MEDIA WG Meeting

April 11, 2023

WebCodecs Issues/PRs for Discussion

- Issue 656: Allow decoder to ignore corrupted frames
- Issue 646/PR 662: Configuration of AV1 screen content coding tools
- Issue 619/PR 654: Extend EncodedVideoChunk metadata for SVC

Issue 656: Allow decoder to ignore corrupted frames

- WebCodecs specifies that decoder (and encoder) errors are fatal.
  - VideoDecoder.decode(chunk)
    - 2. If decoding results in an error, queue a task to run the Close VideoDecoder algorithm with `EncodingError` and return.
  - VideoEncoder.encode(chunk)
    - If encoding results in an error, queue a task to run the Close VideoEncoder algorithm with `EncodingError` and return.
  - AudioDecoder.decode(chunk)
    - 2. If decoding results in an error, queue a task to run the Close VideoDecoder algorithm with `EncodingError` and return.
  - AudioEncoder.encode(chunk)
    - 2. If encoding results in an error, queue a task to run the Close AudioEncoder algorithm with `EncodingError` and return.

Issue 656: Allow decoder to ignore corrupted frames (cont'd)

- GitHub discussion
  - Dale: Can clarify the text to distinguish between fatal and non-fatal errors. For security reasons, all errors in Chromium's software decoder are fatal. Doesn't interfere with resilience.
  - sefless: do we have/need tests? Dale: we have some tests.
  - Matanui159: Instead of EncodingError, Chromium reports an OperationError with error.message = "Decoding error".
- Editor's discussion (captured by Padenot)
  - Should all errors be fatal?
  - If hardware resources are limited and there is a decoder error, can't another application acquire the resources?
  - What is the difference between reset() and close() followed by constructing a new encoder/decoder in terms of performance? Does it affect user experience?

Issue 656: Allow decoder to ignore corrupted frames (cont'd)

- Other questions
  - Should implementations be using EncodingError or OperationError?
  - Is enough information available to allow the application to figure out what to do next?
    - Attempt to construct a new encoder/decoder configured with "prefer-hardware"?
    - Try "prefer-software" instead (e.g. hw resources unavailable)?

Issue 646/PR 662: Configuration of AV1 screen content coding tools

- Issue 646: Currently the AV1 registry doesn't allow configuration of screen content coding tools.
- PR 662 adds `boolean forceScreenContentTools`
  - Default is `false`.
  - Setting `forceScreenContentTools` to `true` causes seq_force_scrren_content_tools to be set to SELECT_SCREEN_CONTENT_TOOLS (see: [AV1] Section 5.5.1).

Issue 646/PR 662: Configuration of AV1 screen content coding tools

```
98     <xmp>
99     dictionary VideoEncoderEncodeOptionsForAv1 {
100      unsigned short? quantizer;
101  +   boolean forceScreenContentTools = false;
102    };
103    </xmp>
104    </pre>


111      </dd>
112    </dl>
113
114  + <dl>
115  +   <dt><dfn dict-member
       for=VideoEncoderEncodeOptionsForAv1>forceScreenContenttools</dfn></dt>
116  +   <dd>
117  +     Indicates whether the encoder should force use of screen content
118  +     coding tools. The default value (false) indicates that use of
119  +     screen content coding tools is not forced. A value of true
120  +     (corresponding to setting seq_force_screen_content_tools
121  +     to SELECT_SCREEN_CONTENT_TOOLS in Section 5.5.1
122  +     of [[AV1]]) indicates that use of screen content coding tools
123  +     is forced.
124  +   </dd>
125  + </dl>
```

Issue 619/PR 654: Extend EncodedVideoChunk metadata for SVC

- Issue 619: Consistent SVC metadata between WebCodecs and Encoded Transform API.
  - WebRTC-SVC has shipped in Chromium as of M111. Supports both temporal (VP8, VP9, AV1, H.264) and spatial scalability (VP9, AV1)
  - Encoded Transform API provides RTCEncodedVideoFrameMetadata.
  - Currently, WebCodecs provides an EncodedChunkMetadata dictionary
    - Includes an svc sub-dictionary that currently only supports temporal scalability.
- Guidance from the MEDIA WG
  - Develop a PR for extensions to the EncodedChunkMetadata dictionary.

Issue 619: Extend EncodedVideoChunk metadata for SVC

- RTCEncodedVideoFrameMetadata.

```
dictionary RTCEncodedVideoFrameMetadata {
    unsigned long long frameId;
    sequence<unsigned long long> dependencies;
    unsigned short width;
    unsigned short height;
    unsigned long spatialIndex;
    unsigned long temporalIndex;
    unsigned long synchronizationSource;
    octet payloadType;
  sequence<unsigned long> contributingSources;
};
```

- WebCodecs EncodedChunkMetadata

```
dictionary EncodedVideoChunkMetadata {
  VideoDecoderConfig decoderConfig;
  SvcOutputMetadata svc;
  BufferSource alphaSideData;
};

dictionary SvcOutputMetadata {
  unsigned long temporalLayerId;
};
```

Original WebCodecs SVC Schema

Things to add to WebCodecs EncodedVideoChunk: basically all of the https://aomediacodec.github.io/av1-rtp-spec/#a8-dependency-descriptor-format RTP header extension.

```
dictionary EncodedVideoChunkMetadata {

// Number for identifying this frame in |dependsOnIds| and |chainLinks| (for other chunks).
unsigned short frameNumber;

// List of frameNumbers that this chunk depends on.
// Used to detect/handle network loss. Decoding out of order is an error.
list<unsigned long> dependsOnIds;

// ID of the spatial layer this chunk belongs to.
unsigned long spatialLayerId;

// ID of the temporal layer this chunk belongs to.
unsigned long temporalLayerId;

// List of decoder targets this frame participates in.
// Used to know whether this frame should be sent (forwarded) to a given receiver depending
// on what decode targets the receiver is expecting.
// Decode target is a numerical index determined by the encoder. No commitment that a
// particular number implies a given layer.
list<unsigned long> decodeTargets;

// Mapping of decode target -> the last important frame to decode prior to "this"
// frame for the given decode target.
// Used to ensure we preserve decode order for the desired decode target. It is insufficient
// to simply satisfy the dependencies for the current frame. See example.
map<unsigned long, unsigned long> chainLinks;

};
```

PR 654: Extend EncodedVideoChunk metadata for SVC

- GitHub discussion
  - What is the difference between `frameId` and `frameNumber`?
    - `frameId` uniquely identifies the frame within the video sequence.
    - The AV1 Dependency Descriptor uses `frameNumber` since it is only refers to the last 2^16 frames and is therefore smaller.
  - Why the difference in the type of dependencies?
    - The AV1 Dependency Descriptor uses `frameNumber` to describe dependencies, rather than `frameId`. This explains the difference in the types.

PR 654: Extend EncodedVideoChunk metadata for SVC

- Other questions?
  - Why do you need decode targets?
    - A forwarder needs a quick way to determine whether a frame will be decodable by a participant based on the resolution/framerate it is receiving.
  - Why do you need chainLinks?
    - To be decodable, it is necessary (but not sufficient) for an EncodedVideoChunk's dependencies to have been received.
    - The dependencies must also have been decodable.
      - Dependencies also have dependencies.
      - Requiring receivers to compute a dependency graph on the fly consumes CPU and memory unnecessarily.
      - More efficient to let WebCodecs encoder do the work, once.
    - chainLinks provide a way for a receiver to quickly determine whether a receiver can submit an encoded chunk to the decoder (whether it will be decodable), if previous frames were lost.

PR 654: Extend EncodedVideoChunk metadata for SVC

- Modified proposal for the svc sub-dictionary:

```
dictionary svcOutputMetadata {
    unsigned long temporalLayerId;
    unsigned long spatialLayerId;
    unsigned long frameNumber;
    sequence <unsigned long> dependencies;
    sequence <unsigned long> decodeTargets;
    map <usigned long, unsigned long> chainLinks;
}
```