

W3C WebRTC WG Meeting

March 26, 2024
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 March 2024 interim meeting of the W3C WebRTC WG, at which we will cover:
 - WebRTC-PC, Reactions, Mediacapture Specifications, RTCRtpSenderEncodedSource, WebRTC Extended Use Cases
- [Future meetings:](#)
 - [April 23](#)
 - [May 21](#)
 - [June 18](#)
 - [July 16](#)

About this Virtual Meeting



- Meeting info:
 - https://www.w3.org/2011/04/webrtc/wiki/March_26_2024
- 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](#)
- 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 Conduct](#) (updated!)
- 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 (still) being recorded

- Click  Raise hand **to get into the speaker queue.**
- Click  Lower hand **to get out of the speaker queue.**
- **Please wait for microphone access to be granted before speaking.**
- **If you jump the speaker queue, you will be muted.**
- **Please use headphones when speaking to avoid echo.**
- **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.

Issues for Discussion Today

- 08:10 - 08:20 AM [WebRTC-PC](#) (Tim Panton)
- 08:20 - 08:35 AM [Reactions](#) (Youenn)
- 08:35 - 09:15 [Mediacapture Specifications](#) (Jan-Ivar)
- 09:15 - 09:35 AM [RTCRtpSenderEncodedSource](#) (Guido)
- 09:35 - 09:50 AM [WebRTC Extended Use Cases](#) (Sun, Shridhar)
- 09:50 - 10:00 AM [Wrapup and Next Steps](#) (Chairs)

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-PC

Start Time: 08:10 AM

End Time: 08:20 AM

For Discussion Today

- [Issue 2944](#): Alternative storage for RTCCertificates needed (Tim Panton)

RTCCertificates in the Wild

- Medium term identity persistence
 - Between peers
 - No need for centralized identity service
 - Useful in outages
 - Or IoT reconnections
-
- Facilitates ToFU.

RTCCertificate Problems

- Valid for up to 1 year
- But Expiry unchecked!
- Can be stored in indexedDB
- Irretrievably lost when indexedDB wiped
- Safari wipes indexedDB after 1 week
- Users regularly wipe storage to get better prices
- No export or backup mechanism provided

(IDP use-case never took off...)

Proposal 1

Save to indexedDB store (borrowed from MediaKeySession) :

RTCCertificate.load(fingerprint)

RTCCertificate.store(certificate)

This assumes the app knows the fingerprint - but that can be stored in a URL param.

Proposal 2

Export to blob allowing the user/app to save it:

```
RTCCertificate.exportPEM(certificate)
```

```
RTCCertificate.importPEM(blob)
```

This goes against the original security design where keys are never exposed in javascript.

However all the E2E implementations have moved beyond that design
Time to revisit?

Other proposals ?

Other ideas welcome.

(e.g. leverage client certificates somehow :

<https://developer.chrome.com/docs/extensions/reference/api/platformKeys>

Perhaps ?)

Discussion (**End Time: 08:20**)

-

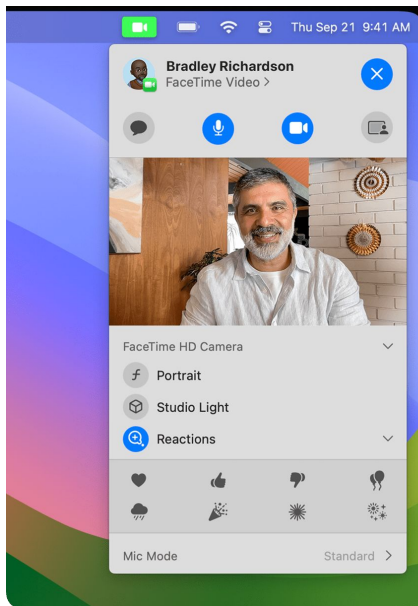
Reactions (Youenn)

Start Time: 08:20 AM

End Time: 08:35 AM

For discussion today

- [Mediacapture-extensions #118](#): Should web applications be aware of reaction effects added by OS to camera feeds?



Not in scope



In scope

[Mediacapture-extensions#118](#)

- Reactions are fun and useful in many contexts
 - Reactions may not be useful in specific contexts, like telehealth
- Web applications want to
 - Provide hints to the UA that gesture reactions are useful or not
 - In scope
 - Be aware that reactions may be added upon user gestures
 - In scope
 - Be aware that reactions are added to specific video frames
 - Not in scope today (could be a future metadata)

[Mediacapture-extensions#118](#)

- Proposed solution
 - Use a boolean constraint/capability/setting for gesture reactions
 - Similar to background blur solution
 - The constraint mechanism is flexible, as it allows:
 - UA to expose whether reactions are on or off
 - UA to expose whether reactions can be turned on/off by web application
 - Web application to provide to UA whether reactions are desired
 - <https://github.com/w3c/mediacapture-extensions/pull/141>
 - Constraint name under discussion
- Is the WG ok with providing a solution to this problem
 - If so, is the WG OK with this solution?

Discussion (**End Time: 08:35**)

-

Mediacapture Specifications (Jan-Ivar)

Start Time: 08:35 AM

End Time: 09:15 AM

For discussion today

- [Mediacapture-main #984](#): Clarify each source is responsible for specifying mute/unmute/ended and constraints behavior
- [Mediacapture-record #194](#): mimeType ambiguity: "video/webm;codecs=vp8" means?

~~Issue 984~~: Clarify each source is responsible for specifying mute/unmute/ended and constraints behavior

This is a spec cleanup to remove any implicitly inherited behaviors, requiring each spec that defines a source of new `MediaStreamTracks` to follow [§ 17.4 Defining a new source of `MediaStreamTrack`](#), specifically:

- declare which constrainable properties (see [4.3.8 Constrainable Properties](#)), if any, are applicable to each [kind](#) of media this new [source](#) produces, and how they work with this source,
- describe how and when to [set a track's muted state](#) for this [source](#),
- describe how and when to [end](#) tracks from this [source](#),

~~Issue 984~~: Clarify each source is responsible for specifying mute/unmute/ended and constraints behavior



This was referenced last month

Review mute/unmute/ended and constraints on tracks from `getDisplayMedia()` [w3c/mediacapture-screen-share#298](#)

Open

Review mute/unmute/ended and constraints on tracks from `element.captureStream()` [w3c/mediacapture-fromelement#98](#)

Open

Review mute/unmute/ended and constraints on tracks from `canvas.captureStream()` [w3c/mediacapture-fromelement#99](#)

Open

Review mute/unmute/ended and constraints on new `VideoTrackGenerator().track` [w3c/mediacapture-transform#109](#)

Open

Review mute/unmute/ended and constraints on track in `audioContext.createMediaStreamDestination().stream` [WebAudio/web-audio-api#2571](#)

Open

Review mute/unmute/ended and constraints on `RTCRtpReceiver`'s track. [w3c/webrtc-pc#2942](#)

Open



alvestrand closed this as completed in [#988](#) last month

Issue 298: Review mute/unmute/ended and constraints on tracks from `getDisplayMedia()`

✓ [§ 5.2 Closed and Minimized Display Surfaces](#) specifies mute/unmute/ended.

A [display surface](#) that is being shared may temporarily or permanently become inaccessible to the application because of actions taken by the operating system or user agent. What makes a [display surface](#) considered inaccessible is outside the scope of this specification, but examples *MAY* include a [monitor](#) disconnecting, [window](#) or [browser](#) closing or becoming minimized, or due to an incoming call on a phone.

Above definition seems to support the "togglecreenshare" UA toggle in [mediasession #306](#)

When [display surface](#) enters an inaccessible state that is not necessarily permanent, the user agent *MUST* queue a task that [sets the muted state](#) of the corresponding media track to `true`.

When [display surface](#) exits an inaccessible state and becomes accessible, the user agent *MUST* queue a task that [sets the muted state](#) of the corresponding media track to `false`.

When a [display surface](#) enters an inaccessible state that is permanent (such as the source [window](#) closing), the user agent *MUST* queue a task that [ends](#) the corresponding media track.

A stream that was just returned by `getDisplayMedia` *MAY* contain tracks that are muted by default. Audio and video tracks belonging to the same stream *MAY* be muted/unmuted independently of one another.

✓ [§ 5.4 Constraining Properties for Captured Display Surfaces](#) covers constraints.

Proposal: close as reviewed, optionally with a PR to include UA privacy toggle among examples.

Issue 2942: Review mute/unmute/ended and constraints on RTCRtpReceiver's track.

✓ [§ 9.3 MediaStreamTrack](#) specifies **mute** from BYE and sRD(inactive|sendonly), **unmute** from incoming RTP, and **ended** from transceiver [stop sending and receiving](#),

✓ [§ 9.3.1 MediaTrackSupportedConstraints, MediaTrackCapabilities, MediaTrackConstraints and MediaTrackSettings](#) covers constraints.

State of implementations (from dup [#2915](#)): ✓ Bugs filed on all UA-specific behavior

- Firefox appears to follow [§ 9.3 MediaStreamTrack](#)
- Safari deviates by
 - unmuting after 1 ms, likely ahead of RTP reception, compared to ~32 ms in Firefox ([webkit 209242](#))
 - muting tracks on stopped transceiver (webkit bug?)
- Chrome deviates by
 - unmuting ahead of RTP / sRD ([crbug 1295295](#))
 - muting video (only!) track ~1 second after sender.track.stop() ([crbug 941740](#))

Proposal: close as reviewed and close [#2915](#). No implementation-defined behavior allowed.

Issue 109: Review mute/unmute/ended and constraints on new VideoTrackGenerator().track

✓ Mute/unmute is controlled by VideoTrackGenerator's [muted](#) attribute:

```
// worker.js
const source = new VideoTrackGenerator();
source.muted = true;
```

The source.track is ended when the source.writable is [closed](#).

✓ [§ 2.2.5.2. Constrainable properties](#) covers constraints.

Proposal: close as reviewed.

Issue 98: Review mute/unmute/ended and constraints on tracks from `element.captureStream()`

✓ [element.captureStream](#) specifies when to **end** a track:

A captured `MediaStreamTrack` **ends** when `playback ends` (and the `ended` event fires) or when the track that it captures is no longer selected or enabled for playback. A track is no longer selected or enabled if the source is changed by setting the `src` or `srcObject` attributes of the media element.

✗ For **mute/unmute** terms like "available" (reactive) and "accessible" (same-origin) content.

Should be clarified. Intent seems to be to carry forward state of MSTs in `video.srcObject`.

✗ No mention of constraints. See [define behaviors of ConstrainablePattern Interfaces #48](#)

Related issues:

- [Inconsistency: Taint, not mute cross-origin element tracks #83](#)

Proposal: PR to clarify “available content” in element playback terms, for `srcObject = MS` and otherwise. No implementation-defined behavior allowed.

Issue 99: Review mute/unmute/ended and constraints on tracks from `canvas.captureStream()`

✗ [`canvas.captureStream`](#) only mentions mute once related to no longer being origin-clean (though would `ended` be more appropriate? Or can dirty canvases be cleaned?)

A captured stream *MUST* immediately cease to capture content if the [origin-clean](#) flag of the source canvas becomes false after the stream is created by `captureStream()`. The captured `MediaStreamTrack` *MUST* become [muted](#), producing no new content while the canvas remains in this state.

✗ It doesn't say anything about `ended` or `unmute`. It should clarify if events are fired. No mention of constraints.

Relevant open issues found:

- [Clarify if CanvasCaptureMediaStreamTrack mute, unmute, and ended events are expected to be fired #82](#)

Proposal: Close as dup of #82. No implementation-defined behavior allowed.

Issue 2571: Review mute/unmute/ended and constraints on track in `audioContext.createMediaStreamDestination().stream`

- ✓ No mention of muted, ended or constraints on `MediaStreamAudioDestinationNode`'s track.
- ✓ If a `MediaStreamAudioDestinationNode`'s track is never muted or ended, and cannot be constrained (with `applyConstraints`), this is now the default behavior, but it might be good to call this out if intentional. E.g. something like *"The `MediaStreamAudioDestinationNode`'s stream's track is never muted or ended, and has no constraints."*

No action: Leave open. Different Working Group.

Issue 194: mimeType ambiguity: "video/webm;codecs=vp8" means?

Works in Chrome but throws `NotSupportedError` in Firefox because *audio codec not mentioned*:

```
const stream = await navigator.mediaDevices.getUserMedia({video: true, audio: true});
const rec = new MediaRecorder(stream, {mimeType: "video/webm;codecs=vp8"}); // no + ",opus"
rec.start();
```

`mimeType` overriding browsers' default codecs seems intuitive. But spec and Firefox also treat it as an input selector, which seems redundant, when better ways to exclude audio exist:

```
new MediaRecorder(new MediaStream(...stream.getVideoTracks()), options); // no audio
```

The latter works the same for default and modified codecs, IOW input filtering and codec overriding are orthogonal, which seems better.

Proposal: Align spec with Chrome

Discussion (**End Time: 09:15**)

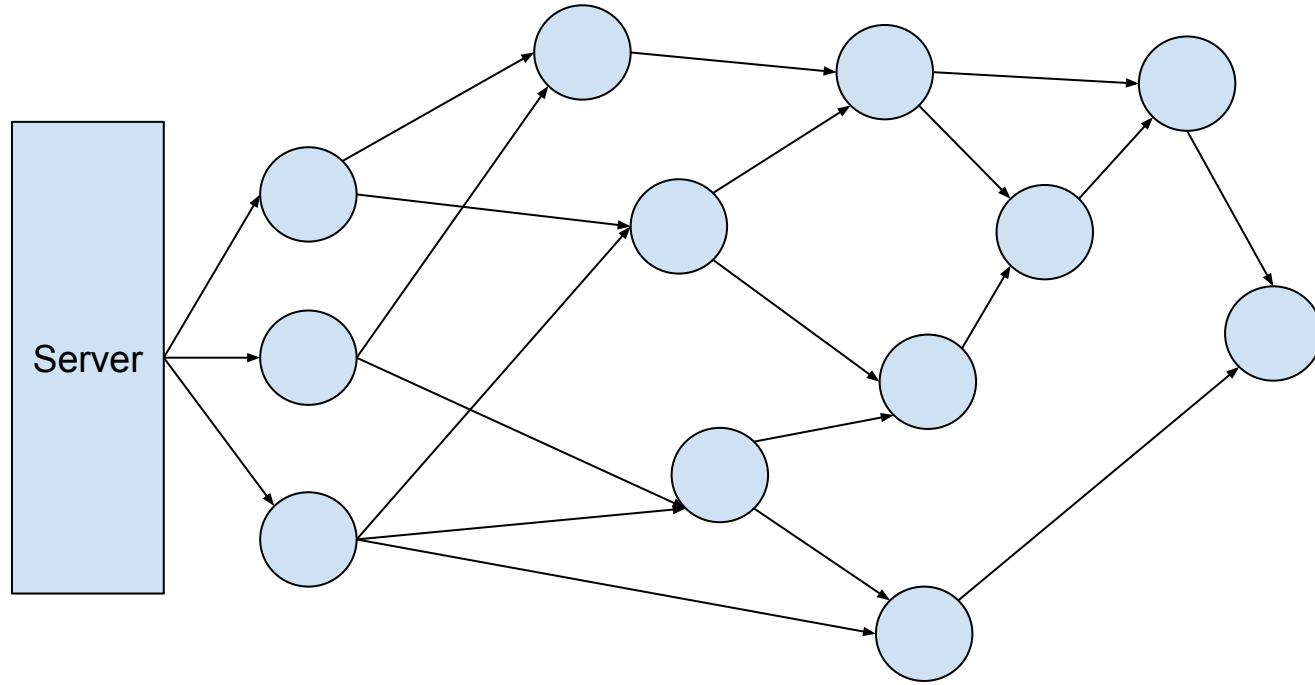
-

RTCRtpSenderEncodedSource (Guido)

Start Time: 09:15 AM

End Time: 09:35 AM

Scenario



- A few nodes read from a server, thousands of nodes in a P2P system
- P2P network topology with redundant paths for reliability
- Nodes are generally unreliable (can join/leave at any time)
- **Use case:** Glitch-free forwarding from redundant paths (Fan-in and Fan-out)

Proposal - RTCRtpSenderEncodedSource

```
// main.js
const worker = new Worker('worker.js');
const encodedSource = new RTCRtpSenderEncodedSource(worker, {name: "encodedSource"})

// Let relayPc be the PC used to relay frames to the next peer.
const [sender] = relayPc.getSenders();
await sender.replaceTrack(encodedSource);

// Let recvPc1, recvPc2 be the receiving PCs.
recvPc1.ontrack = ({receiver}) =>
  receiver.transform = new RTCRtpScriptTransform(worker, {name: "receiverTransform1"});
recvPc2.ontrack = ({receiver}) =>
  receiver.transform = new RTCRtpScriptTransform(worker, {name: "receiverTransform2"});
```

Proposal - RTCRtpSenderEncodedSource

```
// worker.js
let sourceWriter;
onrtcsenderencodedsources = ({controller: {writable}}) => {
  sourceWriter = writable.getWriter();
}
onrtctransform = async ({transformer: {readable, writable, options}}) => {
  await readable.pipeThrough(new TransformStream({transform})).pipeTo(writable);
  function transform(frame, controller) {
    if (shouldForward(frame)) { // application-defined (e.g., drop duplicates)
      const newFrame = new RTCRtpEncodedVideoFrame(frame, getUnifiedMetadata(frame));
      sourceWriter.write(newFrame);
    }
    controller.enqueue(frame);
  }
}
```

API Shape (see [webrtc-extensions #198](#))

```
[Exposed=Window] interface RTCRtpSenderEncodedSource {
    constructor(Worker worker, optional any options, optional sequence<object> transfer)
};

[Exposed=DedicatedWorker] interface RTCRtpSenderEncodedSourceController {
    WritableStream writable;    // Need congestion/error API
};

partial interface DedicatedWorkerGlobalScope {
    attribute EventHandler onrtcsenderencodedsource;
}

partial interface RTCRtpSender {
    undefined replaceTrack(RTCRtpSenderEncodedSource source);
    readonly attribute RTCRtpSenderEncodedSource encodedSource;
}
```

Pros and cons

- Same pattern as encoded transform.
 - Makes it easy to evolve both in parallel
 - Can use same API for bandwidth congestion (see encoded-transform PR #224)
- Builds on existing APIs proven in production
- Good match for SFU-like operations that are frame centric:
 - Zero-timeout, glitch-free forwarding of frames from redundant paths
 - Drop frames from certain layers in response to bandwidth issues
- Requires waiting for a full frame, which introduces extra latency compared with a packet-based API

Discussion (End Time: 09:35)

- Do we have rough consensus on:
 - Introducing RTCRtpSenderEncodedSource
 - Constructors for RTCEncodedVideoFrame and RTCEncodedVideoFrame to create new frames with custom metadata ([encoded-transform #223](#))

WebRTC-Extended Use Cases (Sun, Shridhar)

Start Time: 09:35 AM

End Time: 09:50 AM

PR#118: Bandwidth feedback Speed(Configurable RTCP transmission interval)

- Proposal:
 - To address the need for precise bandwidth management, we propose the implementation of a configurable control knob within the application. This control will enable the application to manage the timing of notifications related to bandwidth changes, specifically those concerning packet loss and packet reception timing.

PR#118: Bandwidth feedback Speed(Configurable RTCP transmission interval)

- Implementation details - not to add this to the PR:
 - **Directional Control**: The application will have the ability to adjust the notification delay in both directions, either to expedite or defer the notification of bandwidth changes.
 - **Degree of Control**: The application will be able to specify the exact delay amount, allowing for granular control over the timing of the notifications. This could be defined in terms of time (e.g., milliseconds) or synchronized with RTP timestamp changes (Detailed syntax has been proposed to [IETF](#)).

PR#118: Bandwidth feedback Speed(Configurable RTCP transmission interval)

- Benefits - not to add this to the PR:
 - **Enhanced Responsiveness:** By controlling the notification delay, the application can better manage its response to bandwidth fluctuations, leading to improved performance and user experience.
 - **Customizable feedback interval:** Different applications may require different levels of sensitivity to bandwidth changes. This control knob will allow for tailored configurations that best suit each application's needs.
 - Ex: L4S(RFC 9330) needs interval to be configured to 25ms or lower.

PR#118: Bandwidth feedback Speed(Configurable RTCP transmission interval)

- Discussion point:
 - The addition of this control knob will significantly improve the application's ability to respond to network conditions in a timely and efficient manner, ultimately enhancing overall service quality.
 - Actual implementation details may not be the scope of this requirements but we are open to continue discussion on it through W3C working group discussions. Could we submit this change?

PR#129: video decoding recovery after packet loss

- Proposal:
 - The application must be able to control video decoding to continue even after a frame-loss without waiting for a key frame.
- Open Question - not to add this to the PR:
 - For which codecs and which platforms do we have experience with video loss recovery without a keyframe that has proved to be helpful? (Not as a suggestion to add this to the PR, but because I wonder how well this works in practice)

PR#129: video decoding recovery after packet loss

- NVIDIA GeForce NOW evaluation - not to add this to the PR
 - Platforms: Native implementation on Windows and Mac.
 - Codecs: H.264, HEVC and AV1
 - Utilizes a custom protocols for client-server communication
 - Benefits: Lower frame sizes compared to IDR(Instantaneous Decoder Refresh), lower bits on network, better quality.
 - Ref> Meta shared that they applied the LTR(Long Term Reference) over H.264 on Messenger and showed 37% reduction of the usage of the key frames: [link](#)

PR#129: video decoding recovery after packet loss

- Discussion point:
 - In cloud gaming scenarios, particularly one-to-one communication, we have observed feasible improvements in video streaming by native implementation and are now seeking industry consensus to extend these improvements to browser-based implementations.
 - Actual implementation details may not be the scope of this requirements but we are open to continue discussion on it through W3C working group discussions. Could we submit this change?

Discussion (**End Time: 09:50**)

-

Wrapup and Next Steps

Start Time: 09:50 AM

End Time: 10:00 AM

Next Steps

- Content goes here

Thank you

Special thanks to:

WG Participants, Editors & Chairs