YouTube TV HTML5 Technical Requirements 2018

Version: 2017/10/05

Purpose of this Document

This document details the base requirements for supporting the YouTube TV HTML5 application on embedded device browsers. The Technical Requirements are intended to provide a low-level description of the standards, capabilities, and configuration details necessary for the application to function.

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", mean that the definition is an absolute requirement of the specification.
- **MAY** - This word, or the adjective "OPTIONAL", mean that an item may or may not be implemented at the discretion of the implementor.

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.

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.
2.2 The target device **MUST** support ECMA-262-6 strict mode.
2.3 The target device **MUST** support Timing control for script-based animations, also known as requestAnimationFrame.
2.4 The target device **MUST** make JavaScript stack traces available from window.onerror.
2.5 The target device **MUST** include column numbers in JavaScript stack traces.

3.0 DOM

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.

4.0 CSS

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

- TTF
- WOFF

4.2 The target device **MUST** use the 'Roboto' web font served by the application.
4.3 If the target device supports system languages with characters not included in the 'Roboto' font, it **MUST** provide the application with system fonts containing those characters.

The application will attempt to use fonts built into the 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 application is infeasible

due to possible memory limitations on a target device.

This information is non-normative.

4.4 The target device **MUST** support the following CSS selector types:

- Type
- ID
- Class (includes multiple parallel classes e.g. .class1.class2)
- Descendant Selector
- Child Selector

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 **MAY** support concurrent playback through two video tags, with the following video stream decode capabilities:

Display resolution supported by device	Dual decode requirement
1280 x 720	Two 1280 x 720 streams at 30fps
1920 x 1080	Two 1920 x 1080 streams at 30fps
Greater than 1920 x 1080	One 3840 x 2160 stream at 30fps One 1920 x 1080 stream at 30fps

5.2 The target device **MUST** be capable of resizing the video layers to arbitrary sizes dynamically during video playback.
5.3 The HTML video element **MUST** be able to download media for playback using HTTPS.
5.4 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

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

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

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

- HTMLMediaElement.volume
- HTMLMediaElement.muted
- HTMLMediaElement.currentTime, including the following configuration requirement:
    - The value of currentTime **MUST** be accurate to within 250 milliseconds during active playback.
    - The value of currentTime **MUST** be accurate to within 32 milliseconds when content is paused.
- HTMLMediaElement.duration
- HTMLMediaElement.buffered
- HTMLMediaElement.paused
- HTMLMediaElement.ended
- HTMLMediaElement.autoplay

5.7 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.

5.8 Media Source and Encrypted Media Extensions

5.8.1 The target device **MUST** properly implement Encrypted Media Extensions W3C Candidate

Recommendation 05 July 2016, including support for the following key system:

- com.widevine.alpha
  - The target device **MUST** implement the com.widevine.alpha key system using OEMCrypto version 12 or greater.

and including the following configuration requirements:

- The key system(s) implemented by the target device **MUST** support key rotation, with a minimum of eight (8) MediaKeySession objects supported. A minimum of sixteen (16) keys per MediaKeySession object **MUST** be supported.
- 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 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.

5.8.2 The target device **MUST** properly implement Media Source Extensions W3C Candidate Recommendation 05 July 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 isTypeSupported method **MUST** accept and return correct values for the following additional parameters:
  - `width`
    - The x-axis video playback resolution in pixels.
  - `height`
    - The y-axis video playback resolution in pixels.
  - `framerate`
    - The video playback frame rate in frames per second.
  - `bitrate`

        - The playback bitrate in bits per second.
    - `channels`
        - The number of absolute audio channels.
    - `cryptoblockformat`
        - The supported encrypted block format.
- 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')
```

This information is non-normative.

6.0 Network Stack

6.1.1 The target device **MUST** support XMLHttpRequest Level 2.
6.1.2 The target device **MUST** support the Fetch API and the Streams API, with ReadableByteStream support for a Fetch API Response.
6.2 The target device **MUST** support Cross-Origin Resource Sharing.
6.3 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.

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.

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

The application makes a non-trivial number of HTTP(S) requests to fetch 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.
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 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 application elements correctly.
7.2 The target device **MUST** display the application in full screen, without browser chrome or other system elements.
7.3 The target device **MUST** be capable of rendering the application interface at one of the following supported resolutions:

- 1280 x 720
- 1920 x 1080
- 2560 x 1440
- 3840 x 2160

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

- window.devicePixelRatio, including the following configuration requirement:
    - If the target device's video decode capability is less than or equal to 1920x1080, the value of this property **MUST** be:
        - 1
    - 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 application is rendered will set an upper bound for the maximum content resolution selected by the application's adaptive bit rate algorithm, regardless of whether or not the device uses separate compositing paths for the 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.

7.5 The browser's spatial navigation and element highlighting **MUST** be disabled when rendering the application UI.
7.6 The target device **MUST** provide the ability to overlay/composite HTML content on top of the video surface.
7.7 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:

- PNG (ARGB 32-bit)
- 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
        - Lossless compression
        - Metadata
        - Transparency
        - Color Profile

9.0 Remote Keys and Key Events

9.1 Device remote keys not implemented by the 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 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 = `0x47`)
- Search (keyCode = `0x53`)
- 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.

9.4.2 The device **MAY** support the following keys if the keys exist on the device remote:

- 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 frames 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.
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** be able to decode and present the following container formats, video codecs, and audio codecs:

- Container: WebM, MP4
  - Video: VP9 Profile 0, VP9 Profile 2 10-bit
  - The target device **MUST** support decoding of VP9 bitstreams at the decoding frequency specified in the table below:

Spatial Resolution	**Displayed Frame Rate**	**Minimum number of display frames between two consecutive ALT-REF frames**	**Corresponding decoding frame rate**
All resolutions up to 1080P included	Up to 30 frames/sec included	4	37.5 frames/sec
	Up to 60 frames/sec included	4	75 frames/sec
2K - 2560x1440	Up to 30 frames/sec included	4	37.5 frames/sec
	Above 30 and up 60 frames/sec included	5	72 frames/sec
	Up to 30 frames/sec included	6	35 frames/sec

4K - 3840x2160	Above 30 and up 60 frames/sec included	12	65 frames/sec

- Video: H.264/MPEG-4 AVC
- Audio: AAC-LC

  The device **MUST** also support these formats for Media Source playback:

  - ISO/IEC 14496-12:2012: ISO Base Media File Format (ISO BMFF)
  - ISO/IEC 23001-7:2012: Common encryption in ISO BMFF (as specified by 23009-1 section 6.3 - DASH MPD and segment formats)
- Audio: Opus, as defined by IETF RFC 6716.
  - Stereo
    - Decoding support for stereo audio at 256kbps
  - If the target device features HDMI output, it **MUST** have decoding support for 6 channels (5.1) at 576kbps
  - The target device **MAY** support the following configurations:
    - Spatial Audio
      - Decoding support for 6 channels under Channel Mapping Family 2 as defined in IETF Ambisonics Specification
      - Decoding support for 18 channels under Channel Mapping Family 3 as defined in IETF Ambisonics Specification

10.5 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 requirement:
    - 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.

10.6.1 If the target device has a GPU, it **MUST** properly 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. This technology is only supported in Cobalt.

This information is non-normative.

10.6.2 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
    - WebGLRenderingContext.texImage2D **MUST** support VideoElement and Image as a source.
- WebGLRenderingContext.texParameterf
- WebGLRenderingContext.texParameteri
- WebGLRenderingContext.texSubImage2D
- WebGLRenderingContext.texSubImage2D
- WebGLRenderingContext.uniform1f
- WebGLRenderingContext.uniform1fv
- WebGLRenderingContext.uniform1fv
- WebGLRenderingContext.uniform1i
- WebGLRenderingContext.uniform1iv
- WebGLRenderingContext.uniform1iv

- WebGLRenderingContext.uniform2f
- WebGLRenderingContext.uniform2fv
- WebGLRenderingContext.uniform2fv
- WebGLRenderingContext.uniform2i
- WebGLRenderingContext.uniform2iv
- WebGLRenderingContext.uniform2iv
- WebGLRenderingContext.uniform3f
- WebGLRenderingContext.uniform3fv
- WebGLRenderingContext.uniform3fv
- WebGLRenderingContext.uniform3i
- WebGLRenderingContext.uniform3iv
- WebGLRenderingContext.uniform3iv
- WebGLRenderingContext.uniform4f
- WebGLRenderingContext.uniform4fv
- WebGLRenderingContext.uniform4fv
- WebGLRenderingContext.uniform4i
- WebGLRenderingContext.uniform4iv
- WebGLRenderingContext.uniform4iv
- WebGLRenderingContext.uniformMatrix2fv
- WebGLRenderingContext.uniformMatrix2fv
- WebGLRenderingContext.uniformMatrix3fv
- WebGLRenderingContext.uniformMatrix3fv
- WebGLRenderingContext.uniformMatrix4fv
- 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

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.
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 device **MUST** implement the out-of-box experience requirements as described in Section 9 of the DIAL 2.1 protocol.
11.5 The 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 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 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, and chipset model number.
    - The network operator field **MUST** contain the name of the network operator 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.
- 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`.

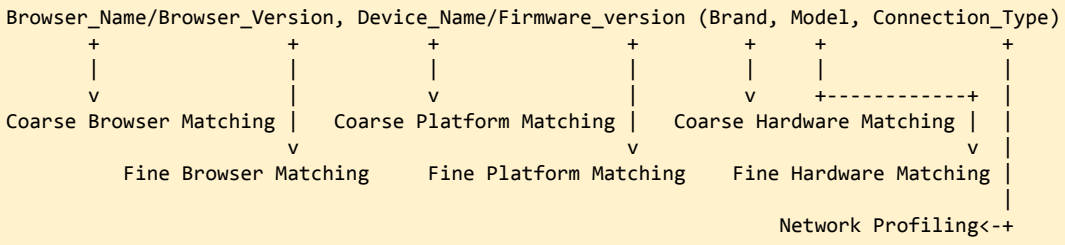
13.1.2 The device **MUST** provide the above User-Agent fields in one of the two following formats:

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:



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 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 the W3C standard (Section 14.4) 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 application. The value:
  - **MUST** be `menu` if the YouTube logo is selected from the device's menu screen,
  - **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 `preload` if the device has automatically opened the application without user input.
  - and **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 launched by the user via selecting a universal search result:
https://www.youtube.com/tv?launch=search

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 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 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 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 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 application directly to a specific YouTube channel.

Example: https://www.youtube.com/tv?c=UC_x5XG1OV2P6uZZ5FSM9Ttw

This information is non-normative.

13.4 Preload API

Devices setting the `launch=preload` parameter as described in section 13.3 of this specification **MUST** use the following fragment identifiers to manage the state of the application while it remains loaded in memory:

- `#start` - Signals the application to reset to a state as if it was started for the first time.

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

This information is non-normative.

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

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

This information is non-normative.

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

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

This information is non-normative.

- `#play` - Signals the application to initiate or resume playback of the currently-loaded video. This fragment may be combined with the `v` and/or `t` parameters to initiate playback of a specific video

and/or at a specific time.

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

- #pause - Signals the application to pause playback of the currently-loaded video.

Example: https://www.youtube.com/tv?launch=preload#pause
This information is non-normative.

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 The application **MUST** respond to user input at a level acceptable to YouTube.
14.2 The target device **MUST** be able to maintain application responsiveness when device system functions are used over or alongside the application.

14.3 Loading time

14.3.1 The target device **MUST** be able to load the application in 10 seconds, measuring from the time the user selects the application for launch to the time the browser finishes rendering the application and factoring out network transfer times.

14.4 Input latency

14.4.1 The target device **MUST** have a physical input latency no greater than 200 milliseconds, measuring from the time a user input signal is received by the device to the time the first frame update is initiated.

14.5 Transition times

14.5.1 The target device **MUST** be able to complete a tile-to-tile transition within 200 milliseconds, measuring from the time the first frame update is initiated to the time the next video thumbnail finishes rendering in place and factoring out network transfer times.

14.5.2 The target device **MUST** be able to complete a row-to-row transition within 210 milliseconds, measuring from the time the first frame update is initiated to the time the next content row finishes rendering in place and factoring out network transfer times.

14.5.3 The target device **MUST** be able to complete a browse-to-watch transition within 1.5 seconds, measuring from the time the first frame update is initiated to the time the video content begins playback and factoring out network transfer times.

14.6 Frame rate

14.6.1 The target device **MUST** be able to maintain a minimum application frame rate of 30 frames per second at all times.

14.7 Video endurance

14.7.1 The target device **MUST** be able to maintain video playback continuously for 12 hours.

15.0 Speech

15.1 If the target device's remote control includes built-in microphone input, it **MAY** 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 application features, such as search.

This information is non-normative.

16.0 High Dynamic Range (OPTIONAL)

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 target device **MUST** be able to decode VP9 Profile 2 at a bit depth of 10 bits, in addition to Profile 0 at bit depth of 8 bits.
16.2 The isTypeSupported method **MUST** support the 'vp9.2' codec string.

16.3 EOTF

16.3.1 The isTypeSupported method **MUST** support the 'eotf' additional mime-type parameter, and support 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 colorimetery, 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.

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 all software-related items of these Technical Requirements, 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.

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
- YouTube TV Preload API examples: https://sites.google.com/a/google.com/youtube-leanback-partners/specifications/preloadapi
- 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
- 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://www.w3.org/TR/2016/CR-media-source-20160705/
- Encrypted Media Extensions: https://www.w3.org/TR/2016/CR-encrypted-media-20160705/
- 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.

Version History

2018/03/23

- Removed typo in Section 13.1.1 regarding model year in Device_Name.

2017/10/05

- CANVAS has been moved to the WebGL section, and is only applicable to devices implementing WebGL
- Removed HTMLIFrameElement.contentWindow
- Removed WOFF2
- Removed CSS Flexible Box Layout Model Level 1
- Dual Video is now MAY
- Clarified that audio may be muted when video.playbackRate is 0.25
- Removed IPv6 v6ops best practices requirement due to ambiguity
- Changed Closed Caption key code to 0x1CC from 0xAF
- Changed YouTube button key code to 0x3000 from 0xAC
- Added optional remote keys that may be passed to the application (9.4.2)
- Opus 5.1 surround decoding is only required for devices featuring HDMI output
- WebAudio can support MP3 or WAV
- Some WebAudio elements and methods are now MAY
- WebGL and CSS Spherical Filter Extensions (360 video) only required for devices featuring GPU
- Added launch=voice to indicate the application was launched via voice command
- Updated MSE and EME non-normative reference links
- Row-to-row transition timing is now 210ms

2017/03/27

- Clarified devicePixelRatio configuration requirements in Section 7.4.

2017/03/20

- Fixed typo in requirement numbering of Section 4.

2016/03/01

- Initial release.