W3C WebRTC WG Meeting

March 30, 2020
8 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 interim meeting of the W3C WebRTC WG!
  - During this meeting, we hope to make progress on features at risk, privacy concerns and potential new work relating to audio acquisition.

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/March_30_2020
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - 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-stats/
  - https://www.w3.org/TR/mst-content-hint/
  - https://w3c.github.io/webrtc-nv-use-cases/
  - https://w3c.github.io/webrtc-dscp-exp/
  - 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.

Issues for Discussion Today

- WebRTC-PC
  - Features at risk
  - Issue 2495: When is negotiation complete? (Jan-Ivar)
  - Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)
- Media Capture and Streams
  - Issue 671: New audio acquisition constraints (Sam Dallstream)
  - Issue 639: Enforcing User Gesture for getUserMedia (Youennf)
  - Issue 640: Only reveal labels of device user has given permission to (Youennf)
  - Issue 669: "user-chooses": Does required constraints make any sense now? (Henrik)
  - Issue 672: Deprecate inputDeviceInfo.getCapabilities() for privacy (Jan-Ivar)

Issues for Discussion Today

- WebRTC-PC
  - Features at risk
  - Issue 2495: When is negotiation complete? (Jan-Ivar)
  - Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)

WebRTC-PC Features at Risk (Bernard)

- Should we mark these unimplemented features as “Features at Risk”?
  - Issue 2496: voiceActivityFlag
    - Exposed in RTCRtpSynchronizationSource
  - Issue 2497: Unimplemented MTI Stat
    - Also remove MTI designation:
    - RTCInboundRtpStreamStats - partialFramesLost
  - Issue 2498: Multiple DTLS Certificates
    - WPT test fails
    - Not a high priority for any browser

Issue 2495: When is negotiation complete? (Jan-Ivar)

This problem arose while writing WPT tests for perfect negotiation. Need something like

```
const transceiver = pc.addTransceiver("video"); // for example
await /* some event or promise */
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");
```

Issue 2495: When is negotiation complete? (Jan-Ivar)

The obvious approach is racy, outside highly controlled cases:

```
const transceiver = pc.addTransceiver("video");
await state(pc, "stable");
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");

function state(pc, s) {
  return new Promise(r => pc.onsignalingstatechange = () => pc.signalingState == s && r());
}
```

We might reach `"stable"` from rollback, or answering a remote offer.

Or from a previous negotiation we just missed locally.

Issue 2495: When is negotiation complete? (Jan-Ivar)

We could write terrible action-specific spin-tests:

```
const transceiver = pc.addTransceiver("video");
while (!transceiver.currentDirection) { // spin-test specific to addTransceiver
  await state(pc, "stable");
}
assert_true(true, "we didn't time out!");
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");
```

But in general, APIs that time out on failure stink.

Issue 2495: When is negotiation complete? (Jan-Ivar)

As a workaround, JS can dispatch its own “negotiated” event from SRD(answer)

```
// - The perfect negotiation logic, separated from the rest of the application ---

signaling.onmessage = async ({data: {description, candidate}}) => {
  try {
    if (description) {
      const offerCollision = description.type == "offer" &&
                             (makingOffer || pc.signalingState != "stable");

      ignoreOffer = !polite && offerCollision;
      if (ignoreOffer) {
        return;
      }
      await pc.setRemoteDescription(description); // SRD rolls back as needed
      if (description.type == "offer") {
        await pc.setLocalDescription();
        signaling.send({description: pc.localDescription});
      } else {
        pc.dispatchEvent(new Event("negotiated")); // <--- here!
      }
    } else if (candidate) {
```

Issue 2495: When is negotiation complete? (Jan-Ivar)

This avoids rollbacks & remote offers, but we must still account for missing a local train:

```
const transceiver = pc.addTransceiver("video");
await negotiated(pc);
if (!transceiver.currentDirection) { // non-spin test specific to addTransceiver
  await negotiated(pc); // catch the next train
}
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");

function negotiated(pc) {
  return new Promise(r => pc.addEventListener("negotiated", r, {once: true}));
}
```

We get rid of the while-loop, but not the action-specific completion-test.

Issue 2495: When is negotiation complete? (Jan-Ivar)

Proposal A: Fire negotiationcomplete from SRD(answer) if renegotiation isn't needed

```
const transceiver = pc.addTransceiver("video");
await new Promise(r => pc.onnegotiationcomplete = r);
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");
```

Simple. One downside is subsequent actions may delay this event.

Issue 2495: When is negotiation complete? (Jan-Ivar)

Proposal B: Expose a `negotiationneeded` boolean attribute.

```
const transceiver = pc.addTransceiver("video");
while (pc.negotiationneeded) await state(pc, "stable");
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");
```

Complication: once set, browser MUST fire negotiationneeded, to meet JS expectation.

Has the same downside that subsequent actions may delay settling.

Issue 2495: When is negotiation complete? (Jan-Ivar)

Proposal C: Expose a `negotiationcomplete` Promise<void> attribute.

```
const transceiver = pc.addTransceiver("video");
await pc.negotiationcomplete;
assert_equals(transceiver.currentDirection, "sendonly", "negotiates to sendonly");
```

Fulfillment of this new promise would ***not*** be delayed by subsequent actions (accomplished by the browser replacing the attribute with a new promise each time a negotiation train leaves the station)

Collectively, *"it's the promise `addTransceiver` et al. should have returned!"*

Bonus:

```
const lastPromise = pc.negotiationcomplete;
await foo();
if (lastPromise == pc.negotiationcomplete) { /* still on the same negotiation train */
```

Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)

Silly question? Spec: In-parallel, do “foo”. When “foo” is done, queue a task.

- Example: In parallel, apply SDP according to JSEP. If successful, queue a task that updates the *set of transceivers* and internal slots accordingly.

It would seem clear that effects are surfaced in a queued task.

… but what about addTrack()? JSEP:

> *If the PeerConnection is in the "have-remote-offer" state, the track will be attached to the first compatible transceiver that was created by the most recent call to setRemoteDescription() and does not have a local track.*

Note:

- JavaScript is supposed to act “single threaded”.
- Most PC states are represented by internal slots, updated in tasks. The *set of transceiver* and *signaling state* are not defined as “internal slots”, but they are defined in webrc-pc and get and set inside tasks (JS thread).
- The same concepts exist in JSEP, which executes in-parallel. Same or different?

Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)

Does this addTrack() create a new transceiver or use the one created by SRD?

```
pc.setRemoteDescription(offerSdpThatCreatesACompatibleTransceiver);
// Don't await in-between.
pc.addTrack(track);
```

Has a compatible transceiver been created yet by "*the most recent call to setRemoteDescription()*" when addTrack() is executed?

If addTrack() operates on webrtc-pc's *signaling state* and *set of transceivers* then:

NO

If addTrack() operates on JSEP's *signaling state* and *set of transceivers* then:

Maybe?

Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)

But aren't webrtc-pc and JSEP's signaling and transceiver states the same?

- No, if so pc.signalingState and pc.getTransceivers() would be racy APIs.

Still, might addTrack() surface an SRD-created JSEP-transceiver to JS?
What do we do today?

- Chrome: SRD creates the transceiver, addTrack exposes it.
  - Well-defined behavior, but the JS thread is blocked on "in-parallel" work!
- Firefox: Mutex access to shared transceiver objects?
  JS and background thread races. addTrack MAY expose SRD-transceiver.
  - Non-blocking, but racy!

Problems: Compatibility concerns and races.
In both cases: "single threaded" principles violated?

Issue 2502: When are effects of in-parallel stuff surfaced? (Henrik)

Proposal A:

- addTrack() should determine whether to create or recycle a transceiver based on JS thread's *set of transceivers*.

  ⇒ addTrack() is not dependent on JSEP states and has no risk of surfacing in-parallel created transceiver objects.

I believe the spec already says this, but it would be good to clarify.
Both Chrome and Firefox are non-compliant.

Proposal B:

- Update the spec to allow addTrack() to add to JS's *set of transceivers* a JSEP transceiver that has not yet been exposed.
- Make it clear that which transceiver is obtained is racy, implementation-specific and that this non-"single threaded" behavior is on purpose. :(

Issues for Discussion Today

- Media Capture and Streams
  - Issue 671: New audio acquisition constraints
  - Issue 639: Enforcing User Gesture for getUserMedia (Youennf)
  - Issue 640: Only reveal labels of device user has given permission to (Youennf)
  - Issue 669: "user-chooses": Does required constraints make any sense now? (Henrik)

Issue 671: New audio acquisition constraints (Sam Dallstream)

- Problem Statement:
  The specification, as it stands today, does not provide enough specificity within constraints (noiseSuppression, echoCancellation) to allow developers to differentiate streams for speech recognition and communication.

- noiseSuppression, echoCancellation, autoGainControl, are only on/off
- Current implementations are geared towards communications.
- Communications modifications generally hurt speech recognition and vice versa.
- Testing current constraints is hard (besides getters/setters). More specific constraints can start to conform to relevant ETSI standards. (communication, speech recognition (Draft ETSI TS 103 504))

Issue 671: New audio acquisition constraints - Technical Details

Examples of areas where speech recognition streams differ from communications streams.

	Communications Audio	Speech Audio
Echo (Suppression)	Echo leakage is intolerable to human listeners >40 dB speakerphone, >46 dB handsets, ITU-T TS 26.131	Echo leakage is tolerable with sufficient speech level Typically >15-20 dB speech to echo
Echo (Switching)	Switching (slight loss of initial syllables) is used to avoid echo leakage, and slight impairments are not noticeable to human listeners ITU-T P.501, P.502, TS 26.131, P.1100, G.131	Any switching resulting in slight loss or attenuation of syllables impairs barge-in and introduces word error rate STQ63-250 Section 5.2
Ambient Noise	Focused on perceived quality/distraction from speech/noise. Some ambient noise is tolerable is it provides contextual cues in human perception without distraction Noise sources diffuse ITU-T TS 26.131, P.835, ETSI EG 202 396	Not concerned with human perception but preservation of source utterance and removal of background noise Noise sources diffuse + discrete in test to evaluate rejection nearby non-users and noises STQ 63-250 Section 4.2 Alexa Acoustic Testing V3.5.6

Issue 671: New audio acquisition constraints - Technical Details

Room Acoustics (Reverb)	Typically ok to have some amount of reverb, as audio provides contextual cues in human perception without distraction	Devices concerned with preservation of source utterance. Room reflections introduce loss of information in frequency regions and phonemes
Comfort Noise	Desired to avoid perception that call has dropped ITU-T G.711	Undesirable to add any additional noise that impairs source utterance
Sound Quality	Sensitive to human perception ITU-T P.863	Sensitive to preserving speech formants STQ 63-250 Section 6 Alexa Acoustic Testing V3
Level/Gain Control	Standardized to match cross-network communications architectures ITU-T G.111, G.121, TS 26.131	Dependent on trained model, often less sensitive

Issue 671: New audio acquisition constraints - Proposal

Proposal #1

- Add a new constraint “category” that takes one of four values: “default”, “raw”, “communication”, “speech” (or “speechRecognition”).
- Link to explainer

Pros

- Fits well into existing constraint model.
- Straightforward translation to implementation on multiple platforms.

Cons

- Competes with existing content-hint draft in a non-productive / confusing way.
- Could have required interactions with other constraints.

Issue 671: New audio acquisition constraints - Proposal

Proposal #2

- Modify existing constraints like echoCancellation to be more specific.
- Add new hints to content-hint draft (communication, speechRecognition).

Pros

- Fits in well with content-hint draft.
- Allows for (possibly) more developer freedom.

Cons

- Implementation route is currently not as clear.

Issue 639: Enforcing user gesture for getUserMedia (Youenn)

- Problem: getUserMedia should only be callable on user gesture
  - Most modern APIs add such restrictions
  - This is not web compatible
- Can we start shipping such restrictions in getUserMedia?
- Proposal: require a user gesture to grant access without a prompt
  - PR: https://github.com/w3c/mediacapture-main/pull/666
- Other ideas
  - Require a user gesture past initial page load
  - Require a user gesture once a previous call to getUserMedia is denied for the page (implemented in Safari)
    - If a call was denied for the current page, Safari will deny all further calls, except if call is made as part of a user gesture, in which case it will prompt
    - Persistent denying stays denied

Issue 640: Only reveal labels of devices user has given permis.. (Youenn)

If difficult to enforce per-device exposure rule, enforce per-device-type exposure

- Expose all microphones if one microphone is granted
- Expose all cameras if one camera is granted
- Do not expose speakers once output speaker picker API is available

Still possible to use `groupId` to get microphone corresponding to camera

Issue 669: "user-chooses": Do required constraints make any sense now? (Henrik)

Constraints are powerful for optimizing resolution, frame rate and other properties.
But **“required”** constraints also remove devices from the selection.

- Does this make any sense if **“user-chooses”**? Example: SD camera faces me, HD camera faces my room. Application opens HD camera because it requires HD, but that’s not what I want! If we have a picker, why not let the user pick?

But I want to avoid re-prompt when user re-visits my website!

- How about {quickJoin:true} that defaults the choice if it’s been selected before and devices have no changed? The app doesn’t even need to remember deviceId.

But I want in-content picker!

In-content selection only works well when exposing all device labels to the application.
… wasn’t one of the points of “user-chooses” to stop leaking device labels?

Issue 669: "user-chooses": Do required constraints make any sense now? (Henrik)

Proposal A: Continue to support in-content picker in “user-chooses”, but prefer not to…

- deviceId can be required, all other constraints are treated as optional if they would result in decreasing the set of devices.
- In order to support in-content picker, deviceId and labels of ALL devices have to be exposed when permission is granted to one device.

Proposal B: Partial in-content picker support in “user-chooses”.

- Like Proposal A except in-content can only pick between devices UA has exposed; other devices only available through re-prompt.
- (Discussion: How would the application know if there were other devices available so that it knows whether to have an “other (re-prompt)” option or not?)

Proposal C: Don’t support in-content picker in “user-chooses”.

- Any device selection is necessarily done by the user. Labels only exposed for current track.
- Avoiding re-prompt is supported by {quickJoin:true}, not by requiring deviceId.

Issue 672: Deprecate inputDeviceInfo.getCapabilities() for privacy (jib)

Chrome/Edge & Safari have `info.getCapabilities()` w/info on all devices after gUM.

Reason: Lets site enforce its constraints while building picker, or choosing other device outright. Most sites enforce some constraints. **But:** It's a trove of fingerprinting info!

`"user-chooses"` provides feature-parity, without the information leak:

```
await navigator.mediaDevices.getUserMedia({video: constraints, semantics: "user-chooses")
```

So once #667 merges, can we deprecate `info.getCapabilities()`?

For extra credit

Name that bird!

Thank you

Special thanks to:

WG Participants, Editors & Chairs

The bird