YouTube Living Room HTML5 Operator Technical Requirements 2019

Version: 2018/08/14

Purpose of this Document

This document details the base requirements for supporting the YouTube TV HTML5 applications for Living Room Devices provided by Internet Network and Service Operators. The Software Requirements for Operator Devices specification is intended to provide a low-level description of the standards, capabilities, and configuration details necessary for the application to function.

The requirements listed in the “YouTube Software Requirements for CE Devices” specification shall take precedence on any requirements defined herein if the Operator Device is also available in retail.

Meeting the Technical Requirements herein does not constitute a certification approval by itself. Certification approval is communicated by YouTube only after full review and verification of the results.

Changes and Versioning

When changes to the Technical Requirements are required, the version number of this document will be updated to reflect the date on which those changes were published. Direct notice will also be given when material changes are made to this document, and YouTube will provide the maximum possible notification time when major changes are necessary.

Terminology and Format

The key words "MUST", "MUST NOT", and "MAY" in the normative parts of this specification are to be interpreted as described in RFC 2119. An abridged list is included below for reference:

- **MUST** - This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
- **MAY** - This word, or the adjective "OPTIONAL", means that an item may or may not be implemented at the discretion of the implementer.

To clarify certain requirements, examples, background information, or other more detailed but non-normative information may be presented alongside the requirement. This information will be contained within a yellow box as below:

Additional information here.

This information is non-normative.

This information is presented to aid in understanding the requirement or related application behavior and is not a part of the official Technical Requirements specification.

Definitions

The term "YouTube application" will herein reference the main YouTube HTML5 application running in Living Room devices (incl. TVs, Game Consoles, OTT devices and Set-Top-Boxes). Other YouTube applications running on the same devices, including the "YouTube Kids Living Room Application" - see https://kids.youtube.com - and the "YouTube TV Living Room Application" - see https://tv.youtube.com - are subject to the same set of technical requirements unless differences are specifically called out in this document.

Requirements

At a base level, the target device **MUST** be able to:

1. Load, parse, and render HTML.
2. Load, parse, and execute JavaScript.
3. Load, parse, and apply CSS.
4. Respond to standard keyboard and/or remote control events.

The following sections go into detail for each of these requirements.

1.0 HTML

1.1 The target device **MUST** support the following HTML elements:

- BODY

- BR
- DIV
- HEAD
- HTML
- LINK
- SCRIPT
- SPAN
- STYLE
- VIDEO

2.0 JavaScript

2.1 The target device **MUST** support ECMA-262 6th edition (including both strict mode code and non-strict mode code)
2.2 The target device **MUST** include column numbers in JavaScript stack traces.

3.0 Web APIs

3.1 The target device **MUST** support DOM Level 1:

- Core
- HTML

3.2 The target device **MUST** support DOM Level 2:

- Core
- CSS/Style
- Events

3.3 The target device **MUST** support the following DOM Level 2 events:

- Document.load
- Document.unload
- Document.resize
- Document.focus
- Document.blur

3.4 The target device **MUST** support the following DOM methods and attributes:

- Document.createElement
- Document.createTextNode
- Document.getElementsByTagName
- DocumentEvent.createEvent
- Node.getElementById
- Node.appendChild
- Node.removeChild
- Node.insertBefore

- Node.insertAfter
- Element.getAttribute
- Element.setAttribute
- Element.removeAttribute
- HTMLAnchorElement.focus
- HTMLInputElement.focus
- HTMLSelectElement.focus
- EventTarget.addEventListener
- EventTarget.removeEventListener
- EventTarget.dispatchEvent
- Event.preventDefault
- Event.stopPropagation
- Window.addEventListener
- Window.removeEventListener
- Window.setInterval
- Window.clearInterval
- Window.setTimeout
- Window.clearTimeout
- Window.close

3.5 The target device **MUST** support the following DOM attributes:

- HTMLDocument.cookie
- HTMLDocument.body
- HTMLElement.className
- Document.documentElement
- HTMLElement.nodeName
- HTMLElement.nodeType
- HTMLElement.id
- HTMLElement.innerHTML
- HTMLElement.textContent
- HTMLElement.style
- HTMLElement.getBoundingClientRect
- Node.attributes
- Node.childNodes
- Node.firstChild
- Node.parentNode
- Node.nextSibling
- Node.previousSibling
- Window.navigator
- Window.location

3.6 The target device **MUST** expose the following DOM-accessible elements:

- Window.performance.memory.totalJSHeapSize
    - This element **MUST** return the total JavaScript heap size in bytes.
- Window.performance.memory.usedJSHeapSize
    - This element **MUST** return the currently allocated JavaScript heap size in bytes.

3.7 The target device **MUST** support DOM Level 3 Events.
3.8 The target device **MUST NOT** make any feature accessible if the underlying function has not been properly implemented by the device.
3.9 The target device **MUST** support Timing control for script-based animations, also known as requestAnimationFrame.
3.10 The target device **MUST** make JavaScript stack traces available from window.onerror as defined in section 8.1.5.2 of the following HTML specification:
https://html.spec.whatwg.org/multipage/webappapis.html#handler-onerror

4.0 CSS

4.1 The target device **MUST** support Web Fonts, including support for the following formats:

- TTF
- WOFF
- WOFF2

4.2 The target device **MUST** use the web font that is served by the YouTube application.
4.3 If the target device supports system languages with characters not included in the font(s) served by the YouTube application, it **MUST** provide the YouTube application with system fonts containing those characters.

The YouTube application will attempt to use fonts built into the YouTube application depending on the characters needed and what external fonts are available on the target device. As complete font files covering Chinese, Japanese, Korean, and other non-Latin character sets have large file sizes, including them directly in the YouTube application is infeasible due to possible memory limitations on a target device.
This information is non-normative.

4.4 The device **MUST** implement Selectors Level 4 specification (W3C Working Draft, 2 February 2018 or later), in particular:

- Simple selectors:
  - Type selector.
  - Universal selector.
  - Attribute selector.
  - Class selector.
  - ID selector.
  - Pseudo-class selectors, in particular:
    - Negation pseudo-class :not().
    - Action pseudo classes :active(), :focus(), :hover().
    - Tree-structural pseudo-class :empty().
- Compound selectors.
- Complex selectors.
- Combinators: descendant combinator, child combinator, next-sibling combinator, subsequent-sibling combinator.
- Selector lists.

4.5 The target device **MUST** support the following CSS properties:

- animation
- background
- background-color
- background-image
- background-position
- background-repeat
- background-size
- border
- border-left, border-right, border-top, border-bottom
- border-radius
- border-style
- border-width
- box-shadow
- color
- content
- display
- font
- font-family
- font-size
- font-style
- font-weight
- @font-face
- height
- left
- line-height
- margin
- margin-left, margin-right, margin-top, margin-bottom
- max-height
- max-width
- @media
- top
- right
- bottom
- left
- opacity
- overflow
- padding
- padding-left, padding-right, padding-top, padding-bottom
- position
- right
- text-align
- text-decoration
- text-indent
- text-overflow
- text-shadow
- text-transform
- top
- transform
- transform-origin

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function
- vertical-align
- visibility
- white-space
- width
- z-index

4.6 The target device **MUST** support CSS Transitions Level 3.

5.0 HTML Video

5.1 The target device **MUST** be capable of resizing the video layers to arbitrary sizes dynamically and **MAY** be capable of doing so via animation while video and UI are in perfect synchronization and the video playback continues at 30fps or more for at least 75% of the transition time.

Perfect synchronization, in the context of video and UI as mentioned above, means that at no point before, during or after a resizing animation of a video element should any visual artifacts be seen other than the entire, un-cropped video and the full UI it is rendered upon. For example, in cases where synchronization is *not* perfect, there will often be a black border or layer visible that is neither part of the UI nor part of the video.

This information is non-normative.

5.2 The HTML video element **MUST** be able to download media for playback using HTTPS.
5.3 The target device **MUST** support the following event types:

- Event.loadstart
- Event.loadeddata
- Event.loadedmetadata
- Event.play
- Event.playing
- Event.pause
- Event.ended
- Event.suspend
- Event.progress
- Event.seeking
- Event.seeked
- Event.timeupdate
- Event.durationchange
- Event.error
- Event.stalled
- Event.waiting

5.4 The target device **MUST** properly implement the following HTMLMediaElement functions:

- HTMLMediaElement.canPlayType
- HTMLMediaElement.load
- HTMLMediaElement.play
- HTMLMediaElement.pause

5.5 The target device **MUST** properly implement the following HTMLMediaElement properties:

- HTMLMediaElement.volume
- HTMLMediaElement.muted
- HTMLMediaElement.currentTime, including the following configuration requirements:
  - The value of currentTime **MUST** be accurate to within 250 milliseconds during active playback and this is applicable whether the video features a standard frame rate or a high frame rate.
  - The value of currentTime **MUST** be accurate to within 32 milliseconds when content is paused and this is applicable whether the video features a standard frame rate or a high frame rate.
- HTMLMediaElement.duration
- HTMLMediaElement.buffered
- HTMLMediaElement.paused
- HTMLMediaElement.ended
- HTMLMediaElement.autoplay

5.6 The target device **MUST** properly implement the following HTMLMediaElement properties:

- HTMLMediaElement.playbackRate
  - The device **MUST** support the following values for this property: 0.25, 0.50, 1.00, 1.25, 1.50, 2.00.
  - Audio **MAY** be muted when this value is 0.25.

The playbackRate property will change the rate at which the media is played. This property does not indicate a "trick play" mode; all frames should still be displayed and audio should be played back at the same rate. As specified in section 10.1.1 of this document, 60 frames per second is the maximum required decode capability, even when playbackRate > 1.0

This information is non-normative.

5.7 Media Source and Encrypted Media Extensions

5.7.1 The target device **MUST** properly implement Encrypted Media Extensions W3C Candidate Recommendation 18 September 2017, for the com.widevine.alpha key system using OEMCrypto version 14 or greater and including the following configuration requirements:

- Devices implementing Widevine **MUST** support subsample encrypted block format with secure hardware decode (L1) for all WebM VP9 streams.
- The isTypeSupported method **MUST** support the 'cryptoblockformat' additional mime-type parameter, and support the following values:
  - 'subsample', indicating the device is able to process and decrypt a WebM encrypted bitstream where the P bit in the Signal Bytes is set to TRUE.
- The device identification values provided by the target device's chosen key system(s) **MUST** reflect the values used by the device in section 13.1 of this specification.
- Devices implementing Widevine **MUST** implement Widevine Level 1 content protection with secure hardware decode.
- The key system(s) implemented by the target device **MUST** support decoding of all

content resolutions, bitrates, codecs, and container formats supported by the device.

- The device **MUST** enforce HDCP revision 1.1 output protection when required by the Widevine license. If the device supports playback above 1920x1080 resolution, it **MUST** enforce output protection up to and including HDCP revision 2.2 when required by the license.

Container/Codec	Resolution supported by device	Key system requirements
WebM/VP9 Profile 0	1920x1080 and lower	Widevine L1
	Greater than 1920x1080	Widevine L1
WebM/VP9 Profile 2 - 10 bits	1920x1080 and lower	Widevine L1
	Greater than 1920x1080	Widevine L1
MP4/H264, MP4/AVC	1920x1080 and lower	Widevine L1
	Greater than 1920x1080	Not applicable, only VP9 is supported at this resolution
MP4/AAC	N/A	Widevine L3
WebM/Opus	N/A	Widevine L3

This table outlines the required key system robustness levels for the formats used in the YouTube TV application today.

This information is non-normative.

YouTube uses WebM subsample encryption for all protected VP9 Profile 0 and VP9 Profile 2 streams. The design can be found here: http://www.webmproject.org/docs/webm-encryption/

This information is non-normative.

5.7.2 The target device **MUST** properly implement Media Source Extensions W3C Candidate Recommendation 17 November 2016, including the following configuration requirements:

- SourceBuffers **MUST** be capable of holding three media segments.
  - If the target device is capable of playing 1080p content, the device **MUST** allow at least 15 megabytes of data to be appended before beginning to evict time ranges.
  - If the target device is capable of playing 4K content, the device **MUST** allow at least 50 megabytes of data to be appended before beginning to evict time ranges.
  - If the target device is capable of playing 4K HDR content, the device **MUST** allow at least 80 megabytes of data to be appended before beginning to evict time ranges.

- Media playback **MUST** begin if at least one second of media has been appended.
- Media playback **MUST** begin within one second of sufficient data for playback being appended.
- Before end of stream is signaled, media playback **MUST** continue until at most one second of the appended media remains unpresented.
- After end of stream is reached, media playback **MUST** continue until all media is presented.

Example:

If the first 3 seconds of a 10-second video are appended, the device must begin playback within 1 second, and continue playing for at least 2 seconds after that before pausing due to buffer underrun. Where possible, the device should play all 3 seconds before pausing. Once the remaining 7 seconds are appended and end of stream is signaled, the device must resume playback and continue until the end.

This information is non-normative.

The YouTube application will primarily utilize MediaSource for adaptive streaming playback, however, traditional progressive download streams may also occasionally be served and are expected to be supported.
This information is non-normative.

- The isTypeSupported method **MUST** accept and return correct values for the following additional parameters:
    - `width`
        - The x-axis decoded video resolution in pixels.
    - `height`
        - The y-axis decoded video resolution in pixels.
    - `framerate`
        - The decoded video frame rate in frames per second.
    - `bitrate`
        - The encoded video bitrate in bits per second.
    - `channels`
        - The number of absolute audio channels.
    - `cryptoblockformat`
        - The supported encrypted block format.
    - `decode-to-texture`
        - Optional parameter that a device can use to report about its capability to decode a video into a texture that will subsequently be processed by a device's Graphic Processing Unit(s) - GPU - (e.g. to perform spherical or 3D to rectangular projection(s)).
        - The value of the parameter shall either be true of false. When the parameter is set to true, the value of the other parameters in the method are scoped to the capability of the GPU and graphics library. When set to false, the value of all other parameters in the method reference the capability of the conventional rectangular display unit.
        - The method shall trivially return false whenever the parameter value is neither equal to "true" or "false" (i.e. "=invalid", "=not supported", etc…)
        - When the parameter is not present, its value shall be assumed to be

equal to false.

- If the target device supports HDR as defined in section 16.0 of this document, it **MUST** also properly implement the following additional parameter:
    - `eotf`
        - The supported electro-optic transfer function, as defined in Section 16.3 of this document.

Example 1:

```
MediaSource.isTypeSupported('audio/mp4; codecs="mp4a.40.2"; channels=6')
```

Example 2:

```
MediaSource.isTypeSupported('video/webm; codecs="vp9"; width=1920; height=1080;
framerate=60; bitrate=4000000')
```

Example 3, for querying if a device supports HDR:

```
MediaSource.isTypeSupported('video/webm; codecs="vp9.2"; width=3840; height=2160;
framerate=60; bitrate=25000000; eotf=smpte2084')
```

Refer to section 16.0 of this document for more information on HDR.

Example 4, for querying if a device supports the WebM Subsample Encrypted Block Format:

```
MediaSource.isTypeSupported('video/webm; codecs="vp9.2";
cryptoblockformat=subsample; width=3840; height=2160; framerate=60;
bitrate=25000000; eotf=smpte2084')
```

Example 5, for querying if a device supports decode to texture needed to render 360 degree videos to be displayed on an 1080P HD resolution TV set. In this example, the GPU in the TV set supports 2K resolution content:

```
MediaSource.isTypeSupported('video/webm; codecs="vp9"; width=2560; height=1440;
framerate=60; bitrate=17000000; decode-to-texture=true’)
```

This information is non-normative.

- The values reported by the isTypeSupported method **MUST** represent the maximum capability that the combined video decoding, rendering and display system can deliver.

5.7.3 The target device **MUST** properly implement the Media Playback Quality APIs define at https://wicg.github.io/media-playback-quality/ with 99% accuracy over any 5 second period of the playback

5.8 Runtime

5.8.1 The target device **MAY** use the Cobalt runtime as the underlying Browser engine for the YouTube application. If the Cobalt runtime is used in the device,

a. The target device **MUST** run a version that is not older than the Cobalt 2019 LTS release (LTS = Long Term Support).
b. The target device **MUST** properly implement all Starboard APIs defined in the Cobalt version that has been selected to run on the device
c. The target device **MUST** pass all items of the following test suites:
    i. All Cobalt tests - including Starboard NPLB Tests (see src/starboard/nplb/).
d. The target device **MUST** remain in compliance with subsequent Long-Term-Support (LTS) versions of Cobalt published by YouTube for a period of three consecutive calendar years after the target device was first made available to users

6.0 Network Stack

6.1 Protocols

6.1.1 The target device **MUST** support XMLHttpRequest Living Standard (12 January 2018 or later).

6.1.2 The target device **MUST** support the Fetch API and the Streams API, with ReadableByteStream support for a Fetch API Response.

6.1.3 The target device **MUST** support TLS version 1.2

6.1.4 The target device **MUST** send a Server Name Indication extension containing the target domain name.

6.1.5 The target device **MUST** support TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 with P-256 and uncompressed points.

6.1.6 The target device **MUST** support certificates with Subject Alternative Names and wildcard characters as the left-most label.

For requirements 6.1.3 through 6.1.6, see https://cloudplatform.googleblog.com/2017/06/Google-Cloud-services-are-switching-Certificate-Authority.html for additional information

This information is non-normative.

6.2 The target device **MUST** support Cross-Origin Resource Sharing.

6.3 Root Certificates

6.3.1 The target device **MUST** support HTTPS/SSL and include the roots.pem certificate file available at http://pki.google.com/roots.pem, and have the ability to update trusted Roots.

- Note: the YouTube service will have transitioned to "GlobalSign Root CA - R2" before end of 2018

Depending on business needs and technology changes, we may regularly change our intermediate certificate authority used for signing SSL server certificates. The current intermediate certificate authority or root certificate authority should never be relied upon as static. Please see https://pki.goog/faq.html for more information

Google & YouTube's production certificates will transition from being rooted at GeoTrust & Equifax, to using the GlobalSign R2 root in early 2017.

This information is non-normative.

6.3.2 The target device **MUST** support the set of Mozilla certificates as produced by the utility posted at https://github.com/agl/extract-nss-root-certs

6.4 The device **MUST** be able to download multiple HTML, JavaScript, CSS, and image assets concurrently.
6.5 The connection overhead (that is excluding network latency) **MUST** be less than 500 milliseconds.

The YouTube application makes a non-trivial number of HTTP(S) requests to fetch YouTube application content, scripts, and data. Resources are typically retrieved using the native browser HTTP stack or via XMLHttpRequest.

This information is non-normative.

6.6 IP and DNS

6.6.1 The target device **MUST** support IPv4 and ensure that AAAA DNS records are safely ignored if IPv6 is not supported.

6.7 Cookies and Storage

6.7.1 The target device **MUST** provide a persistent cookie jar adhering to RFC 2965. Persistence here means that the cookie **MUST** remain intact after at least 200 repeated device turn on/off or repeated YouTube application launch/exit spread uniformly in time over a 24 Hour period.
6.7.1 The target device **MUST** provide a persistent cookie jar adhering to RFC 2965.
6.7.2 The target device **MUST** provide a Web Storage container with at least 1 million characters of quota, adhering to the same-origin policy.
6.7.3 The following methods and properties **MUST** be available:

- window.localstorage.setItem
- window.localstorage.getItem
- window.localstorage.removeItem

6.7.4 Cookies and Web Storage data **MUST** be stored locally on the device.
6.7.5 Cookies and other locally stored data **MUST** be persisted unless manually cleared by a user. This includes, but is not limited to preserving data between YouTube application sessions, during firmware or similar software updates, and after hard or soft power cycles.

6.8 The device **MUST** be able to handle URLs with a length of up to 2,048 characters.
6.9 The device **MUST** support the use of the HTTP protocol for communication with local network addresses.
6.10 The device **MUST** support Content Security Policy Level 2 version 21 July 2015.

7.0 Rendering

7.1 The target device **MUST** be capable of rendering and compositing all YouTube application elements correctly.
7.2 The target device **MUST** display the YouTube application in full screen, without browser chrome or other system elements.
7.3 The target device **MUST** be capable of rendering the YouTube application interface at one of the following

supported resolutions:

Device is <= FHD		Device is > FHD	
Video Resolution (Horizontal x Vertical dimension)	Minimum Render and Display resolution (Horizontal x Vertical dimension)	Video Resolution (Horizontal x Vertical dimension)	Minimum Render and Display resolution (Horizontal x Vertical dimension)
Up to 1280x720	At least 1280x720	Up to 1920x1080	At least 1280x720
		Greater than 1920x1080 and Up to 2560x1440	At least 1920x1080
		Greater than 2560x1440 and Up to 3840x2160	At least 1920x1080

Note: The *≤FHD* and *>FHD* categories are defined in the companion YouTube Hardware Requirements for CE Devices specification. These categories are defined based on the maximum video resolution the target device is capable of outputting to an external or built-in screen:

- *≤FHD*: Any device capable of outputting 2,073,600 or fewer pixels simultaneously.
- *>FHD*: Any device capable of outputting more than 2,073,600 pixels simultaneously

This information is non-normative.

7.4 If target device is set to operate in HDR mode, it **MUST** be capable of rendering the YouTube application interface correctly, independent from the color primaries and transfer function of the HDR video being played. More specifically, any SDR UI element needs to be mapped and rendered into an HDR surface that is compatible with the HDR display mode selected on the device.

7.5 The target device **MUST** implement the following property:

- window.devicePixelRatio, including the following configuration requirements:
  - If the target device's video decode capability is less than or equal to 1920x1080, the value of this property **MUST** be one of the following:
    - 1
    - 1.5
  - If the target device's video decode capability is greater than 1920x1080, the value of this property **MUST** be one of the following:
    - 1
    - 1.5
    - 2

By default, the resolution at which the YouTube application is rendered will set an upper bound for the maximum content resolution selected by the YouTube application's adaptive bit rate algorithm, regardless of whether or not the device uses separate compositing paths for the YouTube application and video image layers. It is the device's responsibility to ensure that video is scaled properly. The device's `MediaSource.isTypeSupported()` extension's `height` and `width` parameters will also set an upper bound for the maximum content resolution selected.

This information is non-normative.

7.6 The browser's spatial navigation and element highlighting **MUST** be disabled when rendering the application UI.
7.7 The target device **MUST** provide the ability to overlay/composite HTML content on top of the video surface.
7.8 If the target device supports 3D output, it **MUST** be capable of displaying 2D UI elements on top of 3D video content without distortion or other visual artifacting.

8.0 Images

8.1 The target device **MUST** be able to render the following image formats:

- JPEG
- WebP
  - Devices **MUST** fully implement the WebP specification, including the support for the following features:
    - Animation, with a framerate of 10 frames per second
    - Both lossy and lossless compression up to 1280x720 resolution
    - Metadata
    - Transparency
    - Color Profile

9.0 Remote Keys and Key Events

9.1 Device remote keys not implemented by the YouTube application or reserved for device system functions **MUST NOT** dispatch any key event or key code.
9.2 Devices implementing a system-wide "exit" key **MUST** use that key to gracefully exit the YouTube application.
9.3 The device **MUST** support six-point navigation at a minimum, which includes the following keys:

- Left (keyCode = 0x25)
- Right (keyCode = 0x27)
- Up (keyCode = 0x26)
- Down (keyCode = 0x28)
- Enter (keyCode = 0x0D)
- Escape / Back / Return (keyCode = 0x1B)

9.4.1 The device **MUST** support the following keys if the keys exist on the device remote:

- Play (keyCode = 0xFA)
- Pause (keyCode = 0x13)
- Play/Pause (keyCode = 0xB3)
  - Only for remotes that implement play and pause functionality on a single button.
- Stop (keyCode = 0xB2)
- Fast Forward (keyCode = 0xE4)
- Rewind (keyCode = 0xE3)

- Space (keyCode = 0x20)
- Backspace (keyCode = 0x08)\
- Delete (keyCode = 0x2E)
- Search (keyCode = 0xAA)
- Microphone / Voice (keyCode = 0x3002)
- Previous (keyCode = 0xB1)
- Next (keyCode = 0xB0)
- Closed Captions / Subtitle (keyCode = 0x1CC)
- Red (keyCode = 0x193)
- Green (keyCode = 0x194)
- Yellow (keyCode = 0x195)
- Blue (keyCode = 0x196)
- YouTube button (keyCode = 0x3000)
    - Only for remotes that implement a dedicated YouTube logo button. This keycode need only be sent while the application is in use.
    - See Section 13.3 below (Request Parameters) for the request to send whether the YouTube application is already running or not
    - This key must be sent when the user has pressed the launch button associated with the YouTube app on a remote. See Terminology section at the top of this specification for the possible YouTube apps.

9.4.2 The device **MAY** support the following keys if the keys exist on the device remote , and are not reserved for device or system functions (see §9.1):

- Channel Up (keyCode = 0x1AB)
- Channel Down (keyCode = 0x1AC)
- Last / Previous / Recall Channel (keyCode = 0x25F)
- Audio Track Selection (keyCode = 0x3001)
- Info / Display (keyCode = 0x1C9)
    - This key displays information about channel/program or input you are watching.
- Guide / EPG (keyCode = 0x1CA)

As remote keys are not standard, there may be confusion when mapping a device's remote keys to the functions specified in this document. If there is any uncertainty, please reach out to your partner manager.
This information is non-normative.

9.5 The device **MUST** dispatch the following key events, as appropriate:

- Window.keydown
    - After a key is held down for 500ms, the Window.keydown event **MUST** repeat every 50ms until a user stops holding the key down.
- Window.keyup

9.6 The device **MAY** dispatch the following key events, as appropriate:

- Window.keypress

10.0 Media Playback

10.1.1 The target device **MUST** be able to correctly parse and display video content at arbitrary aspect ratios, variable bit rates, and variable frame rates up to a maximum of 60 frames per second for all supported resolutions.

10.1.2 The target device **MUST** be capable of decoding and displaying video content at all output resolutions supported by the device.

YouTube does not restrict or control the aspect ratios of available content. While 16:9, 21:9, and 4:3 aspect ratios are most commonly used, content may be in any aspect ratio depending on the uploaded source file.

This information is non-normative.

10.2 The target device **MUST** be able to maintain audio sync with video content at all times. More specifically, audio rendering MUST remain within a [-125ms , 45ms] range of the corresponding rendered video frame.
10.3 If the target device supports 3D output, it **MUST** be able to detect and display 3D content automatically.
10.4 The target device **MUST** properly implement the following W3C Web Audio API elements:

- AudioNode
  - AudioNode.context
  - AudioNode.numberOfInputs
  - AudioNode.numberOfOutputs
  - AudioNode.channelCountMode
  - AudioNode.channelInterpretation
  - AudioNode.connect
  - AudioNode.disconnect
- AudioBufferSourceNode
  - AudioBufferSourceNode.buffer
  - AudioBufferSourceNode.onended
  - AudioBufferSourceNode.start
  - AudioBufferSourceNode.stop
- AudioDestinationNode
  - AudioDestinationNode.maxChannelCount
    - This value **MUST** correctly communicates the number of available output channels.
- AudioContext
  - AudioContext.destination
  - AudioContext.sampleRate
  - AudioContext.currentTime
  - AudioContext.decodeAudioData, with the following configuration requirements:
    - This method **MUST** accept MP3 or WAV data.
  - AudioContext.createBufferSource
- The target device **MAY** support the following methods:
  - AudioBufferSourceNode.playbackRate
  - AudioContext.createMediaElementSource
  - AudioContext.createGain
  - AudioContext.createChannelSplitter
  - AudioContext.createChannelMerger
- The target device **MAY** support the following elements:

- MediaElementAudioSourceNode
- GainNode
- ChannelSplitterNode
- ChannelMergerNode
- BiquadFilterNode
- IIRFilterNode

WebAudio is a high-level JavaScript API for processing and synthesizing audio in web applications, such as navigational sound effects played in response to user actions, or spatial audio.

This information is non-normative.

10.5.1 The target device **MUST** implement one of the following hardware-accelerated rendering specifications for spherical video support:

- WebGL Version 1.0.3, 27 October 2014
- CSS Spherical Filter Extension

CSS Spherical Filter Extension is an alternative to WebGL for spherical video playback. CSS Spherical Filter Extension is the only supported shader in Cobalt.

This information is non-normative.

10.5.2 The target device's GPU **MUST** be capable of decoding video to texture at all resolutions and frame rates supported by the device's video path.

10.5.3 If the target device supports WebGL, it **MUST** support the CANVAS HTML element, and properly implement all specified typedefs, dictionaries, Interfaces, and the following subset of WebGLRenderingContext properties:

- WebGLRenderingContext
  - WebGLRenderingContext.canvas
  - WebGLRenderingContext.drawingBufferWidth
  - WebGLRenderingContext.drawingBufferHeight
  - WebGLRenderingContext.isContextLost
  - WebGLRenderingContext.activeTexture
  - WebGLRenderingContext.attachShader
  - WebGLRenderingContext.bindAttribLocation
  - WebGLRenderingContext.bindBuffer
  - WebGLRenderingContext.bindFramebuffer
  - WebGLRenderingContext.bindRenderbuffer
  - WebGLRenderingContext.bindTexture
  - WebGLRenderingContext.bufferData
  - WebGLRenderingContext.clear
  - WebGLRenderingContext.clearColor
  - WebGLRenderingContext.compileShader
  - WebGLRenderingContext.createBuffer
  - WebGLRenderingContext.createFramebuffer

- WebGLRenderingContext.createProgram
- WebGLRenderingContext.createRenderbuffer
- WebGLRenderingContext.createShader
- WebGLRenderingContext.createTexture
- WebGLRenderingContext.deleteBuffer
- WebGLRenderingContext.deleteFramebuffer
- WebGLRenderingContext.deleteProgram
- WebGLRenderingContext.deleteRenderbuffer
- WebGLRenderingContext.deleteShader
- WebGLRenderingContext.deleteTexture
- WebGLRenderingContext.detachShader
- WebGLRenderingContext.disable
- WebGLRenderingContext.disableVertexAttribArray
- WebGLRenderingContext.drawArrays
- WebGLRenderingContext.drawElements
- WebGLRenderingContext.enable
- WebGLRenderingContext.enableVertexAttribArray
- WebGLRenderingContext.framebufferRenderbuffer
- WebGLRenderingContext.framebufferTexture2D
- WebGLRenderingContext.getAttribLocation
- WebGLRenderingContext.getError
- WebGLRenderingContext.getProgramParameter
- WebGLRenderingContext.getProgramInfoLog
- WebGLRenderingContext.getRenderbufferParameter
- WebGLRenderingContext.getShaderParameter
- WebGLRenderingContext.getShaderPrecisionFormat
- WebGLRenderingContext.getShaderInfoLog
- WebGLRenderingContext.getUniform
- WebGLRenderingContext.getUniformLocation
- WebGLRenderingContext.getVertexAttrib
- WebGLRenderingContext.linkProgram
- WebGLRenderingContext.readPixels
- WebGLRenderingContext.shaderSource
- WebGLRenderingContext.texImage2D
- WebGLRenderingContext.texImage2D
    - **10.5.3.1** WebGLRenderingContext.texImage2D **MUST** support VideoElement and Image as a source.
- WebGLRenderingContext.texParameterf
- WebGLRenderingContext.texParameteri
- WebGLRenderingContext.texSubImage2D
- WebGLRenderingContext.uniform1f
- WebGLRenderingContext.uniform1fv
- WebGLRenderingContext.uniform1fv
- WebGLRenderingContext.uniform1i
- WebGLRenderingContext.uniform1iv
- WebGLRenderingContext.uniform2f
- WebGLRenderingContext.uniform2fv
- WebGLRenderingContext.uniform2i
- WebGLRenderingContext.uniform2iv

- WebGLRenderingContext.uniform3f
- WebGLRenderingContext.uniform3fv
- WebGLRenderingContext.uniform3i
- WebGLRenderingContext.uniform3iv
- WebGLRenderingContext.uniform4f
- WebGLRenderingContext.uniform4fv
- WebGLRenderingContext.uniform4i
- WebGLRenderingContext.uniformMatrix2fv
- WebGLRenderingContext.uniformMatrix3fv
- WebGLRenderingContext.uniformMatrix4fv
- WebGLRenderingContext.useProgram
- WebGLRenderingContext.validateProgram
- WebGLRenderingContext.vertexAttrib1f
- WebGLRenderingContext.vertexAttrib1fv
- WebGLRenderingContext.vertexAttrib2f
- WebGLRenderingContext.vertexAttrib2fv
- WebGLRenderingContext.vertexAttrib3f
- WebGLRenderingContext.vertexAttrib3fv
- WebGLRenderingContext.vertexAttrib4f
- WebGLRenderingContext.vertexAttrib4fv
- WebGLRenderingContext.vertexAttribPointer
- WebGLRenderingContext.viewport
- WebGLRenderingContext.viewPortWidth
- WebGLRenderingContext.viewPortHeight

WebGL is a JavaScript API which allows rendering of 3D graphics and 2D graphics within the browser, and can be used to implement interactive features such as 360 degree video.

The explicitly listed subset of WebGLRenderingContext properties are those that are required for YouTube's usage.

This information is non-normative.

11.0 DIAL

11.1 The device **MUST** implement the DIAL 2.1 protocol and be discoverable by default. In particular, the target device **MUST** be capable of properly parsing arbitrary and multiple parameter values posted to the DIAL server's additionalDataUrl structure.
11.2 The device **MUST** implement DIAL Wake-up functionality as described in Section 7 of the DIAL 2.1 protocol specification.

11.3 Wake-up loading time

11.3.1 The target device **MUST** be able to load the application in 35 seconds, measuring from the time the user initiates Wake-up to the time the video content begins playback.

11.4 The target device **MUST** implement the out-of-box experience requirements as described in Section 9 of the DIAL 2.1 protocol.
11.5 The target device **MUST** have a default configuration enabling discovery of its DIAL server by other devices.

11.6 The target device **MUST** be discoverable via DIAL (i.e. the DIAL server is active) by the time the target device is in its active and ready-to-use state.
11.7 The target device **MAY** provide the user with an option to disable the DIAL server on the device.

12.0 Loading and Unloading the Application

12.1 The YouTube application **MUST** be loaded via the following URL, with any URL parameters and fragments appended as required:

https://www.youtube.com/tv

12.2 When the YouTube application exits, all states **MUST** be reset, except for persistent local storage data and cookies.

13.0 Application Identification & Configuration

When the application starts, the target device **MUST** load or create the following:

13.1 Client ID (User-Agent String)

13.1.1 The User-Agent string **MUST** uniquely identify the brand, model, and Internet connection type for the given device. The following fields **MUST** be present within the User-Agent string:

- Browser_Name
- Browser_Version
- Device_Name
    - This **MUST** be a fully-concatenated and underscore-delimited string containing the network operator, device type, chipset model number, and device model year. The Device_Name format is therefore *<Network Operator>_<Device Type>_<Chipset Model Number>_<Model Year>* and the 5 device attributes in the Device_Name string are defined as follows:
        - The network operator field **MUST** contain the name of the network operator or the commercial name of the service that owns the target device. If the field is not applicable to the target device, the field **MUST** still be present and **MUST** be left blank.
        - The device type field **MUST** contain one of the following values:
            - `BDP`
            - `GAME`
            - `OTT`
            - `STB`
            - `TV`
        - The chipset model number field **MUST** contain the full model number of the main platform chipset, including any vendor-specific prefixes.

An example of chipset model number is “BCM7251” which is a Broadcom chipset (See https://www.broadcom.com/products/broadband/set-top-box/bcm7251)
This information is non-normative.

- The model year field **MUST** contain the year the device was initially certified. In particular, the value of the model year is “2019” if it implements the full set of 2019 YouTube Software and Hardware Requirements for CE Devices requirements specified herein

- Firmware_Version
    - This **MUST** be the production firmware version number which the device is currently running.
- Brand
    - This **MUST** be the name of the brand under which the device is being sold.
- Model
    - This **MUST** be the final production model number of the device.
- Connection_Type
    - This describes the type of network connection used by the device at application launch, and **MUST** be either `Wired` or `Wireless`.

13.1.2 The device **MUST** provide the above User-Agent fields in one of the two following formats:

1. YouTube format:

   Browser_Name/Browser_Version, Device_Name/Firmware_Version (Brand, Model, Connection_Type)

Example YouTube User-Agent string:

Opera/9.80 (Linux armv7l ; U; en), Presto/2.8.115 Version/11.10, GFiber_STB_GC4155/2.22 (Google, GGL36XX200, Wireless)

The ClientID string is used by YouTube to target individual combinations of software and hardware platforms:

```
Browser_Name/Browser_Version, Device_Name/Firmware_version (Brand, Model, Connection_Type)
       +                +           +                +        +     +               +
       |                |           |                |        |     |               |
       v                |           v                |        v     +------------+  |
Coarse Browser Matching |   Coarse Platform Matching |  Coarse Hardware Matching |  |
                        v                            v                           v  |
          Fine Browser Matching     Fine Platform Matching    Fine Hardware Matching |
                                                                                    |
                                                                 Network Profiling<-+
```

This information is non-normative.

2. HbbTV format:
   YouTube supports the HBB User-Agent string as defined in Section 7.3.2.4 of the HbbTV specification. The Browser_Name and Browser_Version parameters **MUST** be included in addition to the HbbTV defined User-Agent.

Example HbbTV User-Agent String:

<Browser_Name> <Browser_Version> HbbTV/1.1.1 (<capabilities>; [<Brand>]; [<Model>]; [<firmware_version>]; [<Device_Name>]; <reserved>)

This information is non-normative.

13.2 Language and Territory

13.2.1 Language and Territory **MUST** be set through the use of the Accept-Language request-header.

Refer to 2-character ISO codes for language and country codes found in Language and Country Code Combinations and Validation. Refer to RFC7231, Section 5.3.5 for more information on the Accept-Language request-header, including language preference.

Example Accept-Language request-header:

```
        da, en-gb;q=0.8, en;q=0.7
```

This information is non-normative.

13.2.2 The language and region sent through the Accept-Language request-header **MUST** match the language setting on the device and **MUST** change when the user changes the device language.

13.3 Specific request parameters

Certain requests require additional query parameters to be present in the URL as name value pairs:

- `launch` - The method by which a user has launched the YouTube application. The value:
  - **MUST** be `preload` if the device has automatically opened the YouTube application without user input. This parameter **MUST** be present for the duration of the preloaded process's life.
    - For non-Cobalt devices, additional `launch` parameters **MUST** be used after the YouTube application state management fragment as appropriate. See section 13.4.1 of this document for further details on combining this parameter with other `launch` parameters.
  - **MUST** be `menu` if the YouTube logo is selected from the device's menu screen.
  - **MUST** be `guide` if a YouTube video is selected from a device's guide or EPG pages.
  - **MUST** be `launcher` if a YouTube video is selected from a device's content launcher.
  - **MUST** be `remote` if a dedicated YouTube button on a device remote has been pressed.
  - **MUST** be `search` if a YouTube video is selected from a device's universal search.
  - **MUST** be `voice` if the YouTube application was directly opened as a result of a voice command.

Example 1, for a scenario when YouTube is launched by the user via a YouTube remote button:
https://www.youtube.com/tv?launch=remote

Example 2, for a scenario when YouTube is preloaded and is first launched by the user via selecting a universal search result:
https://www.youtube.com/tv?launch=preload#play?launch=search&v=1vt2p5ZVZpM

Example 3, for a scenario when YouTube is immediately launched by the user via saying a voice command similar to "Launch YouTube":
https://www.youtube.com/tv?launch=voice

Example 4, for a scenario when YouTube is immediately launched by the user via saying a voice command similar to "Search for music videos on YouTube":
https://www.youtube.com/tv?launch=voice&q=music+videos

This information is non-normative.

- `v` - A YouTube video ID. This parameter **MAY** be used to launch the YouTube application with a specific video.

Example: https://www.youtube.com/tv?v=9bZkp7q19f0
This information is non-normative.

- `list` - A YouTube playlist ID. This parameter **MAY** be used to launch the YouTube application with a specific playlist.

Example: https://www.youtube.com/tv?list=PLOU2XLYxmsIJOOTFfYzhR2d-rcSbBbEE_
This information is non-normative.

- `t` - The time at which to start a video. This parameter **MAY** be used to launch the YouTube application to a specific time index within a video. The value **MUST** follow the syntax of `XmYs`, where X is the minutes and Y is the seconds from the start of the video. This parameter **MUST** be accompanied by a YouTube video ID parameter.

Example: https://www.youtube.com/tv?v=9bZkp7q19f0&t=1m20s
This information is non-normative.

- `q` - A search query. This parameter **MAY** be used to launch the YouTube application with a specific search query. Words in the query string **MUST** be separated by a '+' sign. The query string **MUST** be UTF-8 URL encoded.

Example: https://www.youtube.com/tv?q=Gangnam+Style
This information is non-normative.

- `c` - A YouTube Channel ID. This parameter **MAY** be used to launch the YouTube application directly to a specific YouTube channel.

Example: https://www.youtube.com/tv?c=UC_x5XG1OV2P6uZZ5FSM9Ttw
This information is non-normative.

13.4 YouTube Application State Management

13.4.1 Non-Cobalt Browsers

13.4.1.1 Devices setting the `launch=preload` parameter as described in section 12.3 of this specification **MUST** use the following fragment identifier when initially opening the YouTube application:

- `#start` - Signals the YouTube application to reset to a state as if it was started for the first time. This fragment **MUST** also be paired with the appropriate launch parameter defined in 12.3.

Example:
The device has opened the browser and loaded the YouTube application in the background, and it is not visible to the user. When the user selects the YouTube application launcher icon for the first time in a session, the `#start?launch=menu` fragment would be appended to the URL, indicating that this is the first time the application is visible to the user, and the YouTube application was launched using the menu icon:

https://www.youtube.com/tv?launch=preload**#start**?launch=menu

This information is non-normative.

13.4.1.2 Devices keeping the YouTube application process alive, or in a suspended state upon exit **MUST** use the following fragment identifier when closing the YouTube application:

- `#suspend` - Signals the YouTube application to cease operations and unload elements to reduce system resource usage.

Example 1:
Devices may choose not to preload the YouTube application, but still wish to keep it active in memory once launched. This behavior is encouraged and will make subsequent YouTube application launches faster than killing the process upon each exit. In this case, the `?preload` parameter would not be used, but the `#suspend` fragment would be appended on exit:

https://www.youtube.com/tv**#suspend**

Example 2:
If the YouTube application was initially preloaded on the device, then the `?preload` parameter would be present and the `#suspend` fragment would be used:

https://www.youtube.com/tv?launch=preload**#suspend**

This information is non-normative.

13.4.1.3 Devices keeping the YouTube application process alive, or in a suspended state upon exit **MUST** use the following fragment identifier when re-opening the YouTube application:

- `#resume` - Signals the YouTube application to return to the last session state it was in before being suspended.

Example 1:
Devices may choose not to preload the YouTube application, but still wish to keep it active in memory once launched. This behavior is encouraged and will make subsequent YouTube application launches faster than killing the process upon each exit. In this case, the `?preload` parameter cannot be used, and the `#resume` fragment MUST be appended to reopen the YouTube application.
This example shows the YouTube application being resumed from a `#suspend` state using the menu/launcher icon:

https://www.youtube.com/tv**#resume**?launch=menu

Example 2:
If the YouTube application was initially preloaded on the device, then the `?preload` parameter MUST be present and the `#resume` fragment MUST be used.

This example shows a preloaded YouTube application being resumed from a `#suspend` state using the menu/launcher icon:

https://www.youtube.com/tv?launch=preload**#resume**?launch=menu

Example 3:
If the YouTube application was initially preloaded on the device, then the `?preload` parameter MUST be present and the `#resume` fragment MUST be used.

This example shows a preloaded YouTube application being resumed from a `#suspend` state using a platform-level voice search for "katy perry":

https://www.youtube.com/tv?launch=preload**#resume**?launch=voice&q=katy%20perry

This information is non-normative.

13.4.1.4 Devices keeping the YouTube application process alive, or in a suspended state upon exit **MUST** use the following fragment identifier when re-opening the YouTube application directly to a video playback:

- `#play` - Signals the YouTube application to initiate playback of a video. This fragment **MUST** be combined with the v and, optionally, the `t` parameters to initiate playback of a specific video at a specific time.

Example 1:
A device may need to reopen the YouTube application from a suspended state, and initiate playback of a specific video deeplink. To do so, the `#play` fragment will be appended to the original launch URL, in addition to the `v=` parameter to indicate the video to play and the `launch=` parameter to indicate the method by which the user is resuming the YouTube application.
This example shows a non-preloaded YouTube application playing a specific video selected from a content launcher while the YouTube application was in a `#suspend` state:

https://www.youtube.com/tv**#play**?v=9bZkp7q19f0&launch=launcher

Example 2:
A device may need to reopen the YouTube application from a suspended state, and initiate playback of a specific video deeplink. To do so, the `#play` fragment will be appended to the original launch URL, in addition to the `v=` parameter to indicate the video to play and the `launch=` parameter to indicate the method by which the user is resuming the YouTube application.

This example shows a preloaded YouTube application playing a specific video as a result of a voice command issued while the YouTube application was in a `#suspend` state:

https://www.youtube.com/tv?launch=preload**#play**?v=9bZkp7q19f0&t=1m20s&launch=voice

This information is non-normative.

13.4.2 Cobalt Browsers

13.4.2.1 Cobalt-based browsers must properly implement the Starboard Lifecycle API, as described here. The lifecycle state will be queried by the YouTube application via the Page Visibility State API and the document's focus/blur state.

14.0 Performance

YouTube reserves the right to review the performance of the application running on the target device, including audio and video quality, load times, navigational responsiveness, and system stability as part of its certification process. If a device is not capable of providing an acceptable user experience due to performance issues, YouTube may decline to certify the device.

This information is non-normative.

14.1 Multiple App Environment

14.1.1 The target device **MUST** be able to maintain YouTube application responsiveness when device system functions are used over or alongside the YouTube application.

14.1.2 The performance of the target device **MUST NOT** degrade in any way upon resuming or re-launching the YouTube App after running other applications during a user session.

A typical user interaction with a CE device may involve launching and using multiple applications, switching back and forth between them, and switching the device into a suspended mode. These actions should not negatively affect the user experience, or the availability of system resources on the device.

This information is non-normative.

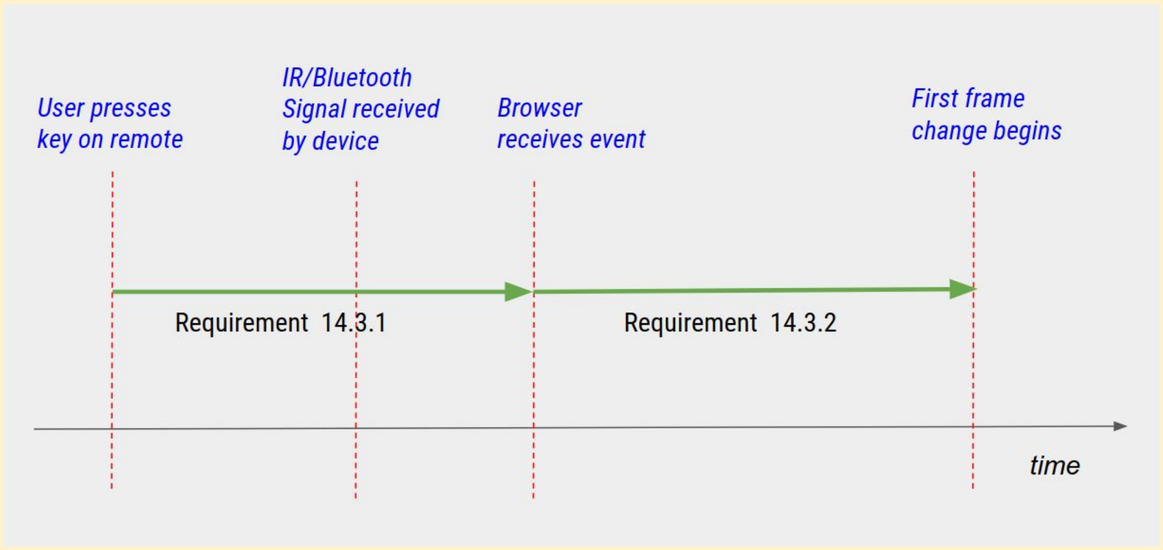
14.2 Loading time

14.2.1 The target device **MUST** be able to load the YouTube application in 10 seconds, this value measuring the 95th percentile value from the time instant the user selects the application for launch to the time instant when the browser finishes rendering the application and factoring out network transfer times.

14.3 Latency on User input

14.3.1 The Remote latency (the time between a user initiating an event and the Browser receiving the event) **MUST NOT** exceed 150 milliseconds.
14.3.2 The Event Handling and Initial Animation latency **MUST NOT** exceed 200 milliseconds, this value measuring the median values for the time elapsed between the Browser receiving an event and the Browser completing the rendering of the first frame

See below for a diagram illustrating the latency on User Input requirements above.


User presses key on remote
IR/Bluetooth Signal received by device
Browser receives event
First frame change begins
Requirement 14.3.1
Requirement 14.3.2
time

This information is non-normative.

14.4 Transition times

14.4.1 The target device **MUST** be able to complete a browse-to-watch transition within 2.5 seconds, this value measuring the median values for the time elapsed between the time the Browser receives the event to the time the video content begins playback and factoring out network transfer times.

14.5 Frame rate

14.5.1 The target device **MUST** be able to maintain a minimum frame rate of 24 frames per second at all times.

14.6 Video endurance

14.6.1 The target device **MUST** be able to maintain video playback continuously for 12 hours without network disconnections or interruptions on all network interfaces, including the case where the network is a Wi-Fi® wireless network

14.7 Overall endurance

14.7.1 The target device **MUST** be able to endure 12 hours of key input stress test.

15.0 Speech

If the target device includes a near field, far field, or other form of microphone input it **MUST** consider this section:

15.1 The target device **MUST** support `MediaDevices.enumerateDevices()` as outlined in the Media Capture and Streams W3C Candidate Recommendation 3 October 2017. This method collects information about the media input and output devices available on the target device.

> **15.1.1** If speech functionality is possible using an available connected device, `enumerateDevices()` **MUST** return at least one media device with `kind` attribute "`audioinput`."
> **15.1.2** The device **MUST NOT** create persistent data storage for any of the media device with `kind` attribute "`audioinput`." that `enumerateDevices()` returns.
> **15.1.3** Each audio input device returned via the API **MUST** support speech functionality *without the need for additional hardware*, otherwise the API **MUST** not include that particular device in the enumerated list of media devices.
> **15.1.4** If there are no connected audio input devices, or all connected devices require additional software in order for speech functionality to be enabled, `enumerateDevices()` **MUST** return an empty list.
> **15.1.5** The `devicechange` event **MAY** be triggered when the set of media devices returned by `enumerateDevices()` changes.

Example: When target device is connected to two microphones fully capable of voice search without any additional hardware,

```
navigator.mediaDevices.enumerateDevices()
.then(function(devices) {
  devices.forEach(function(device) {
    console.log(device.kind);
  });
})
```

would return

```
audioinput
audioinput
```

This information is non-normative.

15.2 The target device **MUST** properly implement the Web Speech API Specification, 19 October 2012.

Web Speech API is a high-level JavaScript API to enable web developers to incorporate speech recognition and synthesis into their applications. The Web Speech API can be used for voice driven YouTube application features, such as search.

This information is non-normative.

15.3 When the YouTube application is running, the target device **MUST NOT** create any persistent local storage of the data input to or output from the Web Speech APIs.

15.4 Expected speech behaviors

15.4.1 From anywhere on the platform, if the user's voice query involved the use of a "condition" (e.g "...on YouTube", "...with YouTube Kids", etc.), it **MUST** be resolved by the specified service by directly launching the app with the appropriate deeplink. Minimally, any self-contained voice query including "YouTube" **MUST** be passed on to the YouTube application on the device; and "YouTube Kids" or "YouTube TV" to the appropriate application (if present on the platform).

A "condition" being defined as any manner of trigger words or phrases issued by the user to designate their intent to scope their query to a given service, e.g. "Open YouTube", "Buscar videos de yoga en YouTube", "Play trap music with YouTube", "Search YouTube for late night comedy".

This information is non-normative.

15.4.2 Otherwise if the query was a non-media request, the device assistant **MAY** handle they query as appropriate to the platform.

A query is defined as "non-media request" when the user is requesting something other than a video / audio result and it is something the device is able to handle directly without involving any content services, e.g. "Turn on the lamp" or "What's the price of TSLA stock?".

This information is non-normative.

15.4.3 Otherwise if the query was a media request, and the user is…

- **15.4.3.a** …in-app...: it **MUST** be directly handed off to the app's search backend to resolve and display results in the app's own UI *unless* the query has an "exclusive" match to some other content service in which case the device assistant **MAY** deep link into the respective service's result(s) page.
- **15.4.3.b** …not in-app...: the device assistant **MAY** determine if there is any service with "exclusive" match and if so **MAY** deeplink the user into the respective service's result(s) page. If, however, there is no exclusive match for the user's query, the device assistant **MUST** either:
  - request results from all services and display them as shelves the platform's "universal search" page; shelves **MUST** be ordered according to user preference / frequency of use
  - or allow for user configurable preferences for various content categories (i.e. shows, movies, music, sports, etc.) and deep link directly to the user's preferred service

A query is defined as "media request" when the user's request has video / audio intent or is anything other than something the device is able to handle as a non-media request, e.g. "Saturday Night Live", or "Comedies", or "Slime Tutorials".

This information is non-normative.

An "exclusive" match is defined as when the most logical user expected result for the query can be satisfied by one and only one service (some common sense should be employed beyond just relying on identifying a singular match in, e.g., Gracenote -- for instance YouTube has a vast library of music content even though it's not indexed by 3rd parties and so music queries will rarely if ever be "exclusive" to a singular service unless no other music services separate of YouTube exist on the platform).

This information is non-normative.

A “universal search” page is a device provided search results interface surfacing results suitable to the user’s query from multiple different services, e.g. for the query “Mickey Mouse” each service with content relevant to “Mickey Mouse” (be that a match related to the character, or a show/movie/clip relevant to that entity) would provide rank sorted results to be presented to the user as a shelf.

This information is non-normative.

15.5 The speech input method for the target device **MUST** maintain a word error rate (WER) no greater than 15% across its supported languages.

16.0 High Dynamic Range

This section applies only to devices marketed as supporting High Dynamic Range. If a device is not marketed as HDR, the terms in this section do not apply.

For the purposes of this document, "HDR" means that a device meets all the requirements in this document.

This information is non-normative.

16.1 The isTypeSupported method **MUST** support the 'vp9.2' codec string.
16.2 The isTypeSupported method **MAY** support the AV1 codec string "av01.01.61.<bitdepth>” where <bitdepth> is either 08 for SDR bitstreams or 10 for HDR bitstreams as specified here: https://aomediacodec.github.io/av1-isobmff/#codecsparam .

16.3 EOTF

16.3.1 The isTypeSupported method **MUST** support the 'eotf' additional mime-type parameter, and support all the following values:

- 'bt709', indicating ITU-R BT.1886 support
- 'smpte2084', to indicate SMPTE 2084 EOTF support
- 'arib-std-b67', to indicate ARIB STD-B67 support

16.3.2 The device **MUST** support reading an explicitly-signaled EOTF via the *TransferCharacteristics* Matroska element with support for the following values:

- ITU-R BT.709
- SMPTE ST 2084
- ARIB STD-B67

and support correct display of content in that EOTF.

Test samples will be provided for both colorspace and EOTF curves that should allow for device makers to verify compliance with a human inspection. Certain properties, like monotonicity of a gamma ramp and accurate reproduction of the Rec. 709 color gamut, will be enforced during certification.

This information is non-normative.

16.4 Color Spaces

16.4.1 The Rec. 601, Rec. 709, and Rec. 2020 color spaces **MUST** be supported via explicit signaling of color information via the *Primaries* and *MatrixCoefficients* Matroska elements as defined by Matroska specification. Support for the Rec. 709 and Rec. 2020 values of these elements is minimally required.
16.4.2 Devices **MUST** minimally decode and display content within the Rec. 709 gamut with correct colorimetry, both in Rec. 709 and Rec. 2020 color spaces.
16.4.3 The device **MUST** produce subjectively acceptable visual behavior across input covering the full gamut of Rec. 2020.
16.4.4 Explicitly-signaled color spaces **MUST** be respected and properly decoded. If no color space is signaled, the device **MUST** default to Rec. 709.
16.4.5 The device **MUST** support 98% or more of the Rec. 709 color space, and **MUST** correctly display a test card which exercises the Rec. 709 gamut, whether that test card is encoded using the Rec. 709 or Rec. 2020 color space.

While we expect to see content that takes full advantage of Rec. 2020, we're not imposing a gamut requirement beyond Rec. 709. The check for "subjectively acceptable visual behavior" is present to avoid behavior like integer wraparound for out-of-gamut colors.

This information is non-normative.

Note: We plan to introduce stricter requirements on Color Gamut rendering capability in the 2020 YouTube Specification. More specifically, clause **16.4.2** and **16.4.5** will likely be updated as follows:

16.4.2 Devices **MUST** minimally decode and display content within the DCI-P3 gamut with correct colorimetry in Rec. 2020 color spaces.
16.4.5 The device **MUST** support 90% or more of the DCI-P3 color space, and **MUST** correctly display a test card which exercises the DCI-P3 gamut, when that test card is encoded using the Rec. 2020 color space.

This information is non-normative.

16.5 HDMI

16.5.1 If the device has an HDMI output, the device **MUST** propagate the HDR metadata conveyed in the Matroska container into HDMI *InfoFrames*.

16.6 Post-processing

16.6.1 By default, any post-processing technologies **MUST NOT** produce noticeable artifacts.

Test samples will be provided that highlight noticeable artifacts produced by existing post-processing technologies.

This information is non-normative.

17.0 WebSocket

17.1 The target device **MUST** implement the WebSocket protocol version 13 as defined by IETF RFC 6455.
17.2 The target device **MUST** implement the WebSocket API.

18.0 WebDriver

18.1 The target device **MUST** implement the WebDriver W3C Working Draft 14 February 2017, enabled for testing during certification.

19.0 Device Maintenance

19.1 The target device **MUST** be capable of receiving and installing firmware or similar software updates over a network connection automatically, and be configured to do so by default.
19.2 The target device **MUST** remain in compliance with Software Requirements for Operator Devices Specification, and of subsequent Technical Requirements versions published by YouTube for a period of three calendar years after the target device is first made available to users.

20.0 End User Privacy

20.1 The Partner **MUST** comply with all privacy laws and regulations, including those applying to Google's end user login information and data.
20.2 The Partner **MUST NOT** collect, use, maintain, or distribute any Google end user login information or user data unless otherwise authorized in writing by Google.

21.0 Internationalization

21.1 The device **MUST** provide normal weight, normal style, sans serif fonts needed to render text in the following languages: Afrikaans,Azerbaijani,Indonesian,Malay,Bosnian,Catalan,Czech,Danish,German,Estonian,English (United Kingdom),English,Spanish (Spain),Spanish (Latin America),Spanish (United States),Basque, Filipino,French,French (Canada),Galician,Croatian, Zulu,Icelandic,Italian,Swahili,Latvian, Lithuanian, Hungarian,Dutch,Norwegian,Uzbek,Polish,Portuguese (Portugal),Portuguese (Brazil),Romanian,Albanian,Slovak,Slovenian,Serbian (Latin),Finnish,Swedish,Vietnamese,Turkish,Belarusian,Bulgarian,Kyrgyz,Kazakh,Macedonian,Mongolian,Russian,Serbian,Ukrainian,Greek,Armenian,Hebrew,Arabic,Persian,Nepali,Marathi,Hindi,Bangla,Punjabi,Gujarati,Tamil, Telugu,Kannada,Malayalam, Sinhala,Thai,Lao,Myanmar (Burmese),Georgian,Amharic,Khmer,Chinese, Chinese (Taiwan),Chinese (Hong Kong),Japanese,Korean.

21.2. The device **MAY** provide bold fonts for the aforementioned languages. When a bold font is requested by the YouTube application, the device **MAY** apply faux bold if it produces legible results, otherwise the device **MUST NOT** apply faux bold.

Appendix

Supporting Resources

- ECMAScript Conformance: https://github.com/tc39/test262
- ISO 639-2 Language Codes: http://www.loc.gov/standards/iso639-2/php/code_list.php
- ISO 3166-1 Country Codes: http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm
- Media Source and Encrypted Media Extensions unit test suite: http://yt-dash-mse-test.commondatastorage.googleapis.com/unit-tests/main.html
- Starboard Lifecycle API: https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/doc/lifecycle.md
- How to test DIAL 2.0: https://sites.google.com/a/google.com/youtube-leanback-partners/testing/dialtesting

This information is non-normative.

Standards References

- HTML 4.01 Specification: http://www.w3.org/TR/html401/
- HTML5 Video Specification: http://www.w3.org/TR/html5/embedded-content-0.html#the-video-element
- Web Storage Specification: http://dev.w3.org/html5/webstorage/
- ECMAScript Language Specification: http://www.ecma-international.org/ecma-262/6.0/
- Timing control for script-based animations Specification: http://www.w3.org/TR/animation-timing/
- CSS Snapshot: http://www.w3.org/TR/CSS/
- CSS 2.1 Specification: http://www.w3.org/TR/CSS2/
- CSS Backgrounds and Borders Module Level 3: http://www.w3.org/TR/css3-background/
- CSS Text Level 3 Specification: http://www.w3.org/TR/css3-text/
- CSS Transitions Module Level 3: http://www.w3.org/TR/css3-transitions/
- CSS Animations Module Level 3: http://www.w3.org/TR/css3-animations/
- CSS Basic User Interface Module Level 3: http://www.w3.org/TR/css3-ui/
- CSS3 Generated and Replaced Content Module: http://www.w3.org/TR/css3-content/
- DOM Level 1 Specification: http://www.w3.org/TR/DOM-Level-1/
- DOM Level 2 Core Specification: http://www.w3.org/TR/DOM-Level-2-Core/
- DOM Level 2 Style Specification: http://www.w3.org/TR/DOM-Level-2-Style/
- DOM Level 3 Events Specification: http://www.w3.org/TR/DOM-Level-3-Events/
- CSSOM View Module: http://www.w3.org/TR/cssom-view/
- Roboto Web Font: https://fonts.google.com/specimen/Roboto
- Cross-Origin Resource Sharing: http://www.w3.org/TR/cors/
- HTTP Over TLS: http://www.ietf.org/rfc/rfc2818.txt
- TLS 1.2: https://www.ietf.org/rfc/rfc5246.txt
- HTTP State Management Mechanism: http://www.ietf.org/rfc/rfc2818.txt
- X.509 v3 Profile: http://www.ietf.org/rfc/rfc3280.txt
- RFC2119: http://www.ietf.org/rfc/rfc2119.txt
- RFC7231: https://tools.ietf.org/html/rfc7231
- Opus Codec: https://tools.ietf.org/html/rfc6716
- Opus Spatial Audio: https://tools.ietf.org/html/draft-ietf-codec-ambisonics
- HTTP/1.1: http://www.w3.org/Protocols/rfc2616/rfc2616.html
- XMLHttpRequest: http://www.w3.org/TR/XMLHttpRequest/
- Fetch API: http://fetch.spec.whatwg.org/
- Streams API: http://streams.spec.whatwg.org/
- WebGL: http://www.khronos.org/registry/webgl/specs/1.0/

- WebP Container Specification: https://developers.google.com/speed/webp/docs/riff_container
- Web Speech API: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html
- WebSocket: https://tools.ietf.org/html/rfc6455
- WebSocket API: https://html.spec.whatwg.org/multipage/comms.html#network
- WebDriver: https://www.w3.org/TR/2017/WD-webdriver-20170214/
- Media Element Effective Media Volume: https://www.w3.org/TR/html5/embedded-content-0.html#effective-media-volume
- Media Source Extensions: https://rawgit.com/w3c/media-source/0e0e6769f8e9968e527e1d5e07a49acc5b12c0c7/media-source.html
- Encrypted Media Extensions: https://www.w3.org/TR/2016/WD-encrypted-media-20160204/
- Web Audio API: http://www.w3.org/TR/webaudio/
- DIAL 2.0 Specification: http://www.dial-multiscreen.org/dial-protocol-specification
- BT.1886: http://www.itu.int/rec/R-REC-BT.1886-0-201103-I
- SMPTE 2048: http://standards.smpte.org/content/978-1-61482-829-7/st-2084-2014/SEC1
- Hybrid Log-Gamma: http://www.itu.int/md/R12-WP6C-C-0481
- Rec. 601: https://www.itu.int/rec/R-REC-BT.601
- Rec. 709: https://www.itu.int/rec/R-REC-BT.709
- Rec. 2020: https://www.itu.int/rec/R-REC-BT.2020-1-201406-I

This information is non-normative.