Media Capabilities API Issues and PRs

Chris Needham, W3C Media WG
Last Updated: 12 December 2023

PR #78 / Issue #79 Define the meaning for CBR and VBR more precisely

Proposed change:

The *bitrate* member represents the**[-average-]** bitrate of the video track given in units of bits per second. In the case of a video stream encoded at a constant bit rate (CBR) this **[-value should be accurate over a short term window.-]{+shall represent the average bitrate of the video track.+}** For the case of variable bit rate (VBR) encoding, this value **{+shall represent the maximum bitrate of the stream. In either case, this value+}** should be usable to allocate any necessary buffering and throughput capability to provide for the un-interrupted decoding of the video stream over the long-term based on the indicated `contentType`.

PR #105 Use "parse a MIME type" to check validity

- Related issue: #69 Use mimesniff's "parse a MIME type" to parse the contentType member
- Requirements for authors: MIME type should indicate a single audio or video codec
- Simplify algorithm language, e.g., “substeps”, and “abort these steps” are unnecessary
- Change wording from “imply a single media codec”
- Need another word/phrase for “valid media MIME type”
    - See mimesniff definition of “valid MIME type string”
- Needs tests - see WPT

PR #186 Add a new webrtcCodec parameter to MediaCapabilitiesInfo

- Related issue: #185 Retrieving RTCRtpCodecCapability from MediaCapabilities when queried for webrtc
- Adds `webrtc.audio` and `webrtc.video` objects to `MediaCapabilitiesInfo`
- Discussion continued since PR was last updated
  - Add steps to clarify how construct the `RTCRtpCodecCapability`
  - Should clockRate be required or optional?
  - Do we want defaulting rules? If I do not provide channel, or I just provide 'video/H264' or just 'video/H264'+profile-level-id, can I still get a valid capability dictionary?
- Needs follow up from Youenn, Bernard, Harald
- Needs tests - see WPT

PR #165 Define codecSwitchingSupported

- Related issue: #102 Discuss `transition()` ergonomics
- Introduces the idea of a “Decoding Pipeline”
- Describes capabilities for MSE `SourceBuffer changeType()`
- Adds `codecSwitchingSupported` to `MediaCapabilitiesDecodingInfo`
    - Alternative to previously suggested API shapes:
        - `mediaCapabilitiesInfo.transition()`
        - `mediaCapabilities.decodingTransitionInfo()`
- Agreed that `smooth` flag in `MediaCapabilitiesDecoding` info does not describe the transition itself. It describes the minimum performance of playback for the given configurations
- Open questions:
    - Reporting seamless transition capability, e.g., timing tolerance
    - How do EME pipelines affect transition capability? Does the API adapt to what is currently in use or playing?

#102 Discuss transition() ergonomics

Proposal 1: Add `MediaCapabilitiesDecodingInfo.transition()` method

```
// Query an initial decoding configuration
const info = await navigator.mediaCapabilities.decodingInfo(...)

if (info.supported) {
  // Query a second decoding config, result shows
  // if the combination is supported, smooth, power efficient
  const transition = await info.transition(...);

  console.log(transition.supported, transition.smooth, transition.powerEfficient);
}
```

#102 Discuss transition() ergonomics

Proposal 2: Add `decodingTransitionInfo()` method

```
const config1 = { ... };
const config2 = { ... };

// Query an initial decoding configuration
const info = await navigator.mediaCapabilities.decodingInfo(config1);

// Query a second decoding config, result shows
// if the combination is supported, smooth, power efficient
const transition = await.navigator.decodingTransitionInfo(config1, config2);

console.log(transition.supported, transition.smooth, transition.powerEfficient);
```

#102 Discuss transition() ergonomics

Proposal 3 (PR #165): Add `codec_transitions_supported` flag to `MediaCapabilitiesDecodingInfo`

```
const config1 = { ... };
const config2 = { ... };

// Query decoding configurations
const info1 = await navigator.mediaCapabilities.decodingInfo(config1);
const info2 = await navigator.mediaCapabilities.decodingInfo(config2);

// Check both are supported, including transitions
return (info1.supported && info2.supported) &&
       (info1.codecSwitchingSupported && info2.codecSwitchingSupported);
```

PR #107 Mark keySystemAccess as default to null and optional and robustness properties as no longer defaulting

- This PR is about removing defaults from

  ```
  DOMString audioRobustness = "";

  DOMString videoRobustness = "";
  ```

  and making MediaCapabilitiesDecodingInfo keySystemAccess optional, defaulting to null.

- Looks to be superseded by current spec text for MediaCapabilitiesKeySystemConfiguration and valid MediaConfiguration - propose closing

#209 Align exposing scalabilityMode with the WebRTC "hardware capabilities" check

- Problem #1: WebRTC-SVC uses Media Capabilities API for discovery
  - Indicates if a configuration is “supported” “powerEfficient” or “smooth”
  - Media Capabilities API not limited to capture context
- Problem #2: SVC rarely supported in hardware
  - Today, few devices support “powerEfficient” SVC
  - Simulcast often preferred to SVC to save power
  - Result: `scalabilityMode` support of little value for hw fingerprinting
- Problem #3: WebRTC-SVC exposes less information than Media Capabilities
  - Calling `RTCRtpSender.setParameters()` or `addTransceiver()` with `RTCRtpEncodingParameters.codec` exposes whether configuration is “supported”, but not “powerEfficient” or “smooth”
- Proposal: (From November 21 WebRTC WG meeting) Limit exposure of power efficient / smooth for `scalabilityMode` to capture context only ?
- See also issue #176 General approach to capability negotiation

#176 General approach to capability negotiation

- PING question: Why expose device capabilities to the app for purposes of negotiation? Couldn't we instead have sites expose available media formats and have browsers (perhaps in a way not exposed the application) pick the one they like best?
- pes10k: Spec needs normative protections against fingerprinting risk
- See Security and Privacy Questionnaire

#203 Browser interop issues

- MediaCapabilities.encodingInfo() type “webrtc” vs “transmission”. Chrome and Safari use “webrtc”, Firefox uses “transmission”
- Safari has special behaviour to show `supported: true` and add a `supportedConfiguration` object to the result. `scalabilityMode` parameter is ignored, see webrtc/samples#1596. Should we spec `supportedConfiguration`?
- Chrome >= 101 reports `supported: true` for type “webrtc” and `scalabilityMode` parameter. But SVC is only supported in Chrome >= 111, see webrtc/samples#1597. This is a browser bug where the MediaCapabilities would report that the encoders are technically able to do SVC but the WebRTC encoders are not able to be configured for SVC.

#202 What is the interaction of media capabilities with WebCodecs?

- WebCodecs has `isConfigSupported()`
- If Media Capabilities indicates that a config will be smooth for WebRTC, does that necessarily imply that it will be smooth for WebCodecs?

#197 Provide an example using MediaCapabilitiesKeySystemConfiguration

- Adding an example is useful

#141 Conformance against enumerated assertions in VideoConfiguration

- Reference conformance points alongside each of the technical specs involved in VideoConfiguration properties (e.g., HdrMetadataType, ColorGamut and TransferFunction)
- For HDR metadata types defined by SMPTE-ST-2086, SMPTE-ST-20894, even if a device can interpret the metadata, the screen may not actually display it. Similarly, it may accept BT.2020 color data and then map everything to 709.
    - This should be solved by CSS media queries

#136 DolbyVision HDR metadata

- Discussed at TPAC 2023. Current proposal is to add a “dvmd” identifier to HdrMetadataType, and move this to a registry.
- Has this come up in other SDOs, e.g., MPEG? Anything we can reference?

#133 Should VideoDisplayConfiguration width & height be in CSS or device pixels

- Propose closing, as Media Capabilities doesn’t include display capabilities
- CSS issue #6891: Expose video plane pixel ratio
- Related issue: #135 hdrSupported: Screen.video or simply Screen?

#113 Multiple stream decoding

- A number of media devices support decoding more than one stream at one time:
  - Some devices may have multiple decoders with identical capabilities
  - Some devices may be able to decode one UHD stream and one HD stream at any time - i.e. the second decoder simply doesn't support UHD ever but can be used at any time regardless of what the first decoder is doing
  - Some devices may have dynamic capabilities, e.g., they can decode one UHD or two HDs at the same time but not two UHDs. This may be due to the amount of available RAM, or memory bandwidth
- Using multiple media decoders can improve client-side advertising use-cases
- Previous answer: Multiple stream is out of scope because it would be hard to give a reliable answer. Related: issue #102 (transition API)

#99 Support for MPEG CMAF Supplemental Data

- CMAF Amendment 2 (now included in ISO/IEC 23000-19:2020) introduces Supplemental data brands, indicating the presence of additional information in a track that is not required to render playback of the track – e.g. NAL units or SEI messages. Two examples:
  - ‘ccac’ indicating that 608/708 captioning is present
  - another 4CC indicating SEI delivered dynamic HDR metadata which can be applied to an underlying 10-bit CMAF video profile
- Media Capabilities has HdrMetadataType ("smpteSt2086", "smpteSt2094-10", "smpteSt2094-40"). Is more needed?
- Media Capabilities does not included captioning. Should it?

#98 Support for MIME Type Profiles subparameter in order to support CMAF, etc.

- @johnsim: It would be desirable for Media Capabilities to be able to indicate support for CMAF media profiles
- CMAF media profiles are identified by 4cc codes which are in the CMAF init segment and characterize the content. The media profile includes bit depth, chroma subsampling, color space, transfer functions, black level – which I believe cannot be expressed in the codecs string
- Proposed resolution from 3 Nov 2020 MEIG meeting was: if Media Capabilities allows the bit depth, color space, transfer functions, etc, to be queried, CMAF profiles can be mapped to this in a JS library. Are there any remaining gaps?

#95 Frame rate configuration

- The current spec uses framerate as a video configuration parameter. This assumes the video has a fixed frame rate. What about variable frame rate, even if less used?
    - “The framerate member represents the framerate of the video track. The framerate is the number of frames used in one second (frames per second)”
- Chris C: Implementers should answer as if everything is fixed-rate, and web authors should query with whatever number they think best describes their stream (taking into account the distribution of frame rate in the video)
- Proposal: Add a note about variable frame rates

#88 Dictionary pattern may be incompatible with WebIDL

- WebIDL issue #76: Sort out when dictionaries and records should have default values
- Is there anything we need to change in Media Capabilities?

#73 Better define “channels” in AudioConfiguration

- It's currently hand-wavy and inconsistent with Web Audio. We should decide whether we want to keep the definition as including the sub (ie. .1) or if we want to stay fully consistent with Web Audio
- Inline spec issue: The channels needs to be defined as a double (2.1, 4.1, 5.1, ...), an unsigned short (number of channels) or as an enum value. The current definition (DOMString) is a placeholder
  - What do current implementations do?
- Needs to be clear about whether multi-channel audio will be downmixed to stereo. Some content providers may wish to provide their stereo audio track rather than rely on an unknown downmix
- Web Audio: unsigned long numberOfChannels

#44 powerEfficient needs some objective definition

- The powerEfficient attribute currently is open ended, so it would be difficult to have multiple UAs respond similarly
- Current wording: *If the UA is able to encode the media in a power efficient manner, set powerEfficient to true. The user agent SHOULD NOT take into consideration the current power source in order to determine the power efficiency unless the device's power source has side effects such as enabling different encoding/decoding modules.*
- Proposed wording: *Decoding is power efficient when the power draw is optimal or close to optimal. The definition of optimal power draw from decoding is left to the UA but common implementation strategies would be to consider hardware decode as optimal. It is NOT RECOMMENDED to only mark as power efficient hard decoding as the power draw of non-accelerated codecs can sometimes be very close to accelerated ones (for example, low resolution videos).*

#43 Need a mechanism for considering device conditions that affect media capabilities

- Make Media Capabilities take into account device state such as:
  - EME keySystem (added in #101)
  - Battery status
  - Whether Remote Playback is intended (issue #26)
  - Others (?)
- Proposal to add a `MediaConditions` object to `MediaConfiguration` to describe such state
- Main question remaining:
  - Battery status - is `powerEfficient` enough?
  - Media Capabilities and Remote playback API

#3 Deprecate isTypeSupported and canPlayType

- Does the API make promises for only HTMLMediaElement and MediaSource, or does it also cover Web Audio `decodeAudioData`?

Developer proposals

#196 Proposal: canPlay (a local file) API

- Proposal is to add an API, where given a media file, returns whether the browser can play it. Alternative to providing MIME type details.

#194 What about still images?

- Proposal is expose display capabilities: “the key capability that matters is the display itself, which influences both video and still images”