W3C WebRTC/MediaCapture WG Meeting

March 1, 2017
8 AM PDT

Chairs: Harald Alvestrand
Stefan Hakansson
Bernard Aboba

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 outstanding issues within both the mediacapture -main and webrtc-pc specifications
- Editor’s Draft updates to follow meeting

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/March_1_2017
- Link to latest drafts:
  - https://rawgit.com/w3c/mediacapture-main/master/getusermedia.html
  - https://rawgit.com/w3c/webrtc-pc/master/webrtc.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**
  - **Pull Requests**
    - Issue 116/PR 1030: Sender and Receiver getStats Selectors (Jan-Ivar)
    - Issue 714/PR 1033: “Hybrid” OAuth Solution (Taylor)
    - Issue 795/PR 1038: RTCDataChannel id assignment (Taylor)
    - Issue 849/PR 1026: Specify an AllowUnverifiedMedia RTCConfiguration property (Fluffy)
    - Issue 305/PR 1023: Describe what happens when media changes (Fluffy)
    - Issue 1024/PR 1025: Codecs[] can be reordered or removed but not modified (Taylor)
  - **Issues**
    - Issue 1040: Codecs supporting multiple clock rates/packetization-mode (Bernard)
    - Issue 1022: sdpFmtpLine isn’t very convenient (Taylor)
    - Issue 845: “Throw a FooError” steps not consistent (Taylor)
    - Issue 526: NetworkError is not defined and might not be needed (Bernard)
    - Issue 1021: Parameter for packetization interval (Bernard)
    - Issue 763: Handling of Simulcast Errors (Bernard)

WebRTC PC Pull Requests

- Issue 116/PR 1030: Sender and Receiver getStats Selectors (Jan-Ivar)
- Issue 714/PR 1033: “Hybrid” OAuth Solution (Taylor)
- Issue 795/PR 1038: RTCDataChannel id assignment (Taylor)
- Issue 849/PR 1026: Specify an AllowUnverifiedMedia RTCConfiguration property (Fluffy)
- Issue 305/PR 1023: Describe what happens when media changes (Fluffy)
- Issue 1024/PR 1025: Codecs[] can be reordered or removed but not modified (Taylor)

Issue 116/PR 1030: Sender/Receiver getStats Selectors (jib)

```
+   getStats (optional (RTCRtpSender or RTCRtpReceiver or MediaStreamTrack)? selector = null);
```

Accepts track for backwards compatibility: Means associated sender/receiver. Throws `InvalidAccessError` if there's none or more than one possible, or wrong pc.

8.5 The stats selection algorithm is as follows:

1. Let result be an empty RTCStatsReport.
2. If selector is null, gather stats for the whole connection, add them to result, return result & abort these steps
3. If selector is an RTCRtpSender, gather stats for and add the following objects to result:
   - All RTCInboundRTPStreamStats objects corresponding to selector.
   - All stats objects referenced directly or indirectly by the RTCInboundRTPStreamStats objects added.
4. If selector is an RTCRtpReceiver, gather stats for and add the following objects to result:
   - All RTCOutboundRTPStreamStats objects corresponding to selector.
   - All stats objects referenced directly or indirectly by the RTCOutboundRTPStreamStats added.
5. Return result.

A stats object is said to "correspond to" a selector if its "ssrc" stats attribute matches an ssrc in the selector.

Issue 116/PR 1030: Sender/Receiver getStats Selectors (cont'd)

- Selecting on a *receiver* would produce the follow stats objects: (map)
  - RTCInboundRTPStreamStats
    - RTCOutboundRTPStreamStats (`.associateStatsId`)
    - RTCMediaStreamTrackStats (`.mediaTrackId`)
    - RTCCodecStats (`.codecId`)
    - RTCTransportStats (`.transportId`)
      - RTCTransportStats (`.rtcpTransportStatsId`)
      - RTCIceCandidatePairStats (`.selectedCandidatePairId`)
        - RTCIceCandidateStats (`.localCandidateId`)
        - RTCIceCandidateStats (`.remoteCandidateId`)
      - RTCCertificateStats (`.localCertificateId`)
      - RTCCertificateStats (`.remoteCertificateId`)

- Selecting on a *sender*, would produce almost the same (switch Inbound<->Outbound).

- Unselectable: RTCPeerConnectionStats, RTCDataChannelStats, RTCMediaStreamStats, unused RTCIceCandidateStats.

Issue 714/PR 1033: “Hybrid” OAuth Solution (Taylor)

“Hybrid” solution uses “username” for the OAuth “kid”, and uses “credential” for “mac_key” and “access_token”:

```
dictionary OAuthCredential {
    DOMString mac_key;      // base64-encoded
    DOMString access_token; // base64-encoded
};

dictionary RTCIceServer {
    required (DOMString or sequence<DOMString>) urls;
    DOMString                          username;
    (DOMString or OAuthCredential)  credential;
    RTCIceCredentialType               credentialType = "password";
};
```

Issue 714/PR 1033: “Hybrid” OAuth Solution (Taylor) (cont.)

- Issue: the working group decided that when a 160-bit key is needed for HMAC-SHA-1, a 256-bit key is simply truncated. But RFC7635 Appendix B states the client “calculates a 160-bit key for HMAC-SHA-1 using SHA1 and taking the 256-bit key as input”. How do we resolve this conflict?
- Even if we decide to go with the “take SHA1 hash” approach, isn’t it an issue that this is only mentioned in Appendix B, and not defined normatively? stun-bis says: “Other credential mechanisms MUST define the key that is used for the HMAC.”
- Is there any reason we can’t just mandate SHA-256 support?

Issue 714/PR 1033: “Hybrid” OAuth Solution (Taylor) (cont.)

- Related issue: The decision to only support 256-bit keys means that WebRTC won’t support hash agility as described in stun-bis, if a new hash algorithm requires a larger key. Is the working group aware of this consequence of the earlier decision?

Issue 795/PR 1038: RTCDataChannel id assignment (Taylor)

- Problem: When a data channel is created without an ID, one is chosen based on DTLS role. But the DTLS role may not be known yet.
- Solution:
  - `id` attribute returns `null` until ID is assigned.
  - Applying an answer that negotiates the DTLS role will assign IDs to any data channels with unassigned IDs.
  - Afterwards, additional data channels will have IDs assigned immediately.
  - If all even/odd IDs are taken, ResourceInUse exception thrown.

Issue 849/PR 1026: Specify an AllowUnverifiedMedia RTCConfiguration property (Fluffy)

- Add a flag that, if set, allows up to 5 seconds of media to received by applications before the SDP fingerprint is received
- Attackers that can compromise the signaling, may be able to launch a MITM attack in this time. However, attackers that can do this can often also comprise the JavaScript and do whatever they want with media
- Default for flag is off, but if enabled, allows early media that arrives before the signalling. Not need by all apps but important for things like 1-800-go-fedex (see https://tools.ietf.org/html/rfc7478#section-2.4.2)

Issue 305/PR 1023: Describe what happens when media changes (Fluffy)

- Summary is “Center the video, then scale to cover, then crop”
- This is what we agreed to in Lisbon.
- Ready to merge after editorial fixes

Issue 1024/PR 1025: Codecs[] can be reordered or removed but not modified (Taylor)

Added the following restriction to setParameters():

When using the `setParameters` method, the `codecs` sequence from the corresponding call to `getParameters` can be reordered and entries can be removed, but entries cannot be added, and the `RTCRtpCodecParameters` dictionary members cannot be modified.

```
Promise<void>      setParameters(optional RTCRtpParameters parameters);

dictionary RTCRtpCodecParameters {
    unsigned short payloadType;
    DOMString      mimeType;
    unsigned long  clockRate;
    unsigned short channels = 1;
    DOMString      sdpFmtpLine;
};
```

Issue 1024/PR 1025: Codecs[] can be reordered or removed but not modified (cont'd)

Added the following restriction to setCodecPreferences():

The `codecs` sequence passed into `setCodecPreferences` can only contain codecs that are returned by `RTCRtpSender.getCapabilities(kind)` or `RTCRtpReceiver.getCapabilities(kind)`, where `kind` is the kind of the `RTCRtpTransceiver` on which the method is called. Additionally, the `RTCRtpCodecParameters` dictionary members cannot be modified. If `codecs` does not fulfill these requirements, the User Agent must throw an InvalidAccessError.

Section 5.4 says:

"When generating a session description using either `createOffer` or `createAnswer`, the user agent must use the indicated codecs, in the order specified in the codecs argument, for the media section corresponding to this `RTCRtpTransceiver`. Note that calls to `createAnswer` will use only the common subset of these codecs and the codecs that appear in the offer. This method allows applications to disable the negotiation of specific codecs. It also allows an application to cause a remote peer to prefer the codec that appears first in the list for sending."

WebRTC PC Issues

- Issue 1040: Codecs supporting multiple clock rates/packetization-mode (Bernard)
- Issue 1022: sdpFmtpLine isn't very convenient (Taylor)
- Issue 845: "Throw a FooError" steps not consistent (Taylor)
- Issue 526: NetworkError is not defined and might not be needed (Bernard)
- Issue 1021: Parameter for packetization interval (Bernard)
- Issue 763: Handling of Simulcast Errors (Bernard)

Issue 1040: Codecs supporting multiple clock rates/packetization-mode (Bernard)

Examples:

- **CN (or SILK) with clock rates of 8000 and 16000**
- **H.264/AVC implementation supporting packetization-mode of 0 or 1**

Question: How do we differentiate these codecs in the following methods:

```
static RTCRtpCapabilities getCapabilities(DOMString kind);

void setCodecPreferences(sequence<RTCRtpCodecCapability> codecs);

dictionary RTCRtpCodecCapability {
    DOMString mimeType;
};
```

Issue 1040: Codecs supporting multiple clock rates/packetization-mode (cont'd)

1. Do we need to add attributes? Strawman:

```
partial dictionary RTCRtpCodecCapability {
    DOMString      mimeType;
    unsigned long  clockRate;
    DOMString      sdpFmtpLine;
};
```

2. If we add attributes, can setCodecPreferences() set any of them? Example:
An application that only requires Opus mono.
Can it set stereo=0 via setCodecPreferences() prior to calling createOffer()?

Issue 1022: sdpFmtpLine isn't very convenient (Taylor)

- How are values of parameters.codecs[].sdpFmtpLine determined? My assumption was:
  - For a sender, sdpFmtpLine value set by the local description
  - For a receiver, sdpFmtpLine value set by the remote description
- Can we change this to a dictionary to allow for easier inspection? Are all parameters used by WebRTC codecs going to be in the form of key/value pairs?
  - For DTMF (RFC 4733), can use events="0-15,66,70" as a key/value pair in this SDP:

```
m=audio 12346 RTP/AVP 100
a=rtpmap:100 telephone-event/8000
a=fmtp:100 0-15,66,70
```

Issue 845: “Throw a FooError” steps not consistent (Taylor)

Proposal:

- For throwing an exception, say “throw a FooError”.
- For rejecting a promise, say “reject p with a newly created FooError and abort these steps.”

This is the wording suggested by WebIDL. The links further clarify what the wording means. We can define the terms “throw” and “created” in the Terminology section, and link to them throughout the document.

Issue 526: NetworkError is not defined and might not be needed (Bernard)

- Section 6.2 (RTCDataChannel) refers to NetworkError:
  - “If the transport was closed with an error, fire a NetworkError event at channel.”
  - “When the user agent determines that an `RTCDataChannel`'s underlying data transport cannot be created, the user agent must queue a task to run the following steps: … 3. Fire a NetworkError event at channel.”
- Section 13 (Event Summary) for RTCDataChannel:
  - close event (Event interface) fired when “The `RTCDataChannel` object's underlying data transport has been closed.”
  - error event fired when “Any error occured from the data channel.”
- Section 12 defines RTCError
  - Attributes: errorDetail, sdpLineNumber, httpRequestStatusCode, message, name
  - Currently not referenced in Sections 6.2 or 13.

Issue 526: NetworkError is not defined and might not be needed (cont'd)

- Stefan: WebSockets uses a close event with reason rather than an error event.
- From https://html.spec.whatwg.org/multipage/comms.html#handler-websocket-onclose :

When *the WebSocket connection is closed*, possibly *cleanly*, the user agent must queue a task to run the following substeps:

1. Change the readyState attribute's value to CLOSED (3).
2. If the user agent was required to *fail the WebSocket connection*, or if the *the WebSocket connection was closed* after being flagged as full, fire an event named error at the WebSocket object. [WSP]
3. Fire an event named close at the WebSocket object, using CloseEvent, with the wasClean attribute initialized to true if the connection closed *cleanly* and false otherwise, the code attribute initialized to *the WebSocket connection close code*, and the reason attribute initialized to the result of applying UTF-8 decode without BOM to *the WebSocket connection close reason*. [WSP]

Issue 526: NetworkError is not defined and might not be needed (cont'd)

User agents must not convey any failure information to scripts in a way that would allow a script to distinguish the following situations:

- *A server whose host name could not be resolved.*
- *A server to which packets could not successfully be routed.*
- *A server that refused the connection on the specified port.*
- *A server that failed to correctly perform a TLS handshake (e.g., the server certificate can't be verified).*
- *A server that did not complete the opening handshake (e.g. because it was not a WebSocket server).*
- *A WebSocket server that sent a correct opening handshake, but that specified options that caused the client to drop the connection (e.g. the server specified a subprotocol that the client did not offer).*
- *A WebSocket server that abruptly closed the connection after successfully completing the opening handshake.*

Issue 526: NetworkError is not defined (cont'd)

- SCTP Operation Error (ERROR) Chunk provides one or more Cause Code/Cause-Specific-Information blocks. From IANA registration page:

Error Cause Codes

Registration Procedure(s)
Specification Required
Reference
[RFC4960]
Available Formats
CSV

Value	Cause Code	Reference
1	Invalid Stream Identifier	[RFC4960]
2	Missing Mandatory Parameter	[RFC4960]
3	Stale Cookie Error	[RFC4960]
4	Out of Resource	[RFC4960]
5	Unresolvable Address	[RFC4960]
6	Unrecognized Chunk Type	[RFC4960]
7	Invalid Mandatory Parameter	[RFC4960]
8	Unrecognized Parameters	[RFC4960]
9	No User Data	[RFC4960]
10	Cookie Received While Shutting Down	[RFC4960]
11	Restart of an Association with New Addresses	[RFC4960]
12	User Initiated Abort	[RFC4460]
13	Protocol Violation	[RFC4460]
14-159	Unassigned	
160	Request to Delete Last Remaining IP Address	[RFC5061]
161	Operation Refused Due to Resource Shortage	[RFC5061]
162	Request to Delete Source IP Address	[RFC5061]
163	Association Aborted due to illegal ASCONF-ACK	[RFC5061]
164	Request refused - no authorization	[RFC5061]
165-260	Unassigned	
261	Unsupported HMAC Identifier	[RFC4895]
262-65535	Unassigned	

Issue 526: NetworkError is not defined and might not be needed (cont'd)

- Does this make sense?

When the RTCDataChannel is closed, possibly *cleanly*, the user agent must queue a task to run the following substeps:

1. Change the readyState attribute's value to CLOSED
2. If the user agent was required to fail the data channel, or if the data channel was closed with buffered data, fire an event named RTCError at the RTCDataChannel object.
3. Fire an event named close at the RTCDataChannel object, using CloseEvent, with the wasClean attribute initialized to true if the connection closed *cleanly* and false otherwise, the code attribute initialized to the transport close code (e.g. the Cause Code provided in the SCTP ERROR chunk), and the reason attribute initialized to the transport result (e.g. the Cause-Specific Information provided in the SCTP ERROR chunk).

Issue 1021: Parameter for packetization interval (Bernard)

- Filed by Varun:
  - “We have a few parameters for video and simulcast, would like to have one for packetization interval. Currently, I cannot find a way to change it without messing with the SDP.”
- First question: Do we want to add maxptime and/or ptime to the object model?
  - maxptime: yes/no?
  - ptime: yes/no?
- If answer to any of the above is “yes”, then how?

Issue 1021: Parameter for packetization interval (cont'd)

- [JSEP] Section 5.2.1 (Initial Offers):
  - `If this m= section is for media with configurable durations of media per packet, e.g., audio, an "a=maxptime" line, indicating the maximum amount of media, specified in milliseconds, that can be encapsulated in each packet, as specified in [RFC4566], Section 6. This value is set to the smallest of the maximum duration values across all the codecs included in the m= section.`
- [JSEP] Section 5.7.2 (Media Section Parsing):
  - `If present, a single "a=ptime" line MUST be parsed as described in [RFC4566], Section 6, and its value stored.`
  - `If present, a single "a=maxptime" line MUST be parsed as described in [RFC4566], Section 6, and its value stored.`
- Taylor: "since "a=ptime" and "a=maxptime" aren't codec-specific attributes, these will be the same for all sent codecs. That's just a limitation of SDP. It may be nice to document this somewhere."

Issue 1021: Parameter for packetization interval (cont'd)

- Potential approaches
  - As a capability/setting of the Sender/Receiver?
  - As a codec property? Example (from ORTC):

```
partial dictionary RTCRtpCodecCapability {
  unsigned long   maxptime; //The maximum packetization time supported by the RTCRtpReceiver.
  unsigned long   ptime; //The preferred duration of media represented by a packet in milliseconds for the RTCRtpSender or RTCRtpReceiver.
};

partial dictionary RTCRtpCodecParameters {
  unsigned long maxptime; // The maximum packetization time set on the RTCRtpSender. Not specified if unset. If ptime is also set, maxptime is ignored.
  unsigned long ptime; // The duration of media represented by a packet in milliseconds for the RTCRtpSender. If unset, the RTCRtpSender may select any value up to maxptime.
};
```

Issue 763: Handling of Simulcast Errors (Bernard)

- How does an application discover how many simulcast streams can be sent?
  - If “too many” simulcast streams are requested, pc.addTransceiver(init) or transceiver.sender.setParameters() can throw an InvalidParameters exception.
  - In existing implementations, simulcast is supported for some codecs, but not others.
- Is trial and error “good enough”?
- If not, how do we discover what the simulcast capabilities are?
  - For a given codec, maximum number of simulcast streams that can be sent.
  - Aggregate simulcast streams that can be sent?
- Does it make sense to add a maxSimulcastStreams attribute in RTCRtpCodecCapability?

```
partial dictionary RTCRtpCodecCapability {
    Unsigned long maxSimulcastStreams = 0;
};
```

Thank you

<u>Special thanks to:</u>

W3C/MIT for WebEx

WG Participants, Editors & Chairs