W3C WebRTC WG Meeting

January 18, 2022
8 AM - 10 AM

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 January 2022 interim meeting of the W3C WebRTC WG, at which we will cover:
  - WebRTC-NV Use Cases
  - Mediacapture-transform
  - Encoded-Transform
  - Capture Handle
- Dates/times for next meetings:
  - February 15, 2022 @ 8 AM Pacific (90 minutes)
  - March 15, 2022 @ 8 AM Pacific (120 minutes)

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/January_18_2022
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - https://w3c.github.io/mediacapture-extensions/
  - 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 - scribing instructions
- The meeting is (still) being recorded. The recording will be public.
- Volunteers for note taking?

W3C Code of Conduct

- This meeting operates under W3C Code of Ethics and Professional Conduct

- We're all passionate about improving WebRTC and the Web, but let's all keep the conversations cordial and professional

Virtual Interim Meeting Tips

This session is being recorded

- **Type +q and -q in the Google Meet chat to get into and out of the speaker queue.**
- **Please use headphones when speaking to avoid echo.**
- **Please wait for microphone access to be granted before speaking.**
- **Please state your full name before speaking.**
- **Poll mechanism may be used to gauge the “sense of the room”.**

Understanding Document Status

- Hosting within the W3C repo does ***not*** imply adoption by the WG.
  - WG adoption requires a Call for Adoption (CfA) on the mailing list.
- Editor's drafts do ***not*** represent WG consensus.
  - WG drafts ***do*** imply consensus, once they're confirmed by a Call for Consensus (CfC) on the mailing list.
  - Possible to merge PRs that may lack consensus, if a note is attached indicating controversy.

Recent CfAs and CfCs

- CfC on WebRTC-NV Use Cases (concluded on December 13):
  - Announcement: Call for Consensus (CfC): WebRTC-NV Use Cases from Bernard Aboba on 2021-11-30 (public-webrtc@w3.org from November 2021)
  - 4 responses with concerns (writeup pending).
  - Discussion at December 21, 2021 Interim.
  - PR by Tim Panton (merged) to address feedback: Tries to reflect Dec-21 interim discussion: by steely-glint · Pull Request #73 · w3c/webrtc-nv-use-cases (github.com)
  - On today's agenda.
- CfA on Region Capture (concluded successfully on December 13):
  - Spec now available at https://w3c.github.io/mediacapture-region
  - Announcement: Re: Follow up on the Call for Adoption of Region Capture from Dominique Hazael-Massieux on 2022-01-17 (public-webrtc@w3.org from January 2022)

Recent CfAs and CfCs (Cont'd)

- CfC on **PR 125** (WebRTC Encoded Transform API): "Add API to request key frames" (Concluded January 17, 2022)
  - REMINDER: Call for Consensus (CfC) on PR 125 (WebRTC Encoded Transform API): "Add API to request key frames" from Bernard Aboba on 2022-01-14 (public-webrtc@w3.org from January 2022)
  - 2 votes for (with a concern relating to sync/races), no objections.
  - Issue 127: manage key frames in case of SFrameTransform key rotation (see slides)
- CfC on Intent to discontinue work on Media Capture Depth Stream Extensions (Concluded on January 26, 2022)
  - Intent to discontinue announcement
  - Call for Consensus (CfC) on Intent to discontinue work on Media Capture Depth Stream Extensions from Bernard Aboba on 2022-01-12 (public-webrtc@w3.org from January 2022)
  - Only two responses so far (from the Chairs).

Issues for Discussion Today

- 08:10 - 08:35 AM WebRTC-NV Use Cases (Tim Panton & Kegan Dougal)
  - 08:10 - 08:25 Slides
  - 08:25 - 08:35 Discussion
- 08:35 - 08:55 AM (mediacapture-transform, Harald)
  - Slides (08:35 - 08:45)
  - Discussion (08:45 - 08:55)
- 08:55 - 09:10 AM (WebRTC-Extensions, Bernard)
  - Slides (09:35 - 09:40)
  - Discussion (09:40 - 09:50)
- 09:10 - 09:50 AM (Capture Handle, Elad)
  - Slides (09:10 - 09:40)
  - Discussion (09:40 - 09:50)
- 09:50 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.

WebRTC-NV Use Cases
(Tim Panton & Kegan Dougal)
End Time: 8:35 AM

WebRTC-NV Use Cases

PR (https://github.com/w3c/webrtc-nv-use-cases/pull/73) merged to reflect Dec-21 interim discussion:

- Tighten up language in use cases
- Remove confusing requirements (deferring solutions)
- Remove N36->N39

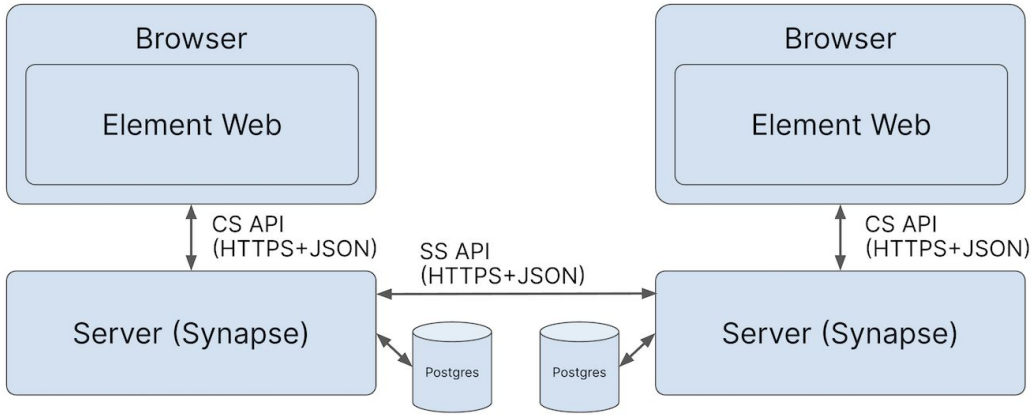
Please review and provide your thoughts. Preview:

https://pr-preview.s3.amazonaws.com/pipe/webrtc-nv-use-cases/pull/73.html

Invited expert input on “3.4 Decentralized messaging”
Thanks to Kegan from matrix for agreeing to help us.

Background: Matrix

- Decentralised communication, mostly instant messaging.
- Core network is an open federation. Very similar to email in network topology design: kegan@matrix.org -> @kegan:matrix.org
- Any client can talk to any server via the Matrix protocol which is an open specification similar to the HTML Living Standard.
- Problem: Everyone ends up registering on the biggest server (matrix.org) -> Not decentralised!
- Solution: make it P2P by default?

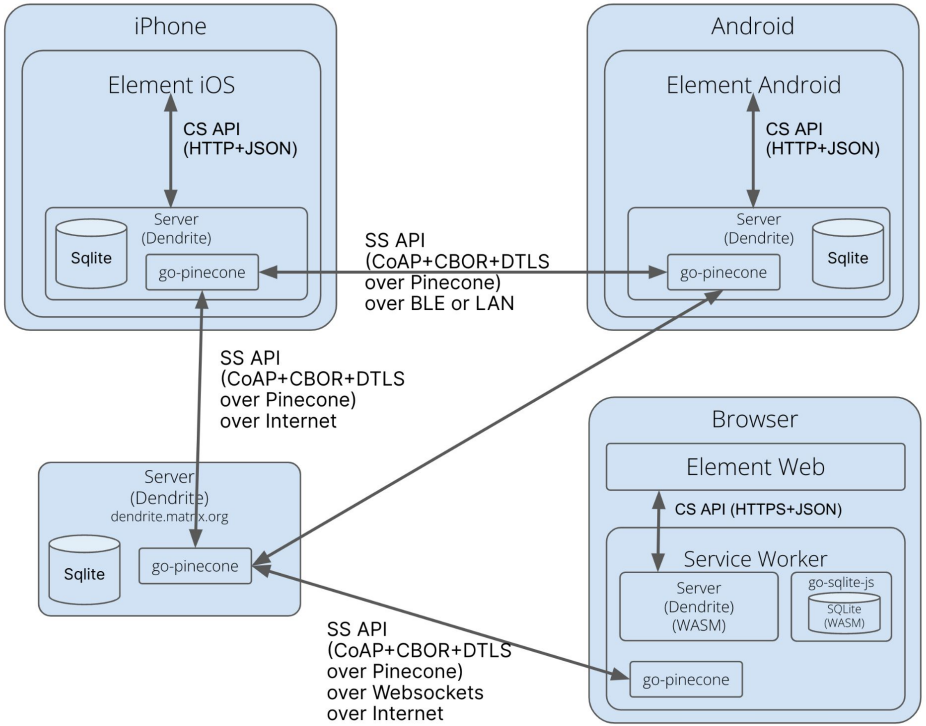


History of P2P Matrix

- We want to reuse as much client code as possible to not fragment the ecosystem, so client protocol shouldn't change.
- So we run a server on the client and change the federation protocol to make it P2P. Clients talk to their local server.
- Compile server down to WASM and run it in service worker (SW) and intercept fetch requests. Send federation requests out over P2P-friendly transports (e.g if I'm on the same LAN as you, I shouldn't need to hit a random server on the internet, just go direct) -> WebRTC connections
- Problem: SWs cannot make WebRTC connections.
- End result: Some early prototypes ended up using websockets to a relay server (for signalling *and data*) -> centralised point, not very P2P! A write-up for these early prototypes (specifically about SWs) can be found here.

P2P Matrix Architecture

Pinecone (our P2P overlay network) works in a range of environments



Design Considerations and Constraints

- Lots of crypto/hashing involved so using a worker of some kind is critical. Sharing work across multiple tabs by using a single service worker rather than N web workers is desirable but not critical.
- It's somewhat desirable to have workers exists for as long as possible as this reduces network churn. Service workers help with that but come with a range of concerns.
- We need to intercept fetch requests from the main window. Shimming is possible (we can monkey patch web SDKs to `postMessage` or something) but then this isn't transparent to client code. Matrix is an open protocol so clients which don't use the SDK won't be able to use P2P.
- We need the ability to establish new P2P connections on-demand as the network topology changes. We could potentially shim data channels from the main window to a worker as the cost of lots of extra boilerplate.

Anti-goals

- SW Lifecycle: We don't need to receive messages in the background when the user doesn't have a tab open. It would be nice as it would reduce network churn and improve stability of the P2P network but that comes at a (frankly) unacceptable cost to individuals due to non consensual usage of network bandwidth, storage, CPU and memory. This could be made consensual by browsers by exposing the existence of service workers.
- Fetch: We can't use fetch as an alternative to true P2P as not all nodes will be directly routable to each other so we won't know what IP address to use. As a rough vision for what we'd like as an outcome: it should be possible to take a few friends out camping in the woods and communicate in-browser without any internet access. Our more advanced P2P overlay networks work transparently over a range of transports including Bluetooth, WebSockets, TCP, etc.

Proposals (one of, preferred first)

- **A: Allow service workers to make WebRTC connections.**
- **B: Allow web workers to intercept fetch requests and make WebRTC connections.**
  - can use web worker instead of service worker.
  - resolves issues around SW lifecycle concerns.
- **C: Allow web workers to make WebRTC connections.**
  - can probably shim fetch traffic from service worker -> window -> web worker. A lot of extra work.
- **D: Allow web workers to control WebRTC data channels.**
  - without this we cannot do true P2P

For prototyping with WebRTC in-browser, we'll probably use libp2p initially.

Discussion (End Time: 8:35 AM)

-

mediacapture-transform (Harald)

End Time: 08:55 AM

Mediacapture-Transform: FPWD

This specification has been adopted by the WG.

The next step is First Public Working Draft publication.

FPWD publication triggers licensing requirements on WG members.

WD publication requirements

- Does NOT require consensus
- Does NOT imply W3C endorsement
- SHOULD document outstanding issues

https://www.w3.org/2021/Process-20211102/#RecsWD

Open issues on this spec

- #4 Right approach?
- #20 “Real-time” warning/note
- #23 Out-of-main-thread processing
- #26 API for tuning MST internal state
- #29 Audio
- #30 Memory locality
- #34 Relationship to WebGPU
- #65 Video rotation

Suggested dispositions (1)

- #4 Right approach?
  - Leave issue open as placeholder. Not blocking.
- #20 “Real-time” warning/note
  - Adopt suggested warning text
- #23 Out-of-main-thread processing
  - Documented in spec as open issue
- #26 API for tuning MST internal state
  - “Muted” attribute is already in spec. Closed issue.

Suggested disposition (2)

- #29 Audio
  - Documented in spec as open issue
- #30 Memory locality
  - Exploratory in nature, leave open
- #34 Relationship to WebGPU
  - Exploratory in nature, leave open
  - Solution proposed in WebCodecs PR 412
- #65 Video rotation
  - Seems to be a VideoFrame metadata issue - move

Call For Consensus for FPWD

In my opinion, none of the issues are blocking for issuing a FPWD CfC.

Obvious WG disagreements are documented; other issues are not important enough to block FPWD publication.

We should issue the CfC right after this meeting.

CfC - possible responses

We will ask for members to respond in one of two ways:

- I support publishing this document as FPWD
- I oppose publishing this document as FPWD at this time, because of issue (#issue number)

If there are objections, we will discuss and attempt to resolve the issues, and evaluate afterwards whether we need a new CfC or not.

If there are none, we will ask W3C to publish this document as a FPWD.

Raising and resolving editorial issues can be done at any time.

Discussion (End Time: 08:55 AM)

-

WebRTC-Extensions (Bernard)

End Time: 9:10 AM

PR 125: Add API to Request Key Frames

- PR 125 (now on WebRTC-Encoded-Transform) Includes three APIs:
  - An API to generate a key frame from a RTCRtpScriptTransformer on sender side.
  - A corresponding API in RTCRtpSender. This API shares the same algorithm as the first one.
  - An API to request a key frame (through FIR) from a RTCRtpScriptTransformer on receiver side

PR 125: Add API to Request Key Frames (cont'd)

- Call for Consensus (ended on January 17, 2022): https://lists.w3.org/Archives/Public/public-webrtc/2022Jan/0002.html
- Concern(s) raised relating to synchronization:
  - PR 125 differs from PR 37 key frame generation in that a timestamp is not returned. Does it matter?
    - Re: Call for Consensus (CfC) on PR 125 (WebRTC Encoded Transform API): "Add API to request key frames" from Bernard Aboba on 2022-01-03 (public-webrtc@w3.org from January 2022)
    - Does the application need to know which keyframe is the one it asked to be generated?
  - Issue 127: How to synchronize key frame generation with key rotation?

Issue 127: How to manage key frames in case of SFrameTransform key rotation

- Youenn: By the time generateKeyframe() promise returns, the corresponding frame might already have been encrypted (in case pipeTo is used with a SFrameTransform in particular). So this won't work:

```
await Promise.all([
  sender.generateKeyFrame()/transformer.generateKeyFrame(),
  sender.transform.setEncryptionKey(newKey, newKeyId)
])
```

- Do we need a single (atomic) method that generates a key frame and sets the Encryption key/keyId?

Discussion (End Time: 9:10 AM)

-

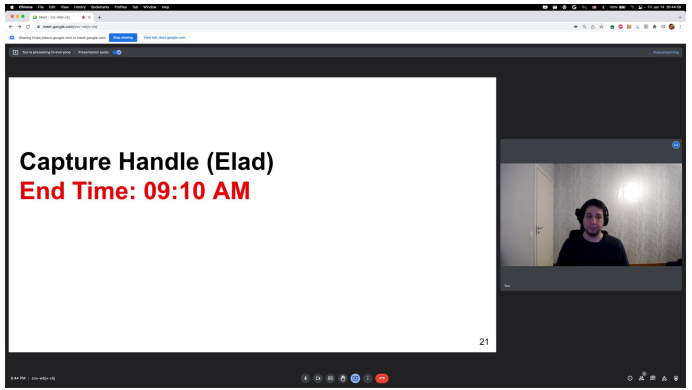
Capture Handle (Elad)

End Time: 09:50 AM

Capture Handle - Reminder 1/2

Assume a video-conferencing application (VC-App) is display-capturing another tab, where a slides-presenting application lives (Slides-App).

Presently, the user knows what they chose to capture, but VC-App does not. The VC-App cannot do much more than pipe the pixels to remote users.



What if the user, while engaged with VC-App, wishes to navigate Slides-App to the next slide? The user would have to switch tabs.
Repeatedly.
Quite distracting.

Capture Handle - Reminder 2/2

I have previously proposed a mechanism that allows applications to declare their identity to capturing applications. This allows cooperating applications to bootstrap communication.

For example, given shared cloud infrastructure, the capturee can advertise its ID. When the capturer gets that ID, it can send the captured messages via that infrastructure. These messages can be simple (prev/next-slide) or arbitrarily complex.

Example follows.

Capture Handle Identity - Example 1/2 (Capturee)

```
function onPageLoaded() {
  ...
  setCaptureHandleConfig({
    // Expose the origin of the THIS document.
    exposeOrigin: true,

    // Expose some ID meaningful in a shared cloud infrastructure.
    handle: getSessionId(),

    // Allow anyone to read the above info (origin and handle).
    permittedOrigins: ['*']
  });
  ...
}
```

Capture Handle Identity - Example 2/2 (Capturer)

```
function startCapture() {
  const stream = await navigator.mediaDevices.getDisplayMedia();
  const [track] = stream.getVideoTracks();
  if (track.getCaptureHandle) {  // Feature detection.
    // Subscribe to notifications of the capture-handle changing.
    track.oncapturehandlechange = (event) => {
      OnNewCaptureHandle(event.captureHandle());
    };
    // Read the current capture-handle.
    OnNewCaptureHandle(track.getCaptureHandle());
  }
}

function OnNewCaptureHandle(captureHandle) {
  if (captureHandle.origin != 'slides-3000.com')
    return;
  // Exposes prev/next buttons to the user. When clicked, these send
  // a message to some REST API, where |sessionId| indicates that the
  // message has to be relayed to the Slides 3000 session in question.
  ExposeSlides300Controls(captureHandle.handle);
}
```

Capture Handle Actions

The aforementioned proposal works for closely cooperating applications. What about less tightly bound applications?

We can do something there, too, using another mechanism. Namely, the capturee could advertise what actions it supports (e.g. prev/next), and the capturer could expose user-facing controls based on that.

It's an open question whether these the two mechanisms belong together. Let's name them for the time being:

- Capture Handle **Identity**: The mechanism presented in previous meetings and slides.
- Capture Handle **Actions**: An additional mechanism for sending simple messages from capturer to capturee, like prev-slide, next-slide.

Capture Handle Actions API (Capturee)

We define a set of supported actions:
`CaptureActions := [“first”, “prev”, “next”, “last”, ...]`

We expose a control for top-level documents to declare the actions they support. (Throws if called from non-top-level document.)
`MediaDevices.setSupportedCaptureActions([“prev”, “next”]);`

We expose a control for top-level documents to set a handler.
`MediaDevices.setCaptureActionsHandler(handler);`

Open question - besides the action, what else should be contained in the events which are passed to the handler. Origin of sender? Opaque source identifier?

Capture Handle Actions API (Capturer)

On the video MediaStreamTrack returned by getDisplayMedia(), we expose:

```
// Returns set of actions supported by the application
// associated with the video track.
// (If not a video track associated with tab-capture - empty.)
MediaStreamTrack.getSupportedCaptureActions();

// If this MediaStreamTrack is a video track associated with
// tab-capture, and if the top-level document in that tab
// registered a CaptureActionsHandler, fire an event for that
// handler with this |capture_action|.
MediaStreamTrack.sendCaptureAction(capture_action);
```

Capture Handle Actions API - Known “Issue”

Note that the captured tab’s top-level frame could be navigated at any time. This means that the message sent from the capturer to the capturee could be asynchronously delivered to an “unintended” recipient. However, since this mechanism (Capture Handle Actions) is intended for use when the capturer/capturee do not know each other anyway, and have no other channel for communication, this is **not an issue**; the process is user-controlled, and this edge case is equivalent to the user pressing “next slide” on their keyboard at an inopportune time in a single-tab experience. Put another way, it’s orthogonal to the capture.

The mechanism of Capture Handle Identity **is** robust to captured-tab navigation.

- Capturer/capturee establish their own communication channel. It is out of scope which, but it would most likely not misfire if the captured tab is navigated.
- Recall the CaptureHandleChangeEvent, which allows establishing a new communications channel with the new application loaded in the captured tab.

Capture Handle - Need for Both APIs

We have previously discussed why the Identification mechanism would still be needed even if the Actions mechanism is introduced. To recap some arguments briefly:

- Allows (voluntarily) sending some information from the capturee to the capturer without alerting the capturee to the presence of a capture.
- Two-way communication if necessary (but not required).
- Greater flexibility in messaging, both in terms of content and implementation.
- Allows verification beyond origin (capturee can require arbitrary type of credentials before acting on messages, building on whichever technology these cooperating applications have previously built).
- Applications that are very tightly cooperating can essentially fuse into one combo-application, cooperating as closely as when embedded in an iframe.
- Share-this-tab-instead (elaborated verbally)

Discussion (End Time: 09:50 AM)

-

Thank you

Special thanks to:

WG Participants, Editors & Chairs