W3C WebRTC WG Meeting

June 18, 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 June 2024 interim meeting of the W3C WebRTC WG, at which we will cover:
  - WebRTC-PC revision process, mediacapture-extensions, IceController API, WebRTC-PC and mediacapture-main
- Future meetings:
  - July 16 (Should we cancel this?)
  - August 27

About this Virtual Meeting

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/June_18_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 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 (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 - 08:30 AM WebRTC-PC revision process (Dom)
- 08:30 - 08:50 AM Mediacapture-Extensions (Jianjun, Youenn)
- 08:50 - 09:20 AM IceController API (Sameer)
- 09:20 - 09:40 AM WebRTC-PC and mediaCapture-main (Jan-Ivar)
- 09:40 - 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 Revision Process & New Charter (Dom)

Start Time: 08:10 AM

End Time: 08:30 AM

Proposal: merging more extensions in WebRTC Rec

- webrtc-extensions collects possible significant new additions to the main WebRTC spec
- Our current policy is to merge these features in the main spec when:
  - Reasonable test coverage
  - **Two passing implementations**
- But webrtc-extensions is not a spec; features there tend to have less visibility and create confusion
- webrtc-charter#83 proposes to migrate features earlier, with only **one implementation and one implementation commitment**

WebRTC Charter Renewal

- Current charter expires end of September
- Proposal is to renew charter as is (with editorial updates)
  - https://w3c.github.io/webrtc-charter/webrtc-charter.html

Discussion (End Time: 08:30)

-

Mediacapture-Extensions (Jianjun, Youenn)

Start Time: 08:30 AM

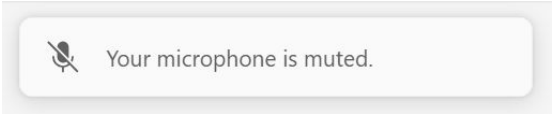
End Time: 08:50 AM

For Discussion Today

- Issue 145: Consider adding onVoiceActivity event on MediaStreamTrack for audio (Jianjun Zhu)
- Issue 149: How to select camera presets that have better power efficiency at the expense of quality? (Youenn Fablet)

Issue 145: Consider adding onVoiceActivity event on MediaStreamTrack for audio (Jianjun Zhu)

- Motivations:
  - A hint to unmute microphone



  - Process audio only when user is actively speaking

- Proposal:
  - Add `voiceactivitystart` and `voiceactivityend` events on MediaStreamTrack for audio.
  - Enabled iff constraints/settings for VAD enabled on the track.

Issue 149: How to select camera presets that have better power efficiency at the expense of quality? (Youenn Fablet)

- UA needs to select a camera preset when starting capture
  - UA MAY resize to fully match web application constraints

- Some camera presets may be more power efficient than others
  - videoBinned formats on iOS
  - Potential consequences in terms of quality

- Allow web applications to hint UA at selecting power efficient formats

- We already have powerEfficientPixelFormat.

Issue 149: How to select camera presets that have better power efficiency at the expense of quality? (Youenn Fablet)

- Option 1
  - Do nothing, UA can use powerEfficientPixelFormat as a hint towards using power efficient camera presets
- Option 2
  - Create a different constraint
  - Or make powerEfficientPixelFormat more general

Discussion (End Time: 08:50)

-

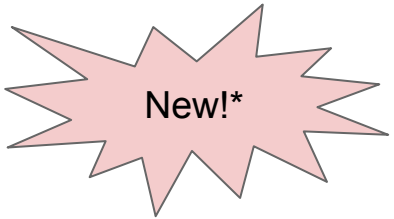
IceController API (Sameer)

Start Time: 08:50 AM

End Time: 09:20 AM

IceController road map

- Prevent removal of candidate pairs
- Remove candidate pairs
- Control selection of candidate pair
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking



* as of TPAC 2023

Purpose of ICE checks *(not exhaustive)*

- Check connectivity over a candidate pair
- Discover reflexive candidates
- Learn reflexive candidate priority
- Determine ICE role
- Determine the nominated candidate pair
- TURN allocation, permission, data exchange
- Keepalives (NAT, TURN)
- Determine RTT

+

Conserve bandwidth / power

What does the proposed API allow

- Insights into ICE checks and responses
  - when and where are checks sent
  - prevent checks from being sent *(sometimes)*
    - …in which case ICE Agent defers until next instance
    - can't prevent triggered checks, nominations
  - when a check response is received
  - when a check times out
- Send an ICE check on a candidate pair
- **Not possible with the new API:**
  - prevent responses from being sent
  - craft the actual STUN packet
  - bypass security mitigations in the ICE agent, eg. rate-limiting

Issue 209 - Options

- Linked Promises *(presented at TPAC)*
  - ICE check event before an ICE check is sent
    - may be canceled
  - ICE check event & check() yield promise of an ICE check request
  - ICE check request yields promise of an ICE check response

- Flat events
  - ICE check event before an ICE check is sent
    - may be canceled
  - ICE check conclusion event after response, timeout, or error
  - More *(and lighter)* events, but a few missing details
    - Was an ICE check initiated by the app or the ICE agent
    - Event fired only after check concludes, not after check sent

IDL - Linked Promises or Flat Events

Linked Promises

```
partial interface RTCIceTransport {
  // Send an ICE check.
  Promise<RTCIceCheckRequest> checkCandidatePair(RTCIceCandidatePair pair);
  // Fired before ICE agent sends an ICE check.
  // Cancellable, unless triggered check or nomination or app initiated.
  attribute EventHandler /* RTCIceCheckEvent */ onicecandidatepaircheck;
}

interface RTCIceCheckEvent : Event {    // Cancellable
  readonly attribute RTCIceCandidatePair candidatePair;
  // Resolves when the check is actually sent. Rejected => send failure.
  readonly attribute Promise<RTCIceCheckRequest> request;
}

interface RTCIceCheckRequest {
  readonly attribute ArrayBuffer transactionId;
  readonly attribute DOMHighResTimeStamp sentTime;
  // Resolves when response is received. Rejected => timeout.
  readonly attribute Promise<RTCIceCheckResponse> response;
}

interface RTCIceCheckResponse {
  readonly attribute DOMHighResTimeStamp receivedTime;
  // No error => success.
  readonly attribute RTCIceCheckResponseError? error;
}
```

Flat Events

```
partial interface RTCIceTransport {
  // Send an ICE check.
  Promise<undefined> checkCandidatePair(RTCIceCandidatePair pair);
  // Fired before ICE agent sends an ICE check.
  // Cancellable, unless triggered check or nomination or app initiated.
  attribute EventHandler /* RTCIceCandidatePairEvent */
onicecandidatepaircheck;
  // Fired when an ICE check concludes.
  attribute EventHandler /* RTCIceCheckEvent */
onicecandidatepaircheckcomplete;
}

interface RTCIceCheckEvent : Event {
  readonly attribute RTCIceCandidatePair candidatePair;
  readonly attribute ArrayBuffer transactionId;
  readonly attribute DOMHighResTimeStamp sentTime;
  // No receivedTime => timeout.
  readonly attribute DOMHighResTimeStamp? receivedTime;
  // No error => success.
  readonly attribute RTCIceCheckResponseError? error;
}
```

Usage - Linked Promises or Flat Events

Linked Promises

```
const pc = ...;
const ice = pc.getTransceivers()[0].sender.transport.iceTransport;

ice.onicecandidatepaircheck = async(event) => {
   if (shouldNotCheck(event.candidatePair)) {
       event.preventDefault();    // prevent a check
       return;
   }
   const request = await event.request;
   handleCheck(request);
}
const request = await ice.checkCandidatePair(alternatePair);// send a check
handleCheck(request);

function handleCheck(request) {
   try {
       const response = await request.response;
       const rtt = response.receivedTime - request.sentTime;
       // ... do something with rtt ...
       if (response.error) {
           // ... do something with error ...
       }
   } catch(error) {
       // ... do something with timeout ...
   }
}
```

Flat Events

```
const pc = ...;
const ice = pc.getTransceivers()[0].sender.transport.iceTransport;

ice.onicecandidatepaircheck = (event) => {
   if (shouldNotCheck(event.candidatePair)) {
       event.preventDefault();     // prevent a check
   }
}

ice.onicecandidatepaircheckcomplete = handleCheck;

ice.checkCandidatePair(alternatePair);    // send a check

function handleCheck(event) {
   if (event.receivedTime) {
       const rtt = event.receivedTime - event.sentTime;
       // ... do something with rtt ...
   }
   else {
       // ... do something with timeout ...
   }
   if (event.error) {
       // ... do something with error ...
   }
}
```

Discussion (End Time: 09:20)

-

Webrtc-pc & mediacapture-main (Jan-Ivar)

Start Time: 09:20 AM

End Time: 09:40 AM

For Discussion Today

- WebRTC-pc
  - Issue 2977: PC.local_description and friends - snapshot views or dynamic views?

- Mediacapture-main:
  - Issue 966 / PR 1007: Should devicechange fire when the device info changes?

Issue 2977: PC.local_description and friends - snapshot views or dynamic views?

The spec says:

```
const x = pc.localDescription, y = x.sdp;
console.log(pc.localDescription == x);   // true
await pc.addCandidate(candidate);
console.log(pc.localDescription == x);   // true
console.log(x.sdp == y);                 // false (x is aaaalive!)
```

But implementations do this instead:

```
const x = pc.localDescription, y = x.sdp;
console.log(pc.localDescription == x);   // true
await pc.addCandidate(candidate);
console.log(pc.localDescription == x);   // false
console.log(x.sdp == y);                 // true (x is a snapshot)
```

Proposal: align with implementations.

Issue 966 / PR 1007: Should devicechange fire when the device info changes?

Last meeting: No new `deviceinserted` event. Instead “extend the existing event”.

“not sure if a boolean is enough, we'll give a bit more thinking”

Draft **PR 1007**:

WebIDL

```
[Exposed=Window]
interface DeviceChangeEvent : Event {
  constructor(DOMString type, optional DeviceChangeEventInit eventInitDict = {});
  [SameObject] readonly attribute FrozenArray<MediaDeviceInfo> devices;
  [SameObject] readonly attribute FrozenArray<MediaDeviceInfo> userInsertedDevices;
};
```

PR 1007: Add userInsertedDevices attribute to DeviceChangeEvent

***userInsertedDevices* of type FrozenArray<MediaDeviceInfo>, readonly**

The `userInsertedDevices` attribute returns an array containing only those `MediaDeviceInfo` objects from `devices` that the user physically inserted or activated recently and are newly exposed with this event as a result. Otherwise, an empty list is returned.

The User Agent *MAY* include devices the user inserted or activated before `getUserMedia()` was called, provided this event marks their first exposure, and the user did not choose devices in `getUserMedia()`.

The `MediaDeviceInfo` objects, if any, *MUST* also exist in `devices`.

> NOTE
>
> A user inserting a device during (or immediately ahead of) a call can be a strong signal that they wish to use the device immediately.
>
> Applications are encouraged to rely on this attribute to disambiguate this signal from differences in `devices` that might happen from changes in device information exposure.

Discussion (End Time: 09:40)

-

Wrapup and Next Steps

Start Time: 09:40 AM

End Time: 10:00 AM

Next Steps

- Content goes here

Thank you

Special thanks to:

WG Participants, Editors & Chairs