

W3C WebRTC WG Meeting

May 25, 2021

8:00 - 10:00 AM Pacific Time

Chairs: Bernard Aboba

Harald Alvestrand

Jan-Ivar Bruaroey

W3C WG IPR Policy

- This group abides by the W3C Patent Policy <https://www.w3.org/Consortium/Patent-Policy/>
- Only people and companies listed at <https://www.w3.org/2004/01/pp-impl/47318/status> are allowed to make substantive contributions to the WebRTC specs

Welcome!

- Welcome to the May 2021 interim meeting of the W3C WebRTC WG!
 - During this meeting, we will cover displayMediaCaptureHandle, Screenshot API, conditional audio suppression, region capture, webrtc-encoded-transform and mediacapture-transform.

About this Virtual Meeting

- Meeting info:
 - https://www.w3.org/2011/04/webrtc/wiki/May_25_2021
- Link to latest drafts:
 - <https://w3c.github.io/mediacapture-main/>
 - <https://w3c.github.io/mediacapture-image/>
 - <https://w3c.github.io/mediacapture-output/>
 - <https://w3c.github.io/mediacapture-screen-share/>
 - <https://w3c.github.io/mediacapture-record/>
 - <https://w3c.github.io/webrtc-pc/>
 - <https://w3c.github.io/webrtc-extensions/>
 - <https://w3c.github.io/webrtc-stats/>
 - <https://w3c.github.io/mst-content-hint/>
 - <https://w3c.github.io/webrtc-priority/>
 - <https://w3c.github.io/webrtc-nv-use-cases/>
 - <https://github.com/w3c/webrtc-encoded-transform>
 - <https://github.com/w3c/mediacapture-transform>
 - <https://github.com/w3c/webrtc-svc>
 - <https://github.com/w3c/webrtc-ice>
- Link to Slides has been published on [WG wiki](#)
- Scribe? IRC <http://irc.w3.org/> Channel: [#webrtc](#)
- The meeting is being recorded. The recording will be public.
- Volunteers for note taking?

Issues for Discussion Today

- 08:10 AM - 08:35 AM Capture Handle (Elad Alon) [#166](#)
- 08:35 AM - 08:40 AM HC: Counter-Proposal (Jan-Ivar / Elad)
- 08:40 AM - 08:50 AM Conditional Audio Suppression (Elad) [#161](#)
- 08:50 AM - 09:10 AM Region Capture (Elad) [#158](#)
- 09:10 AM - 09:25 AM WebRTC encoded-transform (Youenn)
- 09:25 AM - 09:55 AM Media-capture transform (Harald)
- 09:55 AM - 10:00 AM Wrap-up and Next Steps

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

Capture Handle - Exposition

- Introducing VC-MAX, your one stop shop for all your fictional-VC needs.
 - Fictive video-conferencing software specializing in illustrating points.
 - FaaS - fiction as a service.
 - Any resemblance to real software project, living or dead, is purely coincidental.
- VC-MAX exposes a “Share” button that calls getDisplayMedia and transmits the resulting tracks to remote participants.
- VC-MAX wants to collaborate with various presentation software. When the user chooses to share MS ForcePoint, OpenOffice-365 or Doodle Slides, VC-MAX would like to embed user-facing controls for navigating the presentation - previous slide, next slide, etc.

Capture Handle - Puzzle Pieces

- Assume MS ForcePoint, OpenOffice-365 and Doodle Slides all define their own APIs for navigating a presentation, subject to appropriate authentication, access-control, etc.
 - The shapes of these APIs are out of scope.
 - Virtually guaranteed that the API includes some **session ID**.
- VC-MAX needs to discover:
 - The captured application's identity. (ForcePoint? Doodle?)
 - The session ID.
- Our **discussion's scope** is discovery of these generic bare essentials. Their use is left as an exercise to the reader. :-)

Capture Handle - Hacky Approach

What could we do with no changes to the browser?

- The presentation software could embed a QR code¹.
- VC-MAX could scan for that QR code.

Problems?

- Unergonomic.
- Inefficient. (VC-MAX must continuously scan for the QR code and remove the user-facing controls if no longer capturing the presentation.)
- Can be disrupted by other documents drawing to the screen.
- Reader of content needs to verify all of it (none of it is mediated by the UA).

(1) QR code used to keep the illustration simple. Similar solutions yield similar problems.

Capture Handle - Suggested Solution

On the captured app:

- Allow opt-in exposure of the origin.
- Allow a custom handle to be set.
 - Let it be a string. That string could be a serialized JSON carrying an arbitrary amount of data.
- Allow the captured app to specify which origins may read the values it exposes.
- Call these the **capture-handle**.

On the capturing app:

- Allow checking the current capture-handle.
- Allow registering a handler for capture-handle-update events.

Capture Handle - Suggested Solution (Captured App)

```
// Code on Slides-3000
navigator.mediaDevices.setCaptureHandleConfig({
  exposeOrigin: true,
  handle: JSON.stringify({
    description: "See slides-3000.com for our API description. " +
      "It supports such amazing actions as nextSlide, " +
      "prevSlide, goFullScreen, etc.",
    sessionId: mySessionId(),
  }),
  permittedOrigins: {"*"},
});
```

Capture Handle - Suggested Solution (Capturing App)

```
// Code on VC-MAX
const mediaStream = await navigator.mediaDevices.getDisplayMedia();
const captureHandle =
    mediaStream.getVideoTracks()[0].getSettings().captureHandle;
if (captureHandle.origin == "preso.slides-3000.com") {
    const sessionId = JSON.parse(captureHandle.handle).sessionId;
    Let commsChannel = ...; // Specific to Slides-3000.
    commsChannel.postMessage(sessionId, "goFullScreen");
    ...
} else if (captureHandle.origin == "encounter.doodle.com") {
    ...
}
```

Capture Handle - Interesting Note

Application don't actually have to expose their origin. If the **session ID** is sufficiently unguessable, a separate cloud API could be set up exposing **validate()**.

```
const captureHandle =
  mediaStream.getVideoTracks()[0].getSettings().captureHandle;
if (slides3000CloudBasedHelper.validate(captureHandle)) {
  ...
}
```

If a site exposes its origin, though, a round-trip to the identification service could be saved.

Note that the captured site chooses to allow exposing the origin; the browser sets the actual value to be read on the capturing side.

Capture Handle - IDL

Captured-side

```
dictionary CaptureHandleConfig {
    boolean exposeOrigin = false;
    DOMString handle = "";
    sequence<DOMString> permittedOrigins = [];
};

partial interface MediaDevices {
    void setCaptureHandleConfig(
        optional CaptureHandleConfig config = {});
};
```

Capturing-side

```
dictionary CaptureHandle {
    DOMString origin;
    DOMString handle;
};

partial dictionary MediaTrackSettings {
    CaptureHandle captureHandle;
};

partial interface MediaStreamTrack {
    attribute EventHandler oncapturehandlechange;
};

[Exposed=Window]
interface CaptureHandleChangeEvent : Event {
    constructor(CaptureHandleChangeEventInit);
    [SameObject] readonly CaptureHandle captureHandle;
};
```

Capture Handle - Stop Worrying, Love the Bomb.

Privacy/Security concerns? Probably not.

- Opt-in
- User-driven
- Theoretically no-op (see QR-hack - already possible!)
- More robust than steganography
- Captured app only discovers it's captured if capturer informs it
- App-controlled handle-opaqueness (or possibly “opacity”)
- Non-spoofable origin-exposure
- Excessive events not a viable attack

getDisplayMedia() & getViewportMedia() in 2023 (jib)



All major presentation websites use **site-isolation** and opt into **html capture**.

They can capture themselves using `getViewportMedia()` with permission.

- Use case: user clicks button in presentation program to join integrated meeting.

They register for preferential positioning and treatment in `getDisplayMedia()`'s new picker, which relegates non-conforming choices under **other...**, using e.g.:

```
const id = await navigator.mediaDevices.registerIntent("presentation");  
// id is exposed in track.getSettings().presentationId in VC website
```

- Use case: user clicks “Present” in VC meeting to find their open presentation
- Use case: user can navigate slides from VC meeting in most presentations.

All major presentation websites use **site-isolation** and opt into **html capture**, because these APIs are not available otherwise. → a **safer web**.

2021 > 2023 (Elad Alon)

The Capture Handle origin-trial in Chrome starts with m92 (2021-07-20)
Looking forward to sharing developer-feedback soon. Would be interesting to hear if devs are happy with Capture Handle, or if they ask for something more along the lines of the Mozilla's proposal.

Some specific reservations against the previous slide's proposal:

- registerIntent(): Apps jockey for position with false promises.
- Browser-assigned IDs:
 - Force an additional step of translation to the app's actual ID space before collaboration starts.
 - Don't allow (opt-in) exposure of the captured tab's origin.

What are the benefits of the previous slide's approach?

Issue 161 / PR164 **Conditional Audio Suppression**

Suppose this meeting weren't virtual. We're all sitting in a room, taking turns presenting from our laptops to a projector and a set of speakers. The next speaker presents a tab. Do they want the audio to flow out of both their laptop's speakers as well as the PA system's? Probably not.

Solution? Define a new constraint - [suppressLocalAudioPlayback](#).

When this constraint is applied, audio from captured source is not played out over the local speakers.

This can also help with echo cancellation on a single-machine setup, as playing the locally captured audio in a VC as though it came from another participant simplifies things for the echo canceller.

suppressLocalAudioPlayback

ConstrainBoolean

As a setting, this value indicates whether or not the user agent is applying [local audio playback suppression](#) to the source.

As a constraint, this value is only meaningful if the user selects capturing a [browser display surface](#). In that case, a value of **true** indicates that the user agent *SHOULD* perform [local audio playback suppression](#) on the captured [browser display surface](#).

When **local audio playback suppression** is applied, the user agent *SHOULD* stop relaying audio to the local speakers, but that audio *MUST* still be captured by any ongoing audio-capturing capture-sessions.

When a [browser display surface](#) is subject to multiple concurrent captures, [local audio playback suppression](#) *SHOULD* be active as long as at least one active audio-capturing capture-session remains which specified [suppressLocalAudioPlayback](#). (Note that capture-sessions for which the user did not permit audio-capture are not considered audio-capturing for this purpose.)

Region Capture - Introduction (Elad alon)

Underlying assumption - getViewPortMedia has been specced.

Apps can now self-capture.

Apps comprise multiple parts. These parts can be cross-origin. And their location on the screen can move.

How does an app crop its video stream? When would it want to?

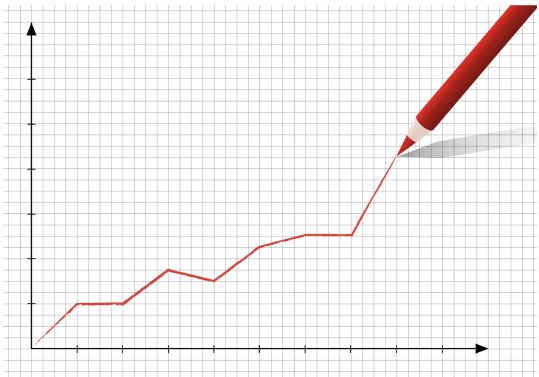

Region Capture - Motivation

Assume this composite app, comprising two separate, collaborating parts.

The presentation software can expose a button for self-capture and sharing of the resulting stream with remote participants.

A performant, ergonomic and secure way of cropping away some content is needed.

Why secure? Because not only the VC part has to be cropped away. Maybe the speaker notes, too. And cropping should be robust to window-resizing, for instance. Not a single mis-cropped frame should be tolerated.

<p>(Presentation)</p> 	<p>(VC)</p> 
<p>Confidential speaker notes. Very embarrassing if glimpsed by remote participants.</p>	

Region Capture - Who Crops?

The browser has to take charge of cropping.

- It is wasteful to produce and shuttle big frames.
- If cropping is carried out by the app, cross-origin iframes cannot guarantee that they would communicate resize events quickly enough to avoid mis-cropped frames (which are a privacy violation).

Region Capture - Who Captures?

Arguments were put forth that getViewportMedia should, by definition, crop to the frame from which it is called. Making tracks transferable, it was argued, would then make this solution complete, allowing the presentation-frame to hand the track over to the VC-frame.

I do not believe that to be a good idea.

It encourages a hacky pattern whereby an invisible/occluded frame lurks in the background, defining the intended crop-region. That frame then invokes getViewportMedia. And that's privacy-violating footgun, on account of window resizing, etc.

A more dev-friendly solution would be allowing an arbitrary capturer to crop to an arbitrary iframe or div. Devs like powerful, flexible tools.

Region Capture - How is the Region Defined?

By cropping to an element known to the browser, we allow the browser to track changes to placement, guaranteeing that all frames are cropped correctly.

How can we specify which element to crop to?

- DOM node references cannot work between cross-origin frames. That's a show-stopper.
- The global attribute "id" is document-global, not page-global. But it's trivially easy to randomly select an ID that is universally unique with very high probability. (If not unique, fail in some sensible way; TBD.)
- A new browser-assigned UUID could be defined. (Assign-on-read will minimize the memory+CPU cost.)

Once assigned/read, the ID can be `postMessage()`-ed to the capturing frame.

Region Capture - Noteworthy Points

- Using the crop-to-ID approach, apps can still crop-to-self and transfer the track to another iframe if they find that pattern useful. This is not an either-or situation.
- Recall that element-capture is a different tool with its own set of trade-offs. Occluded content is captured, occluding content is missing from the capture. Whether this is a bug or a feature depends on the app. Ideally, the Web platform should offer both tools to developers.

Region Capture - API Shape

Crop on capture-start, or crop once the MediaStreamTrack is produced?
Possibly support both flows?

Cropping on capture-start:

- Can be achieved with an additional parameter to getViewportMedia.
- Trivial to know which frame is the first one to be cropped as desired.
- Multiple possibilities for changing the crop-target. [Enumerated verbally.]

Cropping as a control on the track:

- Error if track is not derived from tab-capture of the current tab.
 - Or possibly any tab-captured? Allow a truly universal crop-target? 🤖
- Obvious how to support crop-target change.

Issue 58: SFrame error handling (Youenn)

SFrame processing can generate errors, mostly on decryption side

- Unknown key ID
 - Bad setup, race conditions
 - Application may want to fetch the right key
- Parsing error, authentication error
 - Potential attack by the SFU
 - Application may want to drop the connection

Useful to surface these errors to the JS application

- Should we have a default behavior in case of authentication error?

Issue 58: SFrame error handling (Youenn)

Proposal ([PR103](#))

- Expose an error event handler on SFrameTransform
 - In case SFrameTransform is used standalone
 - In case SFrameTransform is used as part of RTCScriptTransform

```
// main-page.js
const sender = pc.addTrack(track, stream)
sender.transform = new SFrameTransform
sender.transform.onerror = (event) => {
  if (event.type === "keyID")
    handleUnknownKey(event.keyID)
}
```

```
// rtc-script-worker.js
onrtctransform = (e) => {
  const transformer = e.transformer
  const metadataTransform = new MyMetadataTransform
  const sframeTransform = new SFrameTransform
  transformer.readable
    .pipeThrough(metadataTransform)
    .pipeThrough(sframeTransform)
    .pipeTo(transformer.writable);
  sframeTransform.onerror = (event) => {
    if (event.type === "authentication")
      accumulateAuthErrorAndDropConnectionIfNeeded()
  }
}
```

Issue 99: Use WebCodec Encoded Chunk (Youenn)

Two APIs for very similar concepts

- RTCEncodedAudioFrame and RTCEncodedVideoFrame
- EncodedAudioChunk and EncodedVideoChunk

Main differences

- WebCodec uses immutable encoded frames
 - WebRTC encoded transform expect scripts to change frames
 - Data ownership transferred at write step
- RTC specific metadata exposure

Issue 99: Use WebCodec Encoded Chunk (Youenn)

Two options

- Status quo: use two different constructs
 - Keep a transformable frame in WebRTC encoded transform
 - Reuse subtypes and duck taping to maximize common code
- Merge the two APIs
 - Select a common 'data' handling
 - Allow mutation of EncodedXXChunk/Add clone+data method
 - Extend WebCodec frame metadata for RTC
- Proposal: keep status quo
 - Aligned with existing implementations

Mediacapture-transform issues (Harald)

- Main issues under active discussion
 - [#4](#) Is using ReadableStream/WritableStream the right approach?
 - [#23](#) Out-of-main-thread processing by default
 - [#29](#), [#31](#) Is Audio necessary? (suggest leaving alone today)
- Related, but not strictly on this spec
 - Mediacapture-extensions issue #16 / PR #21/#24 - Transfer MST

More items can be found in [last month's slides](#), these are omitted for focus.

#4 Is using ReadableStream/WritableStream the right approach?

I believe the discussion has shown that it is.

- Developers like it
- No significant downsides of using it have been identified

I suggest we adopt this as our strategy for now and move on.

#23 Out-of-main-thread processing by default

Two trains of thought:

- Worker processing should be easy. Main-thread processing should be possible. (Initial proposal)
- Worker processing should be mandatory. Main-thread processing should be hard or impossible. (Proposals for making MSTG/MSTP worker-only)

So far, ~all the arguments have come from 3 browser vendors. We have not come to agreement.

#16 Transfer MediaStreamTrack

Off-main-thread processing without MSP/MSTG on the main thread requires MediaStreamTrack to be movable to a worker. Also good for other use cases.

Two suggestions, two differences:

- **#21 Make Tracks Transferable, and entangle them with origin context**
 - Default operation is “move” - original track is destroyed; if needed, must use “.clone()”
 - Lifetime strongly tied to original document
- **#24 Make Tracks Serializable, and specify that they end when sources do**
 - Default operation is “copy” - original track survives; if not needed, must use “.stop()”
 - Lifetime tied to source; cameras and microphones are original context only; other sources may have other lifetimes.

For extra credit



Name the lizard!

Thank you

Special thanks to:

WG Participants, Editors & Chairs

The lizard