which were deprecated since the previous version and also mark features which might be deprecated in later versions. You should be careful if using features marked for possible deprecation.

2.6.2. Specification and Software Versions

The Application Requirements document defines compliance requirements for a particular version of the Opera TV Device Specification [1], Opera Devices SDK and Chromium. The table below defines corresponding versions:

Device Specification version	Opera Devices SDK version	Google Chromium version
4.10	4.10.0 LTS	56.0.2924.x

3. REQUIREMENTS

This section outlines the requirements for an app to be certified to run on Opera TV Devices.

3.1 Browser engine

An Opera TV Application runs in a browser engine based on Chromium so almost all Chromium features and Chromium-supported web standards are available to app developers. However, a TV is a limited device, normally without keyboard, mouse or touch-screen support, and without a desktop user interface. A typical TV also has a greater screen size, runs apps in fullscreen, and is designed to be viewed at a distance, while having a slower CPU and less memory (RAM) than a typical PC, tablet or mobile phone.

3.1.1 User Agent string

Apps should test for the presence of specific features whenever possible, and SHOULD NOT associate specific User Agent strings with device feature sets. For more information, refer to the Device Specification [1] section 4.8. “User Agent String”.

If the app should be loaded through the Opera TV Store, it can search for the string “TV Store” in the User Agent string.

3.2 Standards compliance

3.2.1. Web specifications

Due to the nature of TV devices, not all features available in desktop and mobile browsers are supported.

Apps MUST NOT rely on the following browser features; they are not supported by Opera TV Devices. Note that this list is not exhaustive or exact, but aims to highlight some important exclusions. There may be other features that are unavailable, or have limited functionality compared to a desktop or mobile browser.

- Adobe Flash
- Audio Output Device API [14]
- Device Orientation [4]
- Downloaded Plugins (including NPAPI and PPAPI)
- Drag and Drop Directories [5]
- Extensions
- FTP
- Geolocation [6]
- High Resolution Timestamp [15]
- JavaScript dialogs
- Password manager
- Pointer Events [7]
- Touch Events [10]
- Vibration API [8]
- Web Audio [13]
- Web Notifications [9]
- Web Speech API [12]
- WebRTC [11]

To assess web standards compliance and the availability of a particular feature you can check these online resources:

- "Can I Use" (http://caniuse.com/)
- "Chromium Platform Status" (http://chromestatus.com/)

Look for the Chrome (Chromium) version in *Specification and Software versions* above.

3.2.2 HbbTV, CE-HTML and OIPF

Apps MUST NOT rely on HbbTV [16] and OIPF [18] standards and related profiles (Freeview Play, TDT Híbrida, TNT2). Support for these standards is out of scope of this specification, even though the Opera Devices SDK itself supports these standards.

Apps MUST NOT rely on the CE-HTML video object defined in CEA-2014 [17] since it is not necessarily supported by Opera TV Devices.

3.2.3 WebGL

Apps MUST NOT rely on the WebGL [19] standard. It is not a mandatory requirement for Opera TV Devices so it will not be available on all devices.

This is normally supported on high-end devices, but typically require more platform resources than CSS/HTML/JavaScript.

The app can detect the availability of WebGL as follows:

```
if (!window.WebGLRenderingContext) {
    console.log("WebGL is not supported by browser");
} else {
    var canvas = document.createElement("canvas");
    var context = canvas.getContext("webgl");
    if (!context) {
        console.log("WebGL is supported by browser but disabled");
    } else {
        console.log("WebGL is supported by browser and enabled");
    }
}
```

3.3 Media streaming and DRM

This section highlights the supported streaming and DRM protocols by Opera TV Devices. See Annex C: Supported media by HTML5 MediaElement for a complete table.

3.3.1 Progressive downloads

Progressive download of video and audio is supported by Opera TV devices, but note that:

- Protected content (DRM) is not supported with progressive download
- In-band subtitles are not supported with progressive download
- Out-of-band subtitles are supported with any streaming or download as they are handled outside of the media streaming

Apps MAY rely on support for the following combinations on Opera TV Devices:

Container	Audio codec	Video codecs
ISO BMFF (MPEG4)	AAC-LC HE-AAC v1 HE-AAC v2 MP3 Dolby AC3	H.264 H.265

	Dolby E-AC-3	
MPEG2-TS	AAC-LC HE-AAC v1 HE-AAC v2 MP3 Dolby AC3 Dolby E-AC-3	H.264
WebM	Opus	VP8 VP9
ADTS / AAC MP3	AAC-LC HE-AAC v1 HE-AAC v2 MP3	None

3.3.2 Adaptive Bitrate protocols

Apps MAY rely on support for the following Adaptive Bitrate Streaming (ABR) protocols:

Streaming Type	MIME-Types	Notes
Apple HTTP Live Streaming (HLS)	application/vnd.apple.mpegurl application/x-mpegURL	VoD (static playlists) and Live (dynamic playlists)
MPEG-DASH	application/dash+xml	Main profile and Live profile of MPEG-DASH
Microsoft Smooth Streaming (MSS)	application/vnd.ms-sstr+xml application/vnd.ms-playready.initiator+xml	

3.3.3 Media Source Extensions (MSE)

Apps MAY rely on Media Source Extensions to be supported according to the MSE specification [30]. The following combinations of containers and codecs are supported:

Container	Audio codec	Video codec
MP4	AAC / MP3	H.264 / H.265
WebM	Opus	VP8 / VP9
MP4	AAC / MP3	
WebM	Opus	
MP4		H.264 / H.265
WebM		VP8 / VP9

3.3.4 Subtitles & Closed captioning

Apps MAY rely on support for in-band and out-of-band subtitles (text tracks) according to the table below:

Media delivery method	In-band Subtitles	Out-of-band Subtitles

Progressive playback	Not supported	Supported
HLS	Not supported	Supported
MPEG-DASH	Supported	Supported
Smooth Streaming	Supported	Supported
MSE	Not supported	Supported

3.3.5 DRM

Apps MAY rely on support for the following DRM technologies:

3.3.5.1 ClearKey

- Supported with EME

3.3.5.2 PlayReady

- Supported with EME
- Supported with WebInitiator
- PlayReady Header Object v4.0.0.0 is supported
- PlayReady Header Object v4.1.0.0 may be supported
- Supported for security level "2000" or higher

For an example of implementing PlayReady with WebInitiator, see: Annex A.

3.3.5.3 Widevine

- Supported if the platform has Widevine DRM installed
- If supported, supported with EME
- If supported, supported with security level “L1”
- MAY support "server certificate" and "privacy mode" features
- MAY support persistent licences

3.3.5.4 AES-128 encryption

- Supported for Apple HLS streams

For more information about Media streaming and DRM, refer to the Device Specification [1].

Note: EME can only be used on secure contexts, it can not be used on any pages served over HTTP. App developers MUST use a secure origin (HTTPS).

3.4 Performance

Performance requirements for Opera TV Applications are significantly stricter than for web applications for PC or mobile. However performance considerations, recommendations for enhancements, and optimization approaches are similar across devices and can be applied to all Opera TV Applications.

It is important to understand that even high-end TV and STB devices have very limited computational, memory and graphics resources. Even though new device models are refreshed and manufactured regularly, the aforementioned resources are unlikely to significantly improve on the previous year's models.

The average specification of a mid-range TV may look like the following:

- Low-range CPU, often under 1GHz, dual-core ARM or MIPS architecture
- Less or equal to 1GB of RAM
- Low range GPU (e.g. Mali-400) with limited or no video memory
- Large screen with Full HD (1920 x 1080, 2M pixels), or Ultra HD/4K (3840 x 2160, 8M pixels)

This makes TV devices much slower than an average PC. Note also that most of these resources are reserved for the platform, operating system and browser engine, so there may be as little as 150MB out of 1GB available for the app.

3.4.1. Memory consumption

Apps SHOULD NOT use more than 150MB of memory.

Memory consumption affects app performance, and as mentioned above, the amount of memory available for apps is very limited on devices. In addition, TV devices typically have a hard memory limit so if an app uses a lot of memory, it could suffer from performance degradation or be killed.

You should be aware of the size of your DOM tree, and prune away unnecessary nodes, as this can be a major factor of uncollected memory consumption.

3.4.2 Graphics requirements and animations

Apps SHOULD NOT use heavy graphics and complex animations.

Even though modern TV devices are now using specialized GPU chipsets, their graphics performance is usually still quite modest.

3.4.3 Hardware-accelerated features

Apps MAY NOT rely on CSS 3D transforms [27] or WebGL [19] features.

Unfortunately, not all TV devices support OpenGL ES 2.0 [28].

3.4.4 Performance acceptance criteria

Applications MUST meet the following performance requirements on Opera TV Devices.

Opera will reject applications that are unable to meet these performance requirements, based on tests run on Opera TV reference devices, and commercial devices currently in development. If your application is unable to meet them, Opera TV will engage with you and attempt to help you reproduce performance issues on commercially available hardware and resolve them.

3.4.4.1 Load time

The time between when the browser first starts loading the application, and when the application is loaded to the point where it is functional, MUST NOT exceed 30 seconds.

3.4.4.2 Internal navigation

The time between when the user initiates an internal navigation within the application, for instance navigating within a menu or selecting between different videos, and when the user is able to activate the new selection, MUST NOT exceed 1 second.

3.4.4.3 Action latency

The time between when the user performs any kind of action on the application, and when the user receives feedback that the action has been initiated, MUST NOT exceed 4 seconds.

3.5 Security

3.5.1 Transport Layer Security (TLS)

You should always protect your apps with HTTPS, even if there is no sensitive communication. HTTPS provides critical security and data integrity both for the app and for users.

3.5.2 Same-Origin policy

Opera TV Devices apply the Same-Origin policy as mentioned in section 4.10 "Security" in the Device Specification [1]. This is sometimes less permissive than other browser engines and previous Opera versions, so apps may need to adapt if they are to be ported from such a platform. To prevent apps breaking the Same-Origin policy, use Cross-Origin Resource Sharing (CORS) [20].

3.5.3 Mixed content

Apps MUST NOT use any kind of active mixed content (mixing secure and insecure requests). Insecure requests will be blocked by the browser engine as they are serious security vulnerabilities.

For information about the security aspects of mixed content and how to fix them, refer to the W3C specification, or articles on the internet, such as "Prevent mixed content" by Google [22] or "Mixed content" by Mozilla [23].

3.5.4 Pop-ups

Apps MUST NOT use any pop-up windows or dialogs. Pop-ups are blocked by Opera TV Devices.

3.6 Input handling

3.6.1 Key mapping

Apps SHOULD NOT use numerical key codes directly.

Opera TV Devices provides standardized key codes as global JavaScript constants to use in apps, as the numerical value of key codes differ between devices (see the Device Specification, section 4.7. Input handling [1]).

Hardware key	JavaScript constant	Availability
←	VK_LEFT	Always present in remote controls

→	VK_RIGHT	Always present in remote controls
↑	VK_UP	Always present in remote controls
↓	VK_DOWN	Always present in remote controls
Confirm / Select / OK	VK_ENTER	Always present in remote controls
Back / Return	VK_BACK	Always present in remote controls
Back / Return	VK_BACK_SPACE **	Always present in remote controls
Exit / Close ***		Usually present in remote controls*
BLUE	VK_BLUE	Usually present in remote controls*
RED	VK_RED	Usually present in remote controls*
GREEN	VK_GREEN	Usually present in remote controls*
YELLOW	VK_YELLOW	Usually present in remote controls*
Menu	VK_MENU	Not available in some remote controls*
0	VK_0	Not available in some remote controls*
1	VK_1	Not available in some remote controls*
2	VK_2	Not available in some remote controls*
3	VK_3	Not available in some remote controls*
4	VK_4	Not available in some remote controls*
5	VK_5	Not available in some remote controls*
6	VK_6	Not available in some remote controls*
7	VK_7	Not available in some remote controls*
8	VK_8	Not available in some remote controls*
9	VK_9	Not available in some remote controls*
PLAY	VK_PLAY	Not available in some remote controls*
PAUSE	VK_PAUSE	Not available in some remote controls*
STOP	VK_STOP	Not available in some remote controls*
NEXT	VK_TRACK_NEXT	Not available in some remote controls*
PREV	VK_TRACK_PREV	Not available in some remote controls*
FF (Fast-Forward)	VK_FAST_FWD	Not available in some remote controls*
REWIND	VK_REWIND	Not available in some remote controls*
SUBTITLE	VK_SUBTITLE	Not available in some remote controls*
INFORMATION	VK_INFO	Not available in some remote controls*

Not available in some remote controls and **Usually present in remote controls** * - not all devices have a corresponding physical key on the remote control and are conditionally required by the Device Specification [1].
VK_BACK_SPACE ** - handling this key code is important for compatibility with the Opera TV Store and older versions of the Opera Devices SDK.
Exit / Close *** - this hardware key will be handled by the browser and can not be overridden by the application.

3.6.1.1 Navigation and Select keys

The Navigation and Select keys are mandatory and result in sending corresponding key codes on key events into the app. If the navigation keys are not handled and prevented by the app, this MIGHT trigger automatic (spatial) navigation, but the result of this is undefined.

3.6.1.2 Back key

The Back/Return button is a mandatory button on the remote control to go back or close the app.

The app MAY handle the Back/Return key event. If it does handle the Back/Return key, the app MUST handle both VK_BACK and VK_BACK_SPACE to ensure compatibility between legacy and current devices. On the remote control, the key may be marked with "Back", "Return" or similar. The Back/Return key SHOULD provide the user with typical back navigation, and finally, an exit path via `window.close()` to leave the app and return to the previous screen (for example, the Opera TV Store if the app was launched from there).

There are additional requirements for apps that are deployed in the Opera TV Store. For more details, refer to Functional Key Handling in Opera TV Store Applications [33].

3.6.1.3 Color keys

The blue, green, red, and yellow color keys are conditionally required on remote controls that have them, however, the placement and order of color keys will differ depending on device manufacturer and geographical region.

3.6.2 Virtual Keyboard

Apps MAY rely on the On-Screen Keyboard or Virtual Keyboard provided on Opera TV Devices to allow the user to enter text. Do not make any assumptions about the visual appearance of this keyboard as this depends on the device make and model. Be aware that when activated, the Virtual Keyboard may be displayed over any portion of the screen, and can cover any part of the app. This sometimes leads to design inconsistencies.

If you want to change the look and feel of text inputs to match the particular design of your app, you MAY implement your own text input mechanism in JavaScript (for example, using a div marked as `contenteditable` instead of a text field).

Also be aware that on some devices the Virtual Keyboard only supports a limited set of languages.

3.6.3 External input devices

Apps MUST NOT reply on the availability of external input devices such as a pointer remote, USB keyboard and mouse, even though some Opera TV Devices support these.

3.7 User interface and usability

3.7.1 TV screen size and resolution

Design apps for at least one of the following screen resolution configurations:

1. HD resolution (1280 x 720)
2. Full HD resolution (1920 x 1080)
3. Both HD and Full HD resolutions can be supported by one application

Important notes:

Manufacturers do not set the physical screen size or pixel ratio in the firmware, so this information is not available for apps.

The media player may be able to play video in other resolutions than the resolution that the app is rendered in. Use the JavaScript `canPlayType` method on the HTML5 Video element to detect this.

3.7.2 Overscan

Be aware of possible overscan problems where margins of your app might be shown outside the visible area of the TV screen. As a general guide, assume that a 4% margin might not be visible to the user when designing your app. Test to ensure that the app works and displays correctly with margins applied for visible area.

This is because TV sets, even the most recent, large Ultra HD/4K models, may set a certain amount of overscan by default. The amount of overscan varies from manufacturer and model. While it is possible for users to turn off overscan, we recommend that you design your app with this invisible margin in mind as most users are likely unaware of this option.

3.7.3 Layout

3.7.3.1 Text Size

Text size MUST be large enough to be seen on a TV screen from 3 meters (10 feet) away: the average distance that users sit from the TV.

The minimum text size RECOMMENDED for Opera TV Applications is 3/100 of the screen height, or "3vh" in CSS (this gives approximately 32px at 1920 x 1080px):

```
p { font-size: 3vh; }
```

See Viewport-percentage lengths in *CSS Values and Units Module Level 3* [24].

3.7.4 Colors

Apps MUST explicitly set colors for all visual elements including text and page background colors.

The app MUST not rely on any default colors defined by the browser engine.

```
* { background-color: #000000; color: #ffffff; }
```

3.7.5 Navigation

Apps MUST NOT rely on platform-specific spatial navigation mechanisms and SHOULD instead provide navigation implemented by the app (e.g. in JavaScript).

Spatial navigation SHOULD be explicitly blocked (by calling Event.preventDefault) if the app implements its own navigation mechanism.

For an example of implementing navigation with JavaScript, see: Annex B: JavaScript navigation.

3.7.6 App exit

You SHOULD provide an explicit option or button in the UI to close the app with a simple call to the `window.close()` method.

This is because when an app is launched, it opens in a full screen in a new window on the TV, it runs completely “chromeless”, with no address bar or user interface controls. Users will be able to close the app and return to the main TV screen (or dashboard) via the remote control's Exit/Close and/or Back/Return" keys (as mentioned above) but it's a good idea to provide another, more explicit option.

3.7.7. Inputs and forms

Apps SHOULD NOT use input elements or forms as these don't provide a good user experience on a TV, even though Opera TV Devices support these elements and also provide a Virtual Keyboard (On-Screen Keyboard). This includes the use of <input> of all types, <textarea>, <form>, <fieldset>, <select>, <option>, <optgroup>, <datalist>, <keygen> elements.

3.7.8 Fonts

You SHOULD NOT rely on built-in fonts but instead use downloadable web fonts.

An Opera TV Device provides Sans Serif fonts containing all the characters required to properly render the text in the languages available on the device, but the exact font used may vary among device models. Right-to-left text rendering is also supported if required by any language available on the device. However, the identity, look and language coverage of these fonts cannot be guaranteed across all devices.

Consider the following when using web fonts:

- Font resources MUST be in the WOFF [31] / WOFF2 [32] (Web Open Font Format) or TTF (TrueType Font) file format.
- Font resources SHOULD NOT be large, otherwise they can take a long time to download and also consume a lot of memory and CPU power.
- The Font Loading API [25] can be used to improve text rendering performance.

3.8 Data management

3.8.1 Temporary and persistent storage

Be aware that devices have very limited storage capacity. For more information, refer to the *Device Specification [1], section 4.12.5. "Storage".*

An app SHALL NOT rely on Persistent Storage Quota request when using the Quota Management API [26].

Chrome 'unlimitedStorage' permission is not available to Opera TV Applications.

3.8.2 Cookies

An app MAY use up to 180 cookies per domain and each cookie will be available for minimum 30 days.

The maximum number of cookies stored in the Opera TV Device database is set to 2000, each using a maximum of 4096 bytes.

3.9 Supplementary multimedia devices

Apps MUST NOT rely on the microphone or camera being available on the device.

4. ABBREVIATIONS

API	Application Programming Interface
CEA-2014	Consumer Electronics Association standard 2014 [17]
CE-HTML	Consumer Electronics Hyper Text Markup Language [17]
CORS	Cross-Origin Resource Sharing [20]
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DOM	Document Object Model
DRM	Digital Rights Management
FTP	File Transfer Protocol
HbbTV	Hybrid Broadcast Broadband TV [16]
HLS	HTTP Live Streaming
HTML5	Hyper Text Markup Language specification version 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISO	International Organisation of Standardization
ISO BMFF	ISO Base Media File Format
MP4	MPEG-4 Part 14
MPEG	Moving Picture Experts Group
MPEG2-TS	MPEG2 Transport Stream
MSE	Media Source Extensions [30]
MSS	Microsoft Smooth Streaming
NPAPI	Netscape Plugin Application Programming Interface
OEM	Original Equipment Manufacturer
OIPF	The Open IPTV Forum [18]
PPAPI	Pepper Plugin Application Programming Interface
SDK	Software Development Kit
STB	Set-Top Box
TTF	TrueType Font

WebGL	Web Graphics Library [19]
WOFF	Web Open File Format [31]
WOFF2	WOFF File Format 2.0 [32]

5. REFERENCES

[1] Opera TV Device Specification
http://dcp.otvs.tv/doc/DeviceSpecificationDCP.html

[2] “Can I Use” on-line service
http://caniuse.com/

[3] “Chrome Platform Status” on-line service
https://www.chromestatus.com/

[4] DeviceOrientation Event Specification (W3C Editor's Draft 26 February 2016)
http://dev.w3.org/geo/api/spec-source-orientation.html

[5] WHATWG DragAndDropEntries
http://wiki.whatwg.org/wiki/DragAndDropEntries

[6] Geolocation API Specification (W3C Recommendation 24 October 2013)
http://www.w3.org/TR/geolocation-API/

[7] Pointer Events (W3C Recommendation 24 February 2015)
http://www.w3.org/TR/pointerevents/

[8] Vibration API (W3C Recommendation 10 February 2015)
https://www.w3.org/TR/vibration/

[9] Web Notifications (W3C Recommendation 22 October 2015)
http://www.w3.org/TR/notifications/

[10] Touch Events (W3C Recommendation 10 October 2013)
http://www.w3.org/TR/touch-events/

[11] WebRTC 1.0: Real-time Communication Between Browsers (W3C Working Draft 31 May 2016)
https://www.w3.org/TR/webrtc/

[12] Web Speech API Specification (W3C, 19 October 2012)
https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

[13] Web Audio API (W3C Working Draft 08 December 2015)
https://www.w3.org/TR/webaudio/

[14] Audio Output Devices API (W3C Working Draft 01 June 2016)
https://www.w3.org/TR/audio-output/

[15] High Resolution Time Stamp for Events (Chrome Platform Status)
https://www.chromestatus.com/feature/5523910145605632

[16] HbbTV
https://www.hbbtv.org/

[17] CE-HTML and CEA-2014
https://en.wikipedia.org/wiki/CE-HTML

[18] Open IPTV Forum (OIPF) specifications
http://www.oipf.tv/specifications/

[19] WebGL Specification (Khronos Group, Version 1.0.3, 27 October 2014)
https://www.khronos.org/registry/webgl/specs/1.0/

[20] Cross-Origin Resource Sharing (W3C Recommendation 16 January 2014)
https://www.w3.org/TR/cors/

[21] Mixed Content (W3C Candidate Recommendation, 08 October 2015)
https://www.w3.org/TR/mixed-content/

[22] Prevent mixed content (Google Developers, Progressive Web App Dev Summit)
https://developers.google.com/web/fundamentals/security/prevent-mixed-content/

[23] Mixed content (Mozilla Development Network)
https://developer.mozilla.org/docs/Web/Security/Mixed_content

[24] CSS Values and Units Module Level 3 (W3C Candidate Recommendation, 11 June 2015)
Viewport-percentage lengths
https://www.w3.org/TR/css3-values/#viewport-relative-lengths

[25] CSS Font Loading Module Level 3 (W3C Last Call Working Draft, 22 May 2014)
https://www.w3.org/TR/css-font-loading/

[26] Quota Management API (W3C Working Group Note 23 May 2016)
https://www.w3.org/TR/quota-api/

[27] CSS Transforms Module Level 1 (W3C Working Draft, 26 November 2013)
https://www.w3.org/TR/css-transforms-1/

[28] OpenGL ES 2.0 (The Standard for Embedded Accelerated 3D Graphics)
https://www.khronos.org/opengles/2_X/

[29] CSS Device Adaptation Module Level 1, Chapter 10: Viewport <META> element
https://drafts.csswg.org/css-device-adapt/#viewport-meta

[30] Media Source Extensions (W3C Candidate Recommendation 5 July 2016)
https://www.w3.org/TR/media-source/

[31] WOFF File Format 1.0 (W3C Recommendation 13 December 2012)
https://www.w3.org/TR/WOFF/

[32] WOFF File Format 2.0 (W3C Candidate Recommendation 15 March 2016)
https://www.w3.org/TR/WOFF2/

[33] Functional Key Handling in Opera TV Store Applications
https://dev.opera.com/tv/functional-key-handling-in-opera-tv-store-applications/

6. Annex A: DRM code Examples

6.1 PlayReady Web-Initiator licence acquisition

Example of usage in HTML5 <video> tag.

```
<video>
    <source src="http://example.com/example_webinitiator.xml"
type="application/vnd.ms-playready.initiator+xml" />
</video>
```

Example of PlayReady WebInitiator. <LicenseAcquisition> tag MUST contain <Content> tag, which is URL to MSSS manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<PlayReadyInitiator xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/">
    <LicenseAcquisition>
        <Content>URL to MSSS manifest</Content>
        <Header>
            <WRMHEADER
xmlns=http://schemas.microsoft.com/DRM/2007/03/PlayReadyHeader version="4.0.0.0">
                <DATA>
                    <PROTECTINFO>
                        <KEYLEN>16</KEYLEN>
                        <ALGID>AESCTR</ALGID>
                    </PROTECTINFO>
                    <LA_URL>http://example.playready.com/rightsmanager.asmx
                    </LA_URL>
                    <KID>ASNFZwEjRWcBI0VnASNFZw==</KID>
                    <CHECKSUM>ASNFZwEjRWc=</CHECKSUM>
                </DATA>
            </WRMHEADER>
        </Header>
        <CustomData>Optional CustomData</CustomData>
    </LicenseAcquisition>
</PlayReadyInitiator>
```

Note that persistent licences MAY be supported but web app MUST BE prepared to apply the license again if the feature is not supported.

7. Annex B: JavaScript navigation

Example of handling navigation with JavaScript.

```
document.addEventListener("keydown", function(ev) {
    switch (ev.keyCode) {
        case VK_LEFT:
            // Handle mandatory key ←
            break;
        case VK_RIGHT:
            // Handle mandatory key →
            break;
        case VK_UP:
            // Handle mandatory key ↑
            break;
        case VK_DOWN:
            // Handle mandatory key ↓
            break;
        case VK_ENTER:
            // Handle mandatory key Confirm / Select / OK
            break;
        case VK_BACK:
        case VK_BACK_SPACE:
            // Handle mandatory key Back / Return
            break;
    }
    // Block the browser from handling the keydown event.
    ev.preventDefault();
}, false);
```

8. Annex C: Supported media by HTML5 MediaElement

Streaming Type	Container	Audio Codec	Video Codec		DRM	DRM Trigger	Inband subtitles	Mime type
Progressive	MP4 (ISO BMFF)	AAC-LC	H.264	H.265 *	None	None	None	video/mp4
		HE-AAC v1						
		HE-AAC v2						
		MP3						
		Dolby AC3 *						
		Dolby E-AC-3 *						
	MPEG2-TS	AAC-LC	H.264		None	None	None	video/mp2t
		HE-AAC v1						
		HE-AAC v2						
		MP3						
		Dolby AC3 *						
		Dolby E-AC-3 *						
	WebM *	Opus *	VP8 *	VP9 *	None	None	None	video/webm
	ADTS / AAC	AAC-LC	None		None	None	None	audio/aac
		HE-AAC v1						
		HE-AAC v2						
	MP3	MP3	None		None	None	None	audio/mpeg
Adaptive (HLS)	MPEG2-TS	AAC-LC	H.264	H.265 *	AES-128	Manifest	None	application/vnd.apple.mpegurl application/x-mpegURL
		HE-AAC v1						
		HE-AAC v2						
		MP3						
		Dolby AC3 *						
		Dolby E-AC-3 *						

		AAC-LC						
	ADTS	HE-AAC v1	None		AES-128	Manifest	None	
		HE-AAC v2						
	MP3	MP3	None		AES-128	Manifest	None	
		AAC-LC			ClearKey			
		HE-AAC v1			PlayReady	EME		
		HE-AAC v2			Widevine *			
Adaptive (DASH)	MP4	MP3	H.264	H.265 *			EBU-TT-D WebVTT	application/dash+xml
		Dolby AC3 *			PlayReady	Manifest		
		Dolby E-AC-3 *						
		AAC-LC			PlayReady	Web Initiator	EBU-TT-D WebVTT	application/vnd.ms-playready.initiator+xml
Adaptive (MSSS)	PIFF	HE-AAC v1	H.264		PlayReady	Manifest	EBU-TT-D WebVTT	application/vnd.ms-sstr+xml
		HE-AAC v2			PlayReady	EME	EBU-TT-D WebVTT	application/vnd.ms-sstr+xml

Note1: * H.265, VP8, VP9, WebM, Opus, Dolby AC3, Dolby E-AC-3, Widevine are conditionally required by the Device Specification [1] and might not be supported by devices. Apps MUST NOT rely on those technologies.

Note2: All streams are supported with out of band subtitles (EBU-TT-D, WebVTT)

Note3: Encrypted text tracks in container are not supported.