W3C WebRTC WG Meeting

June 28, 2017
8 AM PDT

Chairs: Stefan Hakansson
Bernard Aboba
Harald Alvestrand (Emeritus)

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 issues arising from the CR review of webrtc-pc
- Editor’s Draft updates to follow meeting

webrtc-pc

- The CR review completed on May 31.
- We are targeting “Issue clusters” in this meeting - hoping it will be fruitful.
- Reminder: we are continuing to solicit feedback from implementers on features not on their radar for implementation.

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/June_28_2017
- Link to latest drafts:
  - https://rawgit.com/w3c/mediacapture-main/master/getusermedia.html
  - https://rawgit.com/w3c/webrtc-pc/master/webrtc.html
  - https://rawgit.com/w3c/webrtc-stats/master/webrtc-stats.html
- 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

- **WebRTC-PC Issues**
  - **offerToReceive (Jan-Ivar)**
    - Issue 1361: Clarify offerToReceiveAudio and offerToReceiveVideo in renegotiation
    - Issue 1383: offerToReceive legacy behavior does not match implementations
  - **Objects (Taylor)**
    - Issue 1174: ssrc in RTCRtpEncodingParameters is inconsistent with ORTC
    - Issue 1178: When IceTransport and DtlsTransport objects are created/changed
    - Issue 1306: Obtaining a list of all DTLS and ICE transports
    - Issue 1365: No getStats() for DTLS and ICE transports
    - Issue 1406: When ICE restart results in connection to a new endpoint
    - Issue 1413: Should even/odd id validation be enforced in RTCDataChannel when negotiated is “true”?
    - Issue 1415: When is the dtmf attribute in RTCRtpSender set?
    - Issue 1423: What if a datachannel “OPEN” message uses unknown priority?
    - Issue 1424: What happens if SRD has two tracks with the same ID?
  - **Miscellaneous (Bernard)**
    - Issue 1259: What keygenAlgorithm values are supported?
    - Issue 1215: Rollback: a feature “at risk”?
    - Issue 1283: Centering, Scaling, Cropping
    - Issue 763/1323: Handling of encoding/decoding errors

offerToReceive (Jan-Ivar)

- Issue 1383: offerToReceive legacy behavior does not match implementations
- Issue 1361: Clarify offerToReceiveAudio and offerToReceiveVideo in renegotiation

Issue 1383 / PR 1430: offerToReceive legacy behavior does not match implementations (Jan-Ivar)

- Call createOffer({offerToReceiveAudio:true}) and then createOffer({offerToReceiveVideo:true}) without SLD.
- Spec of legacy behaviour != behaviour of legacy implementations
- Proposed solution: offerToReceive* actually creates a transceiver
  - Does not match legacy implementations but createOffer without SLD is rare (hopefully)
- PR: https://github.com/w3c/webrtc-pc/pull/1430

Issue 1361 / PR 1430: Clarify offerToReceiveAudio and offerToReceiveVideo in renegotiation (fluffy)

- Not clear if setting RTCOfferOptions changes what happens in renegotiation or whether this can only be used in an initial offer.
- PR 1430 clarifies this:

```
3480  +                Whenever this is given a non-false value, and the
3481  +                <code><a>RTCPeerConnection</a></code> has no non-stopped
3482  +                "sendrecv" or "recvonly" audio transceivers, createOffer()
3483  +                MUST as its first step invoke the equivalent of
3484  +                <code>addTransceiver("audio")</code> on the
3485  +                <code><a>RTCPeerConnection</a></code> object, except that this
3486  +                MUST NOT <a>Update the negotiation-needed flag</a>,  and,
3487  +                provided this does not fail, proceed with createOffer()'s
3488  +                regular steps.
```

Objects (Taylor)

- Issue 1174: ssrc in RTCRtpEncodingParameters is inconsistent with ORTC
- Issue 1178: When IceTransport and DtlsTransport objects are created/changed
- Issue 1306: Obtaining a list of all DTLS and ICE transports
- Issue 1365: No getStats() for DTLS and ICE transports
- Issue 1406: When ICE restart results in connection to a new endpoint
- Issue 1413: Should even/odd id validation be enforced in RTCDataChannel when negotiated is “true”?
- Issue 1415: When is the dtmf attribute in RTCRtpSender set?
- Issue 1423: What if a datachannel “OPEN” message uses unknown priority?
- Issue 1424: What happens if SRD has two tracks with the same ID?

Issue 1174: ssrc in RTCRtpEncodingParameters is inconsistent with ORTC (Taylor)

- In the ORTC API, the “ssrc” field supports two modes of operation:
  - Set to a specific value, in which case the application must listen for an “ssrcconflict” event and handle SSRC conflicts itself.
  - Undefined, in which case the user agent picks SSRCs itself and handles SSRC conflicts automatically.
- WebRTC always uses the latter behavior. So, should its “ssrc” fields (including rtx.ssrc and fec.ssrc) always be unset? When are they needed by the application?
- A consequence of WebRTC’s choice: if an SSRC conflict occurs between “getParameters” and “setParameters”, setting the old SSRC results in an InvalidModificationError?

Issue 1178: When IceTransport and DtlsTransport objects are created/changed (Taylor)

- When precisely are the RTCIceTransport and RTCDtlsTransport objects created and hooked up to RTCRtpSenders/RTCRtpReceivers?
  - When a local description is applied? (my suggestion)
  - When any offer is applied?
- On a similar note, we should specify that if a remote answer finishes BUNDLE negotiation, the obsolete transports are stopped and the “transport” attribute of senders/receivers starts pointing to the bundle transport.

Issue 1306: Obtaining a list of all DTLS and ICE transports (soareschen)

- In `RTCRtpSender` and `RTCRtpReceiver`, the `transport` and `rtcpTransport` attributes are nullable. In `RTCPeerConnection`, the `sctp` attribute is nullable.
- To obtain all the `RTCDtlsTransport` and `RTCIceTransport` objects, the application needs to wait for the `transport` and `rtcpTransport` attributes to become non-null.
  - Example: listen for the `connectionstatechange` event, then iterate through the `RTCRtpSender`, `RTCRtpReceiver` and `RTCSctpTransport` objects to collect the `RTCDtlsTransport` and `RTCIceTransport` objects.
    - This won't collect all potential `RTCDtlsTransport` and `RTCIceTransport` objects.
- Should/can the API provide a way to accomplish this?
  - `pc.getIceTransports()` and `pc.getDtlsTransports()` methods can already be created based on `pc.getSenders()` and `pc.getReceivers()`.
    - Problem: This only enumerates `RTCDtlsTransport` and `RTCIceTransport` objects once they are created (see Issue 1178).

Issue 1365: No getStats() for DTLS and ICE transports (fippo)

- We have getStats methods both for the RTCRtpSender and the RTCRtpReceiver, but not for the RTCIceTransport and RTCDtlsTransport objects.
- Should the API provide this?
  - Use case: figure out transport used
  - Implies changes to selectorArg

Issue 1406: When ICE restart results in connection to a new endpoint (fippo)

- As the result of a re-offer, an Answer can be received from a new endpoint with a new DTLS certificate.
- When the remote fingerprint changes, is a new RTCDtlsTransport object created, or does the existing one change properties?
- When (and how) does the Javascript layer learn about the changes?
  - After setRemoteDescription()?
  - Does the RTCDtlsTransport.onstatechange EventHandler fire?

Issue 1413: Should even/odd id validation be enforced in RTCDataChannel when negotiated is “true”? (soareschen)

- RTCDataChannelInit has a negotiated parameter:
  - false (default) - in-band negotiation
  - true - out-of-band negotiation
- In-band negotiation sends DATA_CHANNEL_OPEN message
- rtcweb-data-protocol requires odd/even identifier based on DTLS role in DATA_CHANNEL_OPEN message
- Three solution options:
  - Only allow id parameter for out-of-band negotiation
  - Follow original specs and let data channel raise error event when invalid id is provided
  - Change rtcweb-data-protocol to loosen odd/even id requirement

Issue 1423: What if a datachannel “OPEN” message uses unknown priority?

- The “RTCPriorityType” enum attached to a data channel determines both DSCP markings and an integer priority value used for SCTP scheduling.
- It’s RECOMMENDED that the enum maps to SCTP priority values of 128, 256, 512, 1024.
- But an implementation MAY choose another value.
- If it does (say it decides “low = 400, not 512”), and puts this value in the “OPEN” message, what does “RTCDataChannel.priority” return for the created data channel?
- And what DSCP value does it use?

Issue 1415: When is the dtmf attribute in RTCRtpSender set? (soareschen)

- Can the dtmf attribute initially be null and then become non-null at a later time?
- Is dtmf set when setting remote description with m= section having telephone-event lines?
- When calling methods such as addTransceiver('audio'), is the returned sender.dtmf set?
- What happens when another remote description is applied that removes the "telephone-event" format?
  - Does dtmf become null again?
  - What happens if the application holds onto the RTCDTMFSender object and tries to call insertDtmf in this state?

Issue 1424: What happens if SRD has two tracks with the same ID?

- It’s possible to end up with SDP with two identical “a=msid”s using a weird combination of “addTrack”/”replaceTrack”.
- When this happens, do you end up with two remote MediaStreamTracks with the same ID? This could break a lot of people’s assumptions about the uniqueness of IDs.

Miscellaneous (Bernard)

- Issue 1259: What keygenAlgorithm values are supported?
- Issue 1215: Rollback: a feature “at risk”?
- Issue 1283: Centering, Scaling, Cropping
- Issue 763/1323: Handling of encoding/decoding errors

Issue 1259: What keygenAlgorithm values are supported? (fluffy)

- Is there a way to discover what AlgorithmIdentifier values the browser supports?
- Other than the mandatory-to-implement algorithms (or trial and error), the answer appears to be “no”.
  - Mandatory: `{ name: "RSASSA-PKCS1-v1_5", modulusLength: 2048, publicExponent: new Uint8Array([1, 0, 1]), hash: "SHA-256" }`, and `{ name: "ECDSA", namedCurve: "P-256" }`
- Is there interest in providing a solution to this?

Issue 1215: Rollback: Feature “at risk”? (Bernard)

- Should rollback be marked as a “feature at risk”?
  - What is the current implementation state?
  - What are the implementation plans?

Issue 1283: Centering, Scaling, Cropping (EKR)

- WebRTC-PC Section 5.2:

When sending media, the sender may need to rescale or resample the media to meet various requirements including the envelope negotiated by SDP. When resizing video, the source video is first centered relative to the desired video then scaled down the minimum amount such that the video fully covers the desired size, then finally cropped to the destination size. The video remains centered while scaling and cropping. For example, if the source video was 1280 by 720, and the max size that could be sent was 640 by 480, the video would be scaled down by 1.5 and 160 columns of pixels on both the right and left sides of the source video would be cropped off. This algorithm is designed to minimize occurrence of images with with letter box or or pillow boxing. The media must not be upscaled to create fake data that did not occur in the input source.

- JSEP Section 3.6:

If the original resolution exceeds the size limits in the attribute, the sender SHOULD apply downscaling to the output of the MediaStreamTrack in order to satisfy the limits. Downscaling MUST NOT change the track aspect ratio.

Issue 763/1323: Handling of encoding/decoding errors (youennf)

- Section 4.3.2 says:

"If a system has limited resources (e.g. a finite number of decoders), createOffer needs to return an offer that reflects the current state of the system, so that setLocalDescription will succeed when it attempts to acquire those resources. The session descriptions must remain usable by setLocalDescription without causing an error until at least the end of the fulfillment callback of the returned promise. "

- This suggests that errors can occur later - how are they handled? Examples:
  - Resources that were available when the setLocalDescription promise was fulfilled are no longer available when setRemoteDescription() is called.
  - `applyConstraints()` is called to change `width/height/frameRate`
    - Promise is fulfilled, but subsequently encoder resources are not available.
  - `setParameters(parameters)` modifies `scaleResolutionDownBy`
    - Promise is fulfilled (`parameters` are valid), but encoder resources are not available.
- Do we need an `onerror` EventHandler in `RTCRtpSender` and `RTCRtpReceiver` objects?

For extra credit

Name that bird!

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs