W3C WebRTC WG Meeting

June 28, 2016 1:00 PM - 2:30 PM PDT

Chairs: Harald Alvestrand

Stefan Hakansson

Erik Lagerway

W3C WG IPR Policy

- This group abides by the W3C patent policy https://www.w3.org/Consortium/Patent-Policy-20040205
- 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 some outstanding issues before transition to CR
- Editor’s Draft update to follow meeting

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/June_28_2016
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is being recorded.
- WebEx info here

For Discussion Today

- **Media Capture and Streams**
  - **Issues**
    - Issue 350: New permission definitions are wrong (Harald)
    - Issue 359: MUST clear requirement for deviceId (Stefan)
- **WebRTC 1.0 API**
  - **Pull Requests**
    - Issue 644/PR 675: Attribute to turn on/off CN/DTX (Bernard Aboba)
    - Issue 548/PR 647/PR 683: RTX/RED/FEC handling (Bernard Aboba)
    - Issue 706/PR 787: How does setDirection interact with active/inactive sender/receivers? (Bernard Aboba)
  - **Issues**
    - Issue 253:Assurance that requests to IdP proxy originate from the user agent (Martin Thomson)
    - Issue 555: Sort out requirements around IdpLoginError (Martin Thomson)
    - Issue 562: What to do with an RTCIdentityProvider that returns rubbish (Martin Thomson)
    - Issue 678: Support assertions that identify the recipient (Cullen Jennings)
    - Issue 685: JSEP Reference for Receipt of Multiple RTP Encodings (Bernard Aboba)
    - Issue 692: Meaning of “Liveness Checks Have Failed” (Taylor)
    - Issue 700: An event for when a circuit breaker is triggered (Varun Singh)

Media Capture

- **Issues**
  - Issue 350: New permission definitions are wrong (Harald)
  - Issue 359: MUST clear requirement for deviceId (Stefan)

Issue 350: New permission definitions are wrong

- Recap: Why we linked to permissions
  - Own description was growing complex
  - Would like a common permission handling framework
  - Recognized area as difficult, wanted to stabilize spec
- Problems encountered (see bug++):
  - Lack of clarity in our spec: Persistent vs temp permissions
  - Lack of clarity in our spec: getUserMedia(id=a) twice = OK?
  - Lack of clarity in our spec: Revocation = ?
  - Moving target in permissions spec: “request permission” ++

What is a temporary grant? (MC)

Consider:

getUserMedia(id=123, …) => stream A

- Prompts, (temp) access granted, returns

getUserMedia(id=123, ...) => stream B

- Prompt or not?

close(A.tracks); close(B.tracks); getUserMedia(id=123, …) => stream C

- Prompt or not?

A: current spec: Do NOT prompt for B. DO prompt for C.

B: “there is no temporary”. Do NOT prompt for B, C or next session’s call.

C: “each call is an island”. Prompt for A, B and C. JS can use A.clone().

Revocation - what happens then? (P)

- Hook present in permissions spec, devices are not using it yet.
- Alternatives:
  - A - Do nothing
  - B - stop all tracks sourced from this device
- Suggest B
  - This can be done in the Permissions spec

Choosing from possible devices

- Current: Constraints reduce choice set, asking for permission picks one from the set
- Permissions added a “prompt the user” method (“request” can’t do this)
- Requires us to specify what’s passed across
- Proposal: Use “prompt the user” algorithm
  - Pass an ID list, sorted on fitness distance
  - UA is free to do whatever makes sense

Other issues

- Please read the permissions spec!
  - https://w3c.github.io/permissions
- Please file bugs appropriately!

Issue 359: MUST clear requirement for deviceId

- This deals with the situation when access to (microphone or camera) has **never***) been granted to the origin
- The spec says that all *deviceId*s MUST be cleared at the end of the current browsing session
  - ‘Current browsing session’ means until tab is closed or user navigates to another origin
  - ‘MUST be cleared’ added after PING review - otherwise there would be a perfect fingerprint by just doing enumerateDevices()
- In #359, it is proposed to change to MAY be cleared
  - Motivation: ‘similar to cookies’, ‘site architecture’, ‘complexity’
- Chair message sent to the list: https://lists.w3.org/Archives/Public/public-media-capture/2016Jun/0046.html
  - Message notes that proposed change could create a privacy issue and would break an agreement with PING
  - No responses so far
- Can we decide to not change the spec?

*) Really: not since last time the user cleared cookies and other persisted data for the origin

WebRTC PC

- **Pull Requests**
  - **Issue 644**/**PR 675**: Attribute to turn on/off CN/DTX (Bernard Aboba)
  - **Issue 548**/**PR 647**/**PR 683**: RTX/RED/FEC handling (Bernard Aboba)
  - **Issue 706**/**PR 787**: How does setDirection interact with active/inactive sender/receivers? (Bernard Aboba)

Issue 644/PR 675: Turn DTX on/off

- Problem statement:
  - For RtpSender, enable internal/external DTX to be turned on/off
    - Setting takes effect immediately (negotiation-needed flag not set).
    - Note from Stefan:
      - AMR/AMR-WB default implementations run with DTX always on
    - Question from Harald:
      - Can DTX only be turned on if internal/external comfort noise is negotiated?
      - Or can we have DTX on without internal/external comfort noise negotiated?
  - For RtpReceiver, need to control decoding of internal/external comfort noise?
    - Or should decoder always decode internal/external comfort noise if sent by an RtpSender?

Issue 644/PR 675: Turn DTX on/off (cont'd)

- Proposal:

```
partial dictionary RTCRtpEncodingParameters {
    RTCDtxStatus dtx;
}
enum RTCDtxStatus {
     "disabled", //DTX will be disabled
     "enabledIfNegotiated" //DTX will be enabled only if negotiated
}
```

- `dtx` can only be set on an RtpSender (no setParameters() method on an RtpReceiver).
- Question: Value of receiver.getParameters().encodings[].dtx
  - Unset or `"enabledIfNegotiated"`?

Issue 548/PR 647/ RTX/RED/FEC Handling

- Question: Are RTX/RED/FEC treated as codecs within RTCRtpCapabilities and RTCRtpParameters?
- Current status:
  - RTX/RED/FEC included in RTCRtpCapabilities.codecs[]
    - Single entry for “rtx”, “red”, “ulpfec”, “flexfec”, etc.
    - In WebRTC 1.0, getCapabilities(*kind*).codecs[] most useful to enable setting of codec preferences
    - `setCodecPreferences(sequence<RTCRtpCodecCapability> codecs);`
    - Impacts `createOffer` and `createAnswer`
  - RTX/RED/FEC included in RTCRtpParameters.codecs[]
    - Single entry for “red”, “ulpfec”, “flexfec”
    - Multiple entries for “rtx” (one for each codec that is being retransmitted)
      - Entries differ in the value of .payloadType, .sdpFmtpLine (“apt” and “rtx-time” parameters)
      - Possible to indicate if an implementation supporting “rtx”, “red” and “ulpfec” supports retransmission of red/ulpfec.
      - More capability information available from getParameters() than getCapabilities()
- PR 647: clarifies existing use of “rtx” in RTCRtpCapabilities/RTCRtpParameters

PR 683: Add RTCRtpCodecRtxParameters dictionary

- Proposal
  - Adds `RTCRtpCodecRtxParameters` to the `RTCRtpCodecParameters` dictionary
  - Removes “rtx” from both `RTCRtpParameters.codecs`[] and `RTCRtpCodecCapabilities.codecs[]`
- WebIDL

  -
    ```
    partial dictionary RTCRtpCodecParameters {
          RTCRtpCodecRtxParameters rtx;
    };
    ```

  -
    ```
    dictionary RTCRtpCodecRtxParameters {
          unsigned short payloadType; // PT used for retransmission of this codec
          unsigned long rtxTime;
    };
    ```

Issue 706: How does setDirection interact with active/inactive sender/receivers?

- JSEP Section 5.2.4 (Direction Attribute in Offers):

  [RFC3264] direction attributes (defined in Section 6.1) in offers are chosen according to the states of the RtpSender and RtpReceiver of a given RtpTransceiver, as follows:

RtpSender	RtpReceiver	offer direction
active	active	sendrecv
active	inactive	sendonly
inactive	active	recvonly
inactive	inactive	inactive

Question: What does “states of the RtpSender/Receiver” refer to?

Issue 706: How does setDirection interact with active/inactive sender/receivers? (cont'd)

- WebRTC API Section 5.1:
  - `sendrecv:` The `RTCRtpTransceiver`'s `RTCRtpSender` will offer to send RTP, and will send RTP if the remote peer accepts, in which case `active` is set to "true". The `RTCRtpTransceiver`'s `RTCRtpReceiver` will offer to receive RTP, and will receive RTP if the remote peer accepts, in which case `active` is set to "true".
  - `sendonly:` The `RTCRtpTransceiver`'s `RTCRtpSender` will offer to send RTP, and will send RTP if the remote peer accepts, in which case `active` is set to "true". The `RTCRtpTransceiver`'s `RTCRtpReceiver` will not offer to receive RTP, and will not receive RTP (`active` set to "false").
  - `recvonly:` The `RTCRtpTransceiver`'s `RTCRtpSender` will not offer to send RTP, and will not send RTP (`active` set to "false"). The `RTCRtpTransceiver`'s `RTCRtpReceiver` will offer to receive RTP, and will receive RTP if the remote peer accepts, in which case `active` is set to "true".
  - `inactive:` The `RTCRtpTransceiver`'s `RTCRtpSender` will not offer to send RTP, and will not send RTP (`active` set to "false"). The `RTCRtpTransceiver`'s `RTCRtpReceiver` will not offer to receive RTP, and will not receive RTP (`active` set to "false").
- Question: What is `active` referring to here??

Issue 706: How does setDirection interact with active/inactive sender/receivers? (cont'd)

- WebRTC API Section 5.4:
  - The direction attribute [of an RTCRtpTransceiver] indicates the direction of this transceiver. The value of direction is independent of the value of encodings[]. active since one cannot be deduced from the other. If the `stop()` method is called, direction retains the value it had prior to calling `stop()`.
  - The `setDirection` method sets the direction of the `RTCRtpTransceiver`. Future calls to `createOffer` and `createAnswer` mark the corresponding media description as `sendrecv`, `sendonly`, `recvonly` or `inactive` as defined in [JSEP] (section 5.2.2. and section 5.3.2.). Calling `setDirection()` sets the negotiation-needed flag.

PR 683: Add RTCRtpCodecRtxParameters dictionary (cont'd)

- Impact
  - Removing "rtx" from `RTCRtpCodecCapabilities.codecs[]` would remove ability to control "rtx" via `setCodecPreferences()`.
  - Removal from `RTCRtpCodecParameters.codecs[]` implies loss of codecs[]. sdpFmtpLine
    - Equivalent info present in **`RTCRtpCodecRtxParameters`**
    - Alternative: codecs[].sdpRtxFmtpLine

Issue 706: How does setDirection interact with active/inactive sender/receivers? (cont'd)

- Explanation (from Peter):
  - RtpSender/RtpReceiver controls don't affect signaling/negotiation (or vice versa)
  - RtpSender.encodings[i].active controls sending without affecting signaling/negotiation
  - RtpTransceiver.setDirection() controls sending/receiving and affects signaling/negotiation
  - RtpSender sends iff RtpTransceiver.direction is "sendrecv" or "sendonly" and RtpSender.encodings[i].active is true
  - RtpReceiver receives iff RtpTransceiver.direction is "sendrecv" or "recvonly" (RtpReceiver.encodings[i].active can't be set to false)
- Proposal:
  - Remove references to "setting active" in WebRTC 1.0 Section 5.1.
  - Update JSEP Section 5.2.4 to refer to setDirection().

Issue 706: How does setDirection interact with active/inactive sender/receivers? (cont'd)

- Proposed revision of WebRTC API Section 5.1:
  - `sendrecv:` The `RTCRtpTransceiver`'s `RTCRtpSender` will offer to send RTP, and will send RTP if the remote peer accepts and RTCRtpSender.encodings[i].`active` is set to "true". The `RTCRtpTransceiver`'s `RTCRtpReceiver` will offer to receive RTP, and will receive RTP if the remote peer accepts.
  - `sendonly:` The `RTCRtpTransceiver`'s `RTCRtpSender` will offer to send RTP, and will send RTP if the remote peer accepts and RTCRtpSender.encodings[i].`active` is set to "true". The `RTCRtpTransceiver`'s `RTCRtpReceiver` will not offer to receive RTP, and will not receive RTP.
  - `recvonly:` The `RTCRtpTransceiver`'s `RTCRtpSender` will not offer to send RTP, and will not send RTP. The `RTCRtpTransceiver`'s `RTCRtpReceiver` will offer to receive RTP, and will receive RTP if the remote peer accepts.
  - `inactive:` The `RTCRtpTransceiver`'s `RTCRtpSender` will not offer to send RTP, and will not send RTP. The `RTCRtpTransceiver`'s `RTCRtpReceiver` will not offer to receive RTP, and will not receive RTP.

WebRTC PC

- **Issues**
  - Issue 253:Assurance that requests to IdP proxy originate from the user agent (Martin Thomson)
  - Issue 555: Sort out requirements around IdpLoginError (Martin Thomson)
  - Issue 562: What to do with an RTCIdentityProvider that returns rubbish (Martin Thomson)
  - Issue 678: Support assertions that identify the recipient (Cullen Jennings)
  - Issue 685: JSEP Reference for Receipt of Multiple RTP Encodings (Bernard Aboba)
  - Issue 692: Meaning of “Liveness Checks Have Failed” (Taylor)
  - Issue 700: An event for when a circuit breaker is triggered (Varun Singh)

Issue 253: Assurance that requests to IdP proxy originate from the user agent

- PR: https://github.com/w3c/webrtc-pc/pull/719
- Read and comment on github

Issue 555: Sort out requirements around IdpLoginError

- RTCPeerConnection is configured with an IdP
- User is not logged in with that IdP
    - ...or the IdP requires some additional information from the user
- Current API is bad
    - IdP throws, but this is not an exceptional situation
    - It uses a custom DOMException name, which is bad form

Issue 555: … IdpLoginError

- Option 1:Overload return value
  - if typeof(assertionResult) === ‘string’, then login is needed:

```
+ union (RTCIdentityAssertionDetails or RTCIdentityLoginUrl) RTCIdentityAssertionResult;
+ typedef DOMString RTCIdentityLoginUrl;

+ dictionary RTCIdentityAssertionDetails {
- dictionary RTCIdentityAssertionResult {
     required RTCIdentityProviderDetails idp;
     required DOMString                  assertion;
 };

callback GenerateAssertionCallback =
    Promise<RTCIdentityAssertionResult> (DOMString contents, DOMString origin,
                                         optional DOMString usernameHint);
```

Issue 555: … IdpLoginError

- Option 2: extra arguments to callback

```
+ callback RTCIdentityLoginNeeded void (DOMString loginUrl);

  callback GenerateAssertionCallback =
      Promise<RTCIdentityAssertionResult> (DOMString contents, DOMString origin,
-                                          optional DOMString usernameHint,
+                                          RTCIdentityLoginNeeded loginCallback,
+                                          optional DOMString usernameHint);
```

- Option 2 is hard to make backward-compatible
- Pick option 1

Issue 562: What to do with an RTCIdentityProvider that returns rubbish

- PR: https://github.com/w3c/webrtc-pc/pull/716
- Read and comment on github

Issue 678: Support Assertions that Identify the Recipient

- Some identity systems only identify the sender
  - This has the weakness that the assertion can be replayed to many different receivers (whether this is a problem depends on context)
- Some identity systems (such as IETF STIR) include the identity of the intended receiver
- Inelegant solutions exist with usernameHint or protocol, but they aren't clean
- Proposal: Support both types by extending setIdentityProvider and GenerateAssertionCallback to allow an optional peerIdentity argument
  - No real impact on any browser code, just extends API interface
  - Aligns WebRTC with STIR and other identity systems
  - Should we populate this with RTCConfiguration.peerIdentity if no value for this argument is provided?

Issue 685: JSEP Reference for Receipt of Multiple RTP Encodings

- In WebRTC 1.0 Section 5.1 it says:

'When setRemoteDescription is called with a corresponding remote description that is able to receive multiple RTP encodings as defined in [JSEP], the RTCRtpSender may send multiple RTP encodings and the parameters retrieved via the transceiver’s sender.getParameters() will reflect the encodings negotiated.'

- Problem:
  - draft-ietf-mmusic-rid Section 6.2.2 describes the SDP used to indicate ability to receive multiple RTP encodings, but JSEP does not include a discussion of this.
  - Possible explanation: SDP indicating ability to receive multiple RTP encodings not emitted by a browser, only by Selective Forwarding Unit (SFU).
- Suggested resolution:
  - Add discussion in JSEP

Issue 692: Meaning of “Liveness Checks” Have Failed

- RTCIceTransportState’s “disconnected” state definition says:

  ”Liveness checks have failed. This is more aggressive than `failed`, and may trigger intermittently (and resolve itself without action) on a flaky network.”
- Questions
  - What is a liveness check? Is it a STUN consent check?
  - "Checks" is plural. So how many checks must fail before entering `disconnected`? Does the state return to `connected` when the next check succeeds?

Issue 692: Meaning of “Liveness Checks” Have Failed (cont’d)

- Recommendation
  - Change “liveness check” to “STUN consent check” [RFC7675]
  - How many checks must fail before entering `disconnected`?
    - No guidance in either RFC 5245bis (which doesn’t cover consent) or RFC 7675 (which only discusses consent failure)
    - Is this therefore implementation dependent?
  - Does the state return to `connected` when the next check succeeds?
    - Yes. That is consistent with “may trigger intermittently (and resolve itself without action)”.

Issue 700: An Event for when a Circuit Breaker is Triggered

- Question
  - Has any browser implemented circuit breakers or are there plans to implement circuit breakers?
- If "yes", how does the implementation behave?
  - draft-ietf-avtcore-rtp-circuit-breakers Section 4.5:

The intention is that the application will stop sending RTP data packets on a particular 5-tuple (transport protocol, source and destination ports, source and destination IP addresses), until whatever network problem that triggered the RTP circuit breaker has dissipated. This could mean stopping a single RTP flow, or it could mean that multiple bundled RTP flows are stopped. RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated, or can reasonably be expected to have dissipated.

Issue 700: An Event for when a Circuit Breaker is Triggered (cont'd)

- Questions:
  - Is the effect of triggering the circuit breaker visible to the application?
    - Via an event?
    - Via change in the value of an attribute?
  - Can the implementation automatically resume sending?
    - When congestion abates?
    - When the ICE selected pair changes?
  - Is there something the application can or should do on triggering the circuit breaker (e.g. ICE restart)?

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs