YouTube Software Requirements for CE Devices

2020

Version: 2019/11/06

Purpose of this Document

This document details the base requirements for supporting the YouTube HTML5 applications for Living Room Devices on embedded device browsers. The Software Requirements for CE Devices specification is intended to provide a low-level description of the standards, capabilities, and configuration details necessary for the YouTube application to function.

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 Software Requirements for CE Devices specification 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 this official Software Requirements for this CE Devices specification.

Terminology

The term “YouTube application” will herein reference the main YouTube HTML5 application (“YouTube on TV Living Room 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

0.0 General Run-time Requirements

0.1 The target device **MUST** use the Cobalt runtime as the underlying Browser engine for the YouTube applications and the Starboard version for that runtime shall be equal or greater than 20.lts.stable (LTS = Long Term Support). In addition,

0.1.1 The target device **MUST** properly implement all Starboard APIs
0.1.2 The target device **MUST** pass all items of the following test suites:

- All Cobalt tests - including Starboard NPLB Tests (see src/starboard/nplb/).

0.1.3 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
0.1.4 The Integration of Cobalt with the target device operating system / software environment, achieved through the Starboard (Sb) interfaces shall be such that the Cobalt functionality is complete, including the following features (which were explicitly referenced in past years) :

- Response to standard keyboard and/or remote control events.
- Web-API
- Web Fonts
- W3C Selectors

- CSS properties and transitions
- HTML Video events
- Full HTTPS/TCP/IP Network protocol stack
- Websocket APIs and WebDriver Interface
- W3C Web Audio API
- Local Storage APIs

Additional normative aspects of the software environment are captured in subsequent sections

1.0 CSS

1.1 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.

***Implementation notes*:**

- System fonts can be added to Cobalt by creating a custom fonts.xml file in which file paths to system fonts are listed. The fonts.xml file should live in the directory specified by SbSystemGetPath(kSbSystemPathFontConfigurationDirectory, …).
- More details on configuring Cobalt's font for a specific platform can be found here: https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/content/fonts/README.md

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.

2.0 HTML Video

2.1 Dual Video Playback

2.1.1 The target device **MUST** support concurrent playbacks of two video elements, with the following conditions on video decoding capabilities being satisfied:

- The target device **MUST** be capable to decode two video streams concurrently, one at the maximum decoding resolution associated with the category to which the target device belongs and the second one is a video decoder capable of decoding H.264 Main Profile, 432x240, 15 frames/sec video bitstreams.
- The secondary decoder **MAY** be implemented in hardware.

1. Tolerance on decoding frame rate for the secondary video is motivated by the fact that there is no audio and therefore no synchronization requirements for that second stream
2. There is no DRM capability requirements for the secondary video stream

This information is non-normative.

2.2 The target device **MUST** be capable of resizing each of the video layers to arbitrary sizes dynamically while meeting the following requirements

2.2.1 The video and UI are synchronized within the following margins

- The maximum lag/advance **MUST** be less than 100ms and the appearance of any black bars **MUST** be avoided for as long as the resizing is happening.

An option for making animated transitions look as clean as possible is to crop the video so as to avoid displaying graphics black bars. This likely means resizing in the graphics layer should never be faster than resizing in the video layer on maximizing the window and conversely, resizing in the graphics level should never be slower than resizing in the video layer on minimizing the window

This information is non-normative.

2.2.2 The video playback **MUST** continue at 30fps or more for at least 75% of the transition time and the UI frame rate **MUST** remain above 27fps.

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

- HTMLMediaElement.currentTime, including the following configuration requirements:
    - The value of currentTime **MUST** be accurate to within 250 milliseconds during active playback, 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, applicable whether the video features a standard frame rate or a high frame rate.
- HTMLMediaElement.playbackRate
    - The target device **MUST** support the following values for this property: 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00.

***Implementation note*:**

- The function SbPlayerSetPlaybackRate() should accept and properly react to the playback rate parameters listed above
    - 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. 60 frames per second is the maximum required decode capability, even when playbackRate > 1.0

This information is non-normative.

2.4 Media Source and Encrypted Media Extensions

2.4.1 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 15 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 sixteen (16) MediaKeySession objects supported. A minimum of sixteen (16) keys per MediaKeySession object **MUST** be supported.
- Devices implementing Widevine **MUST** support subsample encrypted block format for any encrypted 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 12.1 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.

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 Living Room 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.

2.4.2 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 target device **MUST** allow at least 30 MBytes of compressed video data and 5 MBytes of audio data to be appended before beginning to evict time ranges.
    - If the target device is capable of playing 4K content, the target device **MUST** allow at least 50 MBytes of compressed video data and 5 MBytes of audio data to be appended before beginning to evict time ranges.
    - If the target device is capable of playing 4K HDR content, the target device **MUST** allow at least 80 MBytes of compressed video data and 5 MBytes of audio data to be appended before beginning to evict time ranges.
    - If the target device is capable of playing 8K content, the target device **MUST** allow at least 300 MBytes of compressed video data and 5 MBytes of audio data to be appended before beginning to evict time ranges.

***Implementation note*:**
- These buffer budgets are specified by the value returned by SbMediaGetVideoBufferBudget() and SbMediaGetAudioBufferBudget()

- Media playback **MUST** begin if at least 0.5 second of media has been appended.
- Media playback **MUST** begin within 0.5 second of sufficient data for playback being appended.
- Before end of stream is signaled, media playback **MUST** continue until at most 0.5 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 target device must begin playback within 0.5 second, and continue playing for at least 2.5 seconds after that before pausing due to buffer underrun. Where possible, the target device should play all 3 seconds before pausing. Once the remaining 7 seconds are appended and end of stream is signaled, the target 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 decoder 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`
        - Parameter that the target device uses to report about its capability to decode a video into a texture that will subsequently be processed by the target 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.

***Implementation note*:**
- Calls to isTypeSupported() method are forwarded to the SbMediaCanPlayMimeAndKeySystem method

- If the target device supports HDR, it **MUST** 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 target 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 target 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 target 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.

2.4.3 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

***Implementation note*:**

- The SbPlayerGetInfo2() function will return a SbPlayerInfo2 struct. The |total_video_frames|, |dropped_video_frames| and |corrupted_video_frames| fields of this struct must be populated correctly

3.0 Network Stack

3.1 Root Certificates

3.1.1 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.

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

This information is non-normative.

3.2 IP and DNS

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

3.3 Cookies and Storage

3.3.1 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.
3.3.2 Cookies and Web Storage data **MUST** be stored locally on the device.
3.3.3 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.
3.3.4 The target device **MAY** implement duplication of Cookie and Web Storage data across an additional one or two physical and internal memory bank(s) if the target device includes such independent units of memory units. In that case, the firmware in the target device will provide the ability to reference the secondary or tertiary local storage if the primary or secondary unit , respectively, failed.

***Implementation note*:**
- Cookies and local storage will call the SbStorage API defined in storage.h

4.0 Rendering

4.1 The target device **MUST** be capable of rendering and compositing all YouTube application elements correctly
4.2 The target device **MUST** display the YouTube application in full screen, without browser chrome or other system elements.
4.3 The target device **MUST** be capable of rendering the YouTube application interface at one of the following supported resolutions:

Target Device is <= FHD		**Target Device is > FHD or >4K**	
Video Resolution (Horizontal x Vertical dimension)	Miinimum 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

The *≤FHD*, *>FHD* and >4K device categories are defined in the companion *"YouTube Hardware Requirements for CE and Operator 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
- >4K: Any device capable of outputting more than 8,294,400 pixels simultaneously

This information is non-normative.

4.4 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 converted to the corresponding luminance and chrominance values in the new extended luma and color system used by the display. The resulting sample values are rendered into an HDR surface that is compatible with the HDR display mode currently selected on the target device.

4.5 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 and less or equal than 3840x2160, the value of this property **MUST** be one of the following:
    - 1
    - 1.5
    - 2
  - If the target device's video decode capability is greater than 3840x2160, the value of this property **MUST** be one of the following:
    - 2
    - 4

***Implementation note*:**

- The window.devicePixelRatio property is provided by the |video_pixel_ratio| field of the SbWindowSize struct returned by the SbWindowGetSize() function

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.

5.0 Remote Keys and Key Events

5.1 Device remote keys not implemented by the YouTube application or reserved for device system functions **MUST NOT** dispatch any key event or key code.
5.2 Devices implementing a system-wide "exit" key **MUST** use that key to gracefully exit the YouTube application.
5.3 The Lifecycle event generation **MUST** be implemented in full.

***Implementation note*:**

- The Starboard Lifecycle API is documented here: https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/doc/lifecycle.md

In particular,

5.3.1 The lifecycle state **MUST** be queried by the YouTube application via the Page Visibility State API and the document's focus/blur state.

5.3.2 When the YouTube app becomes non-interactive, the implementation **MUST** generate a 'blur' event on the window object.

***Implementation note*:**

- Starboard in that case should generate the kSbEventTypePause event

5.3.3 When platform makes Cobalt interactive again, the implementation **MUST** generate a 'focus' event on the window object.

***Implementation note*:**

- Starboard in that case should generate the kSbEventTypeUnpause event

5.3.4 When the app is made invisible, this **MUST** correspond to document.hidden being set to true and a 'visibilitychange' event on the document object.

***Implementation note*:**

- Starboard in that case should generate the kSbEventTypeSuspend event

5.3.5 When the app is made visible, this **MUST** correspond to document.hidden being set to false and a 'visibilitychange' event on the document object.

***Implementation note*:**

- Starboard in that case should generate the kSbEventTypeResume event

5.4 The implementation **MUST** support deep links

- The target device **MUST** be able to launch (or resume) the YouTube application within the latency range specified in the Performance section of this specification and have it immediately play a specific video.

***Implementation note*:**

- In this case, Starboard should generate a kSbEventTypeLink event with the URL representing the video to play as a parameter.

- If the YouTube application has not been launched yet, the platform **MUST** provide an initial deep link URL parameter.

***Implementation note*:**
- The deep link is provided as a value for the SbEventStartData::link field

5.5 The target 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)

***Implementation note*:**

The platform generates the appropriate kSbEventTypeInput events for key presses and releases. At a minimum the platform must support six-point navigation, which includes the following keys:
- Left (kSbKeyLeft = 0x25)
- Right (kSbKeyRight = 0x27)
- Up (kSbKeyUp = 0x26)
- Down (kSbKeyDown = 0x28)
- Enter (kSbKeyReturn = 0x0D)
- Escape (kSbKeyEscape = 0x1B)

5.6.1 The target 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)
    - Only for remotes that have a microphone and the microphone button is dedicated to voice queries or commands directed to one of the YouTube apps.
- 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)
    - For remotes that implement dedicated YouTube application hardware buttons. This keycode needs only be sent when the matching YouTube application is in use.
    - See Section 11.5 below (Request Parameters) for the request to send whether the corresponding YouTube application is already running or not
    - This key must be sent when the user has pressed the launch button associated with the corresponding YouTube app on a remote. See Terminology section at the top of this specification for the possible YouTube apps.

***Implementation note*:**

The corresponding keys in Cobalt are

- Play (kSbKeyPlay = 0xFA)
- Pause (kSbKeyPause = 0x13)
- Play/Pause (kSbKeyMediaPlayPause = 0xB3)
    - Only for remotes that implement play and pause functionality on a single button.
- Stop (kSbKeyMediaStop = 0xB2)
- Fast Forward (kSbKeyMediaFastForward = 0xE4)
- Rewind (kSbKeyMediaRewind = 0xE3)
- Space (kSbKeySpace = 0x20)
- Backspace (kSbKeyBackspace = 0x08)
- Delete (kSbKeyDelete = 0x2E)
- Search (kSbKeyBrowserSearch = 0xAA)
- Microphone / Voice (kSbKeyMicrophone = 0x3002)
- Previous (kSbKeyMediaPrevTrack = 0xB1)
- Next (kSbKeyMediaNextTrack = 0xB0)
- Closed Captions / Subtitle (kSbKeySubtitle = 0x1CC)
- Red (kSbKeyRed = 0x193)
- Green (kSbKeyGreen = 0x194)
- Yellow (kSbKeyYellow = 0x195)
- Blue (kSbKeyBlue = 0x196)
- YouTube button (kSbKeyLaunchThisApplication = 0x3000)

5.6.2 The target 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)

***Implementation note*:**

The corresponding keys in Cobalt are

- Channel Up (kSbKeyChannelUp = 0x1AB)
- Channel Down (kSbKeyChannelDown = 0x1AC)
- Last / Previous / Recall Channel (kSbKeyLast = 0x25F)
- Audio Track Selection (kSbKeyMediaAudioTrack = 0x3001)
- Info / Display (kSbKeyInfo = 0x1C9)
- Guide / EPG (kSbKeyGuide = 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.

5.7 The target 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

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

- Window.keypress

6.0 Media Playback

6.1 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.

6.2 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.

6.3 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.

6.4 The target device **MUST** implement the CSS Spherical Filter Extension for hardware-accelerated rendering of spherical videos

***Implementation note*:**

- The platform **MUST** support decode-to-texture and set the 'enable_map_to_mesh': 1 gyp variable.

- More detailed instructions for enabling spherical (360) video in Cobalt: https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/doc/spherical_video.md

6.5 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, that is the set of video compression formats, resolutions and frame rates that are associated with the target device category.

7.0 DIAL

7.1 The target device **MUST** implement DIAL protocol version 2.2. In particular, the target device **MUST** be capable of properly parsing arbitrary and multiple parameter values posted to the DIAL server's additionalDataUrl structure.
7.2 The target device **MUST** implement DIAL Wake-up functionality as described in Section 7 of the DIAL 2.2 protocol specification

7.3 Wake-up loading time

7.3.1 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.

7.4 The target device **MUST** implement the out-of-box experience requirements as described in Section 9 of the DIAL 2.2 protocol.
7.5 The target device **MUST** have a default configuration enabling discovery of its DIAL server by other devices.
7.6 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.
7.7 The target device **MAY** provide the user with an option to disable the DIAL server on the device.
7.8 The target device's DIAL server **MUST NOT** accept connections from the Default Gateway on the local subnet.

8.0 Loading and Unloading the YouTube Application

8.1 The YouTube application **MUST** be loaded via the following URL, with any URL parameters and fragments appended as required: https://www.youtube.com/tv

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

9.0 YouTube Application Identification & Configuration

9.1 Device Authentication

9.1.1 The target device **MUST** include a Security Hardware or Trusted Execution Environment to hold the value of a secret_key field. The device **MUST** configure three distinct secret key values, in case the first secret key has been leaked.
The value of certification_scope can be stored in an un-secure storage. The length of each of these two fields will

not exceed 128 bytes. YouTube will supply the values of the certification_scope and secret_key fields, through the YouTube Partner portal.
Both secret_key and certification scope are used jointly for the purpose of device authentication and access control to the YouTube application.

9.1.2 If the current secret_key is leaked, device **MUST** activate a different secret_key in the Security Hardware or Trusted Execution Environment, within a reasonable time. In a rare case, if all keys on the device are leaked, device MUST use firmware or software updates, to deploy a new secret_key, within a reasonable time. Failing to update secret_key might lead to a dysfunctional YouTube experience.

9.1.3 Secret_key values **MUST** be treated as confidential. To avoid accidental leakage, communication of secret_key values **MUST** be encrypted and within trusted group.

***Implementation note*:**

The YouTube application launcher will obtain the certification scope by calling the SbSystemGetProperty API with the kSbSystemPropertyCertificationScope parameter.
Cobalt will then call SbSystemSignWithCertificationSecretKey() to sign (via HMAC-SHA256) a message using the device's secret key. The signed message hash will then be sent as a URL parameter to verify that the device has been certified and authorized to use the YouTube service.

Access to the secret_key field values should be managed as any other decryption key or other sensitive information on the target device.

This information is non-normative.

9.2 Client ID (User-Agent String)

9.2.1 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
  - This field **MUST** be set to “Cobalt”
- Browser_Version
- Device_Name
  - This **MUST** be a fully-concatenated and underscore-delimited string containing the device manufacturer, device type, chipset model number, model year. The Device_Name format is therefore <*Original Design Manufacturer*>_<*Device Type*>_<*Chipset Model Number*>_<*Model Year*> and the 5 device attributes in the Device_Name string are defined as follows:
    - The Original Design Manufacturer - ODM - field **MUST** contain the name of the Corporate entity responsible for the manufacturing/assembly of the device on behalf of the Business entity owning the Brand. The field **MUST** still be present even if the name of the device manufacturer is the same as the device Brand name
    - The device type field **MUST** be one of the following values: "BDP", "GAME",

"OTT", "STB" or "TV" according to the following definitions
    - BDP: Blu-Ray Player. Such a device is capable of playing a Blu-Ray disc and also has the capability to connect to the Internet and to process IP-traffic
    - GAME: Game Console. Such device is designed to play games on-line and/or from disk and it also has the capability to connect to the Internet
    - OTT: Over-The-Top device. Such devices are solely designed to connect to the Internet and to process IP traffic and do not feature any integrated display
    - STB: Set-Top-Box: Such devices include a tuner and demodulator for non-IP -based video services and there is no integrated display
    - TV : Television Set : Such devices features an integrated display
- 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 "2020" if it implements the full set of 2020 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 field **MUST** be set according to the following rules:
        - If the Device Type is "BDP": In this case, the brand shall be the name of the Corporate entity selling the device as seen by the consumer purchasing the device.
        - If the Device Type is "GAME": In this case, the brand shall be the name of the Corporate entity providing the gaming software environment in the device
        - If the Device Type is "OTT": In this case, the brand shall be the name of the Corporate entity selling the devices as seen by the consumer purchasing the device.
        - If the Device Type is "STB": In this case, the brand shall be the name of the Service Operator. This should be the Business Entity providing the service and NOT the name of the Internet Service
        - If the Device Type is "TV": In this case, the brand shall be the name of the manufacturer selling the device as seen by the consumer purchasing the device
- Model
    - This **MUST** be the final production model number or name as it appears on the device label.
- Connection_Type
    - This describes the type of network connection used by the device at YouTube application launch, and **MUST** be either `Wired` or `Wireless`.

***Implementation note*:**

The Cobalt user agent is constructed in this code (from which the information in this document is extracted): https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/browser/user_agent_string.cc

It has this format:

Mozilla/5.0 (**{OS_NAME_AND_VERSION}**) Cobalt/19.lts.1.12345-devel (unlike Gecko) v8/6.5.254.43 gles Starboard/10,
{ORIGINAL_DESIGN_MANUFACTURER}_{DEVICE_TYPE}_{CHIPSET_MODEL_NUMBER}_{MODEL_YEAR}/{FIRMWARE_VERSION} (**{BRAND}**, **{MODEL}**, **{CONNECTION_TYPE}**)

All of the implementation-specified parameters are bolded. They are derived as follows:

- **OS_NAME_AND_VERSION**
  - SbSystemGetProperty(kSbSystemPropertyPlatformName, …)
- **ORIGINAL_DESIGN_MANUFACTURER**
  - SbSystemGetProperty(kSbSystemPropertyOriginalDesignManufacturerName, …)
- **DEVICE_TYPE**
  - SbSystemGetDeviceType()
- **CHIPSET_MODEL_NUMBER**
  - SbSystemGetProperty(kSbSystemPropertyChipsetModelNumber, …)
- **MODEL_YEAR**
  - SbSystemGetProperty(kSbSystemPropertyModelYear, …)
- **FIRMWARE_VERSION**
  - SbSystemGetProperty(kSbSystemPropertyFirmwareVersion, …)
- **BRAND**
  - SbSystemGetProperty(kSbSystemPropertyBrandName, …)
- **MODEL**
  - SbSystemGetProperty(kSbSystemPropertyModelName, …)
- **CONNECTION_TYPE**
  - SbSystemGetConnectionType()

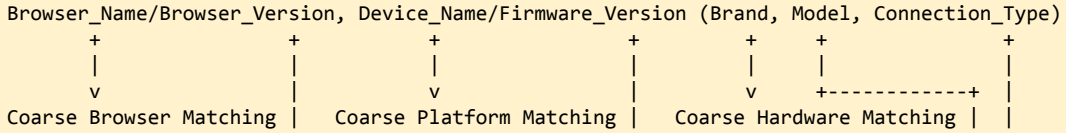
9.2.2 The characters used to represent the Device_Name field MUST be from the following character set: [a-zA-Z0-9\-] and the characters used to represent all other User-Agent fields MUST be from the following character set: [a-zA-Z0-9\-_]

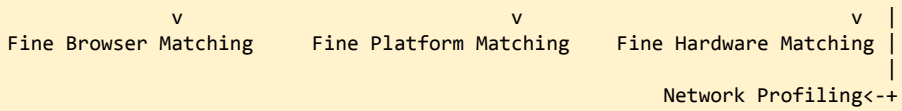
9.2.3 The device **MUST** provide the above User-Agent fields in one of the two following formats: Browser_Name/Browser_Version, Device_Name/Firmware_Version (Brand, Model, Connection_Type)

Example of two YouTube User-Agent strings:

> Mozilla5.0 (Linux) Cobalt/11.119147-gold (unlike Gecko) Starboard6,Google, _TV_MK120P_2020/V3.02 (Google, 43D2700K-US, Wireless)

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





This information is non-normative.

9.3 Language and Territory

9.3.1 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.

9.3.2 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.

9.4 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. A launch= parameter **MUST** be included for any deeplinks when YouTube is not already open and in the foreground. Moreover, the parameter 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.
    - **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 `promo` if the selected YouTube video is a result of a promotional campaign.
    - **MUST** be `search` if a YouTube video is selected from a device's universal search, whether the query comes from Voice or Text or any other user input.
    - **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

Note: There should not be any 'launch=' parameter sent when the app is launched on the target device via DIAL.

This information is non-normative.

- v - A YouTube video ID.
  - This parameter **MAY** be used to launch the YouTube application with a specific video.
  - The **“v=”** parameter **MUST** be present for any play a specific YouTube video queries
    - The value of the “v” parameter **MUST** be set to a YouTube video ID

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.
  - The **“q=”** parameter **MUST** be present for any search YouTube queries
    - The value of the “q” parameter **MUST** be set to the search keywords. Keywords in the

string **MUST** be separated by a '+' sign (e.g. "q=binging+with+babish"). The query string **MUST** be UTF-8 URL encoded

- For voice searches, if possible, remove grammar around search keywords. For example, if the user says "Search for Binging with Babish on YouTube" then the parameter should be captured as q=binging+with+babish

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.

- **"inApp=true" MUST** be included for all deeplinks when the YT app is already open and running in the foreground.
- **"vq=" MUST** be included for all voice deeplinks when an action is taken within YouTube (e.g. "Show me yoga videos on YouTube")
  - The value of "vq" **MUST** be set to the full/unedited text voice query. Keywords in the string **MUST** be separated by a '+' sign (e.g ."vq=search+for+binging+with+babish+ on+YouTube"). The query string **MUST** be UTF-8 URL encoded.
  - **Exceptions - "Open YouTube"**: If user issues a voice query that just launches YouTube (e.g. "Open YouTube") and does not play a video or execute a search, then do not include vq parameter.

URL Examples

User enters YouTube via choosing a result from the device's Universal Search

Universal Search where the user input was by keyboard

- Use Case: User types "yoga videos" into device universal search and then selects a YouTube video from the results.
- URL: https://www.youtube.com/tv?launch=search&v=wU90dfDDUiQ
- Note: Don't send over navigational/selection commands issued by the user, e.g. if after the user searched for "yoga videos" they then selected a video from the search results by saying "play the second one" the deeplink URL should still be as shown above.

Universal Search where the user input was by voice

- Use Case: User issues a device universal search by saying something like "Show me yoga videos" from the device home screen. Then the user selects a YouTube video from the search results.
- URL: https://www.youtube.com/tv?launch=search&launch=voice&v=wU90dfDDUiQ
- Note: As with the above example, don't send over navigational/selection commands issued by the user.

Universal Search where the user selects "Search on YouTube"

- Use Case: User issues a device universal search via keyboard or voice. Then the user selects "Search on YouTube" button as the last tile on the Universal Search Results.
- URL voice: https:///www.youtube.com/tv?launch=search&launch=voice&q=yoga+videos

- URL keyboard: https:///www.youtube.com/tv?launch=search&q=yoga+videos
- Note: As with the above example, don't send over navigational/selection commands issued by the user.

User requests content directly from YouTube

Launch YouTube command

- Use Case: User launches YouTube by saying something similar to "Open YouTube":
- Deeplink: https://www.youtube.com/tv?launch=voice

Direct Voice Search/Playback on YouTube

- Use Case: User deeplinks directly into YouTube by issuing a voice query such as "Search for / Show me / Play yoga videos on YouTube"
- URL: https://www.youtube.com/tv?launch=voice&vq=search+for+yoga+videos+on+YouTube
- Note: the resulting in-app handling of these queries by YouTube may differ based on how the user made the request, e.g. "Play X on YouTube" may be handled by directly taking the user to a video where as "Search for X on YouTube" may be handled by taking the user to the YouTube search results page. The deeplink URL structure that should be sent, however, is the same.

Voice Search when YouTube already in the Foreground

- Use Case: YouTube is already open and in the foreground and user issues a voice search by saying something like "Show me videos from Ryan Toys Review"
- URL: https://www.youtube.com/tv?inApp=true&vq=show+me+videos+from+ryan+toys+review
- Note: Since the user is already actively using YouTube, do not omit the existing "launch=" parameter. inApp=true must be added after the fragment identifier.

This information is non-normative.

10.0 Performance

Note 1: YouTube reserves the right to review the performance of any YouTube Main Living Room 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.
Note 2: Performance targets are the results of both Software and Hardware requirements.

This information is non-normative.

10.1 Multiple App Environment

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

10.1.2 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.

10.2 Loading time

10.2.1 The target device **MUST** be able to load the YouTube application in 9 seconds, measuring the 95th percentile duration value from the time the user selects the YouTube application for launch to the time the browser finishes rendering the YouTube application and factoring out network transfer times.

Latency should be measured when the Cobalt process has not been started.

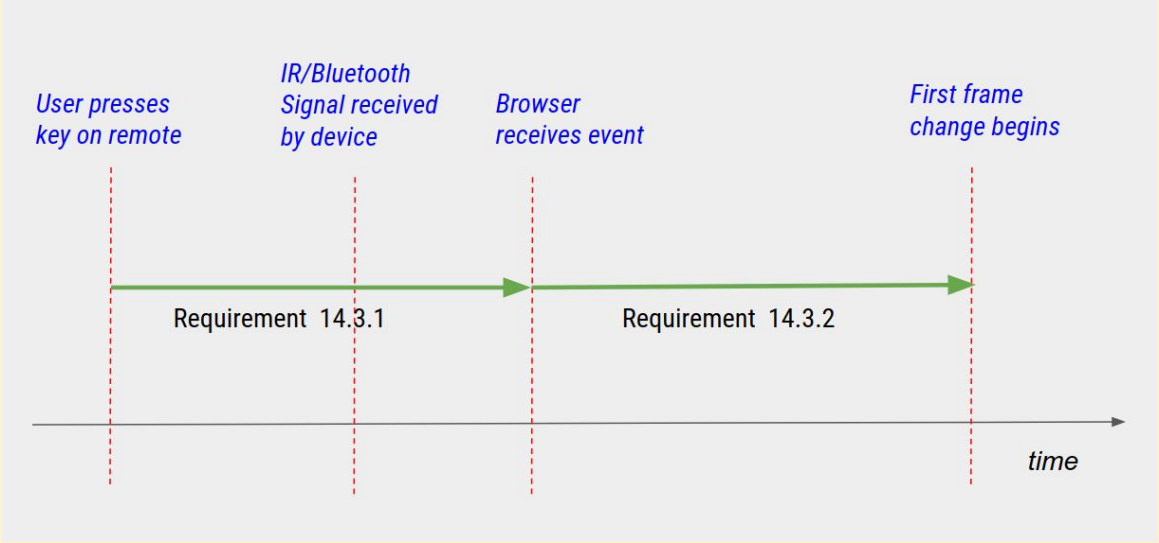
This information is non-normative.

10.3 Latency on User input

10.3.1 The Remote latency (the time between a user initiating an event and the Browser receiving the event) **MUST NOT** exceed 150 milliseconds.

10.3.2 The Event Handling and Initial Animation latency (the time between the Browser receiving an event and the Browser completing the rendering of the first frame) **MUST NOT** exceed 200 milliseconds.

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



This information is non-normative.

10.4 Transition times

10.4.1 The target device **MUST** be able to complete a browse-to-watch transition within 1.5 seconds, measuring from the time the Browser receives the event to the time the video content

begins playback and factoring out network transfer times.

Transition times should be measured on non-live, non-encrypted videos that are also not an Ad

This information is non-normative.

10.5 UI Frame rate

10.5.1 The UI frame rate **MUST** be equal or greater than 30fps for 95% of the time.

10.6 Video endurance

10.6.1 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

10.7 Overall endurance

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

11.0 Speech Functionality

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

11.1 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.

11.1.1 If speech functionality is possible using an available connected device, `enumerateDevices()` **MUST** return at least one media device with `kind` attribute "`audioinput`."
11.1.2 The device **MUST NOT** create persistent data storage for any of the media device with `kind` attribute "`audioinput`." that `enumerateDevices()` returns.
11.1.3 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.
11.1.4 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.
11.1.5 The `devicechange` event **MAY** be triggered when the set of media devices returned by `enumerateDevices()` changes.

***Implementation note*:**

The platform defines SB_HAS_MICROPHONE to enable the MediaDevices API and the MediaStream Recording API.

- MediaDevices.EnumerateDevices() will call in to SbMicrophoneGetAvailable() to determine which microphone devices are available.

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.deviceId);
  });
})
```

might return

```
GNZqLVbbgQNXByj+XQNLdmXlSOYkETYCIxnbAD0PGvM=
fV1lGrr2/IyZunKOma5wTOvCk8XCndWfZ36xw1xUPm0=
```

This information is non-normative.

11.2 The target device **MUST** implement at least one of the following two APIs if input speech functionality is provided and supported on the device:

- SpeechRecognition API

***Implementation note*:**

- The platform defines SB_HAS_SPEECH_RECOGNITION if it supports speech recognition, which will enable the SpeechRecognition API.
- Instructions for enabling voice search in Cobalt via SB_HAS_SPEECH_RECOGNITION: https://cobalt.googlesource.com/cobalt/+/master/src/cobalt/doc/voice_search.md

- MediaRecorder API

***Implementation note*:**

- A MediaRecorder object can then be constructed from a MediaDevices.getUserMedia() call, after which the MediaRecorder object will act as the web app's interface to the SbMicrophone implementation
- Details of the API are available at https://www.w3.org/TR/mediastream-recording/

11.3 If support for Speech Synthesis is required, then the target device **MUST** properly implement the SpeechSynthesis Interface from the Web Speech API Specification, 19 October 2012.

***Implementation note*:**

The platform defines SB_HAS_SPEECH_SYNTHESIS 1 and implements the SbSpeechSynthesis Starboard interface to enable the SpeechSynthesis web interface on Cobalt

11.4 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.

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

12.0 In-application and Universal Search

12.1 From anywhere on the target device and its peripherals, if the user's text or voice query includes the use of a "condition" (e.g "...on YouTube", "...on YouTube Kids", "...on YouTube TV", etc.), or any self-contained search query including "YouTube", such query **MUST** be passed on to the YouTube on TV 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.

12.2 If the user's query is a non-media request, the device assistant **MAY** handle the 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.

12.3 Otherwise if the query is a media request, and the user issued the query

- **12.3.a** …Within the application..: the query **MUST** be directly handed off to the application for fulfillment 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.
- **12.3.b** …As text or voice outside the application...: the device assistant **MAY** determine if there is any service with an "exclusive" match
    - If an exclusive match is identified:
        - The device platform **MAY** deeplink the user into the relevant service's result(s) page for the identified exclusive match
    - If, however, there is no exclusive match for the user's query, the device assistant:
        - **MUST** use the YouTube data API to request results from YouTube as documented here : https://developers.google.com/youtube/v3/docs/search
        - **MUST** display all returned YouTube API results solely on an accessible and dedicated YouTube shelf as part of the device platform's "universal search" page; fair ranking of the shelves is expected (e.g. based on popularity, user preferences, frequency of use).
        - **MUST** comply with the YouTube API Services Terms of 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.

When a YouTube user submits a voice query to YouTube, said YouTube app can have either one of the following states: Launched and running in the foreground, Launched and running in the background, or, Not launched.

This information is non-normative

13.0 High Dynamic Range

This section applies only to devices supporting High Dynamic Range, that is devices belonging to either the “>FHD” or the “>4K” category . 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.

13.1 The isTypeSupported method **MUST** support the 'vp9.2' codec string.
13.2 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 .

***Implementation note*:**

- Platforms that indicate support for HDR (via isTypeSupported) will receive valid data in the the |color_metadata| field of the SbMediaVideoSampleInfo struct, and they must handle this properly.

13.3 EOTF

13.3.1 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

13.3.2 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.

13.4 Color Spaces

13.4.1 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.

13.4.2 The device **MUST** produce subjectively acceptable visual behavior across input covering the full gamut of Rec. 2020.

13.4.3 Explicitly-signaled color spaces **MUST** be respected and properly decoded. If no color space is signaled, the device **MUST** default to Rec. 709.

13.5 HDMI

12.5.1 If the device has an HDMI output, the device **MUST** propagate the HDR metadata conveyed in the Matroska or MP4 container into HDMI *InfoFrames*.

13.6 Post-processing

12.6.1 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.

14.0 On-Screen Keyboard Support

14.1 The platform **MUST NOT** support a native on-screen keyboard.

***Implementation note*:**

- The platform needs to define SB_HAS_ON_SCREEN_KEYBOARD to 0.

15.0 Accessibility

15.1 If the target device supports high contrast settings. It **MUST** notify the YouTube application of any changes in the settings.

***Implementation note*:**

- The kSbEventTypeAccessiblitySettingsChanged event should be generated if the settings are modified while Cobalt is running

15.2 If the target device supports closed captions system settings, it MUST notify the YouTube application of any changes in the settings.

***Implementation note*:**

- The platform can enable this by defining SB_HAS_CAPTIONS.
- The platform must generate a kSbEventTypeAccessibilityCaptionSettingsChanged event anytime system caption settings change.

16.0 Device Maintenance

16.1 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.
16.2 The target device **MUST** remain in compliance with Software Requirements for CE Devices specification for a period of three calendar years after the target device is first made available to users. More specifically, the Cobalt run-time **MUST** be updated to the latest LTS version (See section 0.1 above) at least once a year and for a period of three years after its first availability on the market.

YouTube partners should expect that Cobalt will include a feature to support automatic updates in the very near future. With that feature, partners will be able to rely on a YouTube updater service to have their devices upgraded to a newer version of Cobalt runtime periodically.
For partners who elect to continue relying on manual updates, and in addition to requirement 16.2 above, there will be a new requirement to carry out and deploy at least one Cobalt bug fix patch update every year.
If needed, an amendment will be issued to formalize these new requirements for Cobalt runtime updates.

This information is non-normative.

16.3 Any firmware update planned for the target device **MUST** be communicated to YouTube at least 12 calendar days before the effective first day of the firmware deployment.

17.0 End User Privacy

17.1 The Partner **MUST** comply with all privacy laws and regulations, including those applying to Google's end user login information and data.
17.2 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.

18.0 Internationalization

18.1 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.

18.2. 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

- 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

- HTML5: https://www.w3.org/TR/html5/
- Web Storage Specification: http://dev.w3.org/html5/webstorage/
- 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/
- Selectors Level 4: https://www.w3.org/TR/selectors-4/
- 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
- 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://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.