W3C WebRTC/MediaCapture WG Meeting

June 7, 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.
  - 107 new issues were filed!
- 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 7_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**
  - **Tracks (Taylor)**
    - **Issue 1161: Do we need a “trackremoved” event?**
    - **Issue 1181: End removed tracks remotely again**
    - **Issue 1207: Setting a remote description may cause discontinuity**
  - **Data Channel (Taylor)**
    - **Issue 1148: How do applications know a DataChannel’s buffer capacity?**
    - **Issue 1287: When data cannot be sent**
  - **DTLS, Certificates and Algorithms (Bernard)**
    - **Issue 1092/PR 1115: DTLS Failures**
    - **Issue 1159/PR 1160: Do we need getAlgorithm()?**
    - **Issue 1250: RTCDtlsTransport: why no getCertificates()?**
    - **Issue 1259: What keygenAlgorithm values are supported?**

For Discussion Today (cont'd)

- **WebRTC-PC Issues**
  - **Objects (Taylor)**
    - **Issue 1174: ssrc in RTCRtpEncodingParameters is inconsistent with ORTC**
    - **Issue 1178: When IceTransport and DtlsTransport objects are created/changed**
  - **Miscellaneous (Bernard)**
    - **Issue 1215: Rollback: a feature "at risk"?**
    - **Issue 1283: Centering, Scaling, Cropping**

Tracks (Taylor)

- Issue 1161: Do we need a “trackremoved” event?
- Issue 1181: End removed tracks remotely again
- Issue 1207: Setting a remote description may cause discontinuity

Issue 1161/Issue 1181: What happens remotely when a track is “removed”? (Taylor)

- Before the introduction of transceivers, calling removeTrack resulted in the remote track being ended. This works in Firefox today, or in Chrome with adapter.js.
- Now, calling removeTrack will only result in the transceiver’s direction being changed; **not clear what event(s) will fire**.
  - Section 5.1 says “ended” event will fire (is this leftover text?)
  - Section 5.3 says “muted” event will fire on receiver.track.
- Should this be changed, and if so how?
  - Suggestion in issue 1161 is to simply add a “trackremoved” event.
  - Suggestion in issue 1181 is to end the remote track when this occurs, and create a new one when applicable. Meaning an RTCRtpReceiver **can have multiple associated tracks over its lifetime**.

Issue 1207: Setting a remote description may cause discontinuity (Taylor)

- Currently, setting a remote description discards any codec changes the application might have done:
  - “The effect of reordering or removing codecs lasts until the codecs are renegotiated. After the codecs are renegotiated, they are set to the value negotiated, and setParameters needs to be called again to re-apply codec preferences.”
- So, **there is some time in between “setRemoteDescription” and “setParameters” where a frame encoded with an unintended codec could leak out**. This could feasibly cause negative side effects, since a new encoder needs to be prepared, a new I-frame created, etc.
- Can we change the API such that the application’s selected codec doesn’t need to change due to setRemoteDescription? For example, allow codec selection via a codecPayloadType RTCRtpEncodingParameter, like in ORTC?

Data Channel (Taylor)

- Issue 1148: How do applications know a DataChannel's buffer capacity?
- Issue 1287: When data cannot be sent

Issue 1287: When data cannot be sent (Taylor)

- A data channel has a buffer; when “send” is called while this buffer is full, **the data channel is closed with prejudice**. This was done to match WebSockets.
- The reasoning for the WebSocket decision (as far as I’ve gathered) was: “Throwing an exception if the buffer is full might be ignored by the application, causing a loss of data integrity. Closing the connection doesn’t have this problem, and is something applications likely are prepared to handle anyway.”
- However, this reasoning doesn’t completely apply to WebRTC DataChannels. A DataChannel doesn’t typically get closed due to errors (such as an SCTP-level timeout); before that happens an ICE restart will typically have occurred.
- **My proposal**: Make “send” throw an exception when the buffer is full, like a non-blocking socket.

Issue 1148: How do applications know a DataChannel's buffer capacity? (Taylor)

- If the answer to the previous discussion was "keep things the way they are": how does an application reliably prevent itself from filling the DataChannel's buffer?
  - Look at each browser's code or experiment to see how big its buffer is (what people do today).
  - Compute pc.sctp.maxMessageSize - channel.bufferedAmount?
  - Expose a "bufferSize" attribute.
  - Introduce a guaranteed minimum buffer size to the spec. Applications will know as long as "bufferedAmount" doesn't go above this, they're safe.

DTLS, Certificates and Algorithms (Bernard)

- Issue 1092/PR 1115:DTLS Failures
- Issue 1250: RTCDtlsTransport: why no getCertificates()?
- Issue 1159/PR 1160: Do we need getAlgorithm()?
- Issue 1259: What keygenAlgorithm values are supported?

Issue 1092/PR 1115:DTLS Failures (Bernard)

- Currently we have RTCPeerConnection.onfingerprintfailure
  - Covers fingerprint verification failures, but not other DTLS errors.
- PR 1115:
  - Utilize RTCDtlsTransport.onerror for all DTLS errors.
  - Add “dtls-failure” to RTCErrorDetailType.
    - receivedAlert set to the value of the DTLS alert received
    - sentAlert set to the value of the DTLS alert sent
  - Do we need to add “fingerprint-failure” to RTCErrorDetailType?
    - RFC 8122 Section 6.2 mandates both clients and servers terminate with “bad_certificate” on fingerprint mismatch.

Issue 1250: RTCDtlsTransport: why no getCertificates()? (Fippo)

- In WebRTC-PC, the RTCDtlsTransport does not provide a way to retrieve the DTLS certificates the transport was created with, whereas in ORTC, there is:
  - ```
    partial interface RTCDtlsTransport : RTCStatsProvider {
        sequence<RTCCertificate> getCertificates();
    };
    ```
- PR 1130 clarified that when RTCConfiguration.certificates is not specified in the RTCPeerConnection constructor, subsequently getConfiguration().certificates is undefined but one or more RTCCertificate instances (“default certificates”) are generated.
  - getConfiguration().certificates cannot be used to retrieve default certificates.
- Do we need to be able to retrieve default certificates?
    ```

Issue 1159/PR 1160: Do we need getAlgorithm()? (Bernard)

- **Issue 496** asked how to tell if a cert was RSA or ECDSA.
  - Use cases: inspection of default certificates, debugging/unit tests, convenience
  - getAlgorithm() subsequently added in PR 499
- Subsequent realization:
  - Without a way to retrieve default certificates, getAlgorithm() can only be called on application-created certificates.
  - Application can store the keygenAlgorithm used to create the certificate.
- Proposal in **PR 1160**: remove getAlgorithm(). No known implementations.

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?

Objects (Taylor)

- **Issue 1174: ssrc in RTCRtpEncodingParameters is inconsistent with ORTC**
- **Issue 1178: When IceTransport and DtlsTransport objects are created/changed**

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?
- Similarly, 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.

Miscellaneous (Bernard)

- Issue 1215: Rollback: a feature “at risk”?
- Issue 1283: Centering, Scaling, Cropping

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.

For extra credit

Name that bird!

Thank you

Special thanks to:

Harald!!!

W3C/MIT for WebEx

WG Participants, Editors & Chairs