W3C WebRTC WG Meeting

May 27, 2016 8:00AM-9:30AM PDT

Chairs: Harald Alvestrand

Stefan Hakansson

Erik Lagerway (absent)

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:

- ~~Hangouts Meeting~~
  - ~~Participatory Hangout Link~~
  - Move to WebEx cisco.webex.com/meet/fluffy 204 753 464
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is being recorded.

For Discussion Today

- **Pull Requests**
  - Issue 597/PR 662: Calling RTCRtpReceiver.track.stop()(Bernard Aboba)
  - Issue 644/PR 675: Attribute to turn on/off CN/DTX (Bernard Aboba)
  - PR 646: Table of RTCRtpEncodingParameters (Bernard Aboba)
  - Issue 650/PR 648: mimeType clarification (Bernard Aboba)
  - Issue 651/PR 666: addTransceiver/addTrack: need to be async? (Taylor)
- **Issues**
  - Issue 571: Mechanisms for populating the contents of RTCRtpSender/Receiver are missing (AdamBe)
  - Issue 583: Is it OK to call addTransceiver() with a track already added by addTrack()? (AdamBe)
  - Issue 585: Unclear if RTCRtpTransceiver.stop() acts right away or requires negotiation (AdamBe)
  - Issue 568: Should we specify how addStream()/"addstream event" should behave? (adambe)
  - Issue 548/PR 647: RTX/RED/FEC handling (Bernard Aboba)

Issue 597/PR 662: Calling `receiver.track.stop()`

- What happens when `receiver.track.stop()` is called? Proposal:
  - `track.stop()` is final, so `receiver.track` cannot be rendered after that (clones are not affected).
  - Receiver Reports continue to be sent.
  - If it is desired to stop the transceiver, call `transceiver.stop()`.
- Stefan comment:
  - Is `receiver.track` always a single track? What happens if you do `receiver.track.clone()`? Will the first one be a "master" track?
  - It would have made me feel better if this was analogous to mediacapture-main: all tracks from a source (in this case an RTCRtpReceiver) are created equal and the source is stopped when all associated tracks are.

Issue 644/PR 675: Turn on/off sending CN/DTX

- Cullen: In the case where SDP negotiates the use of CN, there are situations where it is desirable to turn off CN/DTX.
  - Example: conference call with participants generating CN producing a high noise level.
- Proposal:

```
partial dictionary RTCRtpEncodingParameters {
    boolean vadActive;
}
```

  - For an RTCRtpSender, indicates whether voice activity detection (if negotiated) will be used (true) or not (false).

PR 646: Table of RTCRtpEncodingParameters

Attribute	Type	Receiver/Sender	Read/Write
ssrc	unsigned long	Sender	Read-only
fec	RTCRtpFecParameters	Receiver/Sender	Read-only
rtx	RTCRtpRtxParameters	Receiver/Sender	Read-only
vadActive	boolean	Sender	Read/Write
active	boolean	Sender	Read/Write
priority	RTCPriorityType	Sender	Read/Write
maxBitrate	unsigned long	Sender	Read/Write
maxFramerate	unsigned long	Sender	Read/Write
scaleResolutionDownBy	double	Sender	Read/Write
rid	DOMString	Sender	Read-only

Issue 650: mimeType clarification

- RTCRtpCodecParameters and RTCRtpCodecCapabilities have a *mimeType* attribute.
  - Current description: “The codec MIME type.”
  - Is this the “Media Type”, the “Subtype” or both?

Media Type	Subtype	Clock Rate (Hz)	Channels (audio)	Reference
application	1d-interleaved-parityfec			[RFC6015]
application	h224	4800		[RFC4573]
application	parityfec			[RFC3009]
application	raptorfec			[RFC6682]
application	rtx			[RFC4588]
application	smpte336m			[RFC6597]
application	ulpfec			[RFC5109]
audio	1d-interleaved-parityfec			[RFC6015]
audio	32kadpcm	8000		[RFC3802][RFC2421]
audio	ac3			[RFC4184]
audio	AMR	8000		[RFC4867][RFC3267]
audio	AMR-WB	16000		[RFC4867][RFC3267]
audio	amr-wb+	72000		[RFC4352]
audio	atrac3	44100		[RFC5584]
audio	ATRAC-ADVANCED-LOSSLESS			[RFC5584]
audio	atrac-x			[RFC5584]
audio	BV16	8000		[RFC4298]
audio	BV32	16000		[RFC4298]
audio	clearmode	8000	1	[RFC4040]
audio	CN			[RFC3389]
audio	DAT12			[RFC3190]

PR 648: mimeType clarification

- Proposal: *mimeType* contains the "subtype" value.
  - Example: receiver.getCapabilities("audio").codecs[0].mimeType has a value of "CN".
- Would getCapabilities(*kind*) ever need to return a mime media Type value different from *kind*? Examples:
  - Could getCapabilities("depth") return "video/<*depth-codec*>"?
  - Could getCapabilities("audio") return "text/t140" or "text/RED"?
- Taylor comments:
  - "If the set of supported MIME media types is different than the set of supported *kind* values, *mimeType* should be "type/subtype".
  - But is there any reason we can't assume that *kind* == MIME media type? With "audio", "video", "text", "application", "message" and "image" it seems like this is the case so far.

Issue 651/PR 666: addTransceiver/addTrack: need to be async?

addTransceiver/addTrack create a sender with a DtlsTransport. Should these methods be async so that the promise is only resolved when a certificate is ready?

Rough consensus is “no” for a few reasons:

- It’s already possible to have a DtlsTransport without a certificate if setRemoteDescription(offer) is called.
- There doesn’t seem to be a real need for knowing when the certificate is ready.
- If the application *does* need to know when a certificate is ready, it can call createOffer/createAnswer, and doesn’t even need to use the result.

Issue 651/PR 666: Should ‘transport’ be nullable?

Somewhat related to the previous question. Are DTLS (and ICE) transports created when addTransceiver/addTrack is called, or when setLocalDescription/setRemoteDescription is called?

Advantages of creating them later:

- The IceTransport is never in a state where it has an unknown IceRole.
- If a remote offer comes in using “bundle-only”, transports wouldn’t have been created just to be immediately destroyed.

Advantage of creating them earlier:

- EventHandlers can be connected as soon as possible.

Proposal (PR 666):

1. Make *transport* nullable in the RTCRtpSender and RCRtpReceiver.
2. Add text: “RTCDtls(/Ice)Transport objects are constructed as a result of calls to setLocalDescription and setRemoteDescription.”

Issue 571: Mechanisms for populating the contents of RTCRtpSender/Receiver are missing

```
interface RTCRtpSender {
    readonly attribute MediaStreamTrack? track;
    readonly attribute RTCDtlsTransport  transport;
    readonly attribute RTCDtlsTransport? rtcpTransport;
    static RTCRtpCapabilities getCapabilities(DOMString kind);
    Promise<void>       setParameters(optional RTCRtpParameters parameters);
    RTCRtpParameters    getParameters();
    Promise<void>       replaceTrack(MediaStreamTrack withTrack);
};
```

- All info isn't available at creation time - we need to update!
- Related to Issue 651 (addTransceiver/addTrack: need to be async?)
- Options
  - Schedule tasks and fire new event(s)
  - See how far we get with setLocal/RemoteDescription promise fulfillment and 'track' event

Issue 583: Is it OK to call `addTransceiver()` with a track already added by `addTrack()`? *)

- Not allowed by `addTrack()` (`InvalidAccessError`)
- However, the discussion in the Issue concludes that it should be allowed for `addTransceiver()`
  - Argument: `addTransceiver()` used by advanced users who know what they are doing
  - Also: doing `sender.replaceTrack()` with a track already being the `track` of another Sender would give the same result - and we don't forbid that
- Anyone against this resolution?

Side note: We might be able to define `addTrack()` in terms of `addTransceiver()`

*) Should really read "... with a track that is already the `track` of an existing Sender"

Issue 585: Unclear if RTCRtpTransceiver.stop() acts right away or requires negotiation

“The stop method stops the RTCRtpTransceiver. The sender of this transceiver will no longer send, and the receiver will no longer receive.”

- Current text could be interpreted as acting right away, yet we also have:
  - Issue 674: negotiation-needed flag should be set
- We probably want a [[isStopped]] internal slot to set
- What should happen?

Issue 568: Should we specify how addStream()/" addstream event" should behave? (1/3)

- Legacy API related to addStream()/”addstream event”
  - addStream, removeStream, getLocalStreams, getRemoteStreams, getStreamById
  - ‘addstream’ and ‘removestream’ events
- These are removed from the spec but widely used (or?)
- Most functions fairly easy to polyfill; events are harder
- Simplification: If a track is added to a stream added with addStream() then we do nothing (i.e. no ‘negotiationneeded’ event)!

Issue 568: Should we specify how addStream()/" addstream event" should behave? (2/3)

```
addStream(stream):
     do addTrack with each track in stream
     push stream to [[localStreams]]

removeStream(stream)
     let 'senders' be all senders representing the tracks in stream
     do removeTrack each sender in 'senders'
     remove stream from [[localStreams]]

getLocalStreams()
     return [[localStreams]]

getRemoteStreams()
     return [[remoteStreams]]

getStreamById(id)
     if a stream in [[localStreams]] or [[remoteStreams]] has a matching id, return that stream
```

Issue 568: Should we specify how addStream()/" addstream event" should behave? (3/3)

- ‘remoteStreams’ still exists under the hood
- A remote track is added to a set of streams specified by the sending side

```
'addstream' event:
     // these are additions to the 'dispatch a receiver' steps
     if a new stream needs to be created for the 'track' event then:
          add the new stream to [[remoteStreams]] and create an ‘addstream’ event for it
     before setRemoteDescription() fulfills, dispatch all 'addstream' events created above

'removestream' event:
     TBD
```

Issue 548: RTX/RED/FEC Handling

- Are RTX/RED/FEC treated as codecs within RTCRtpCapabilities and RTCRtpParameters?
- Issue:
  - If RTX/RED/FEC are included in getCapabilities(*kind*).

    codecs[], implicit assumption is that they can be used with any codec in the sequence.
    - Problem: Not all implementations that support “rtx”, “red” and “ulpfec” support retransmission of red/ulpfec.
    - Selective support expressible in SDP (or RTCRtpParameters).
      - Result: getParameters() provides more info than getCapabilities()

Issue 548: RTX/RED/FEC Handling (cont'd)

- Alternative proposal (from Robin Raymond):
  - Include features like RTX/RED/FEC as codec attributes in RTCRtpCodecCapabilities and RTCRtpParameters rather than as codecs.
  - Some codecs features (like RTX) need properties like the RTX PayloadType to use. Some codec features just need to announce "I support this feature"
  - Once validity checks pass (to make sure e.g. no RTX PT uses same value as existing codec or other RTX), it's much

PR 647: RTX in Codec Capabilities/Parameters

- In RTCRtpCodecCapability:
  - Only a single entry in *codecs[]* for retransmission via rtx
  - Assumes that RTX can be used with any codec.
- In RTCRtpCodecParameters:
  - Multiple entries in *codecs[]*, each with *codecs[j].mimeType* for the “rtx” codec (setting aside mimeType issue).
  - Each entry has a distinct *codecs[j].sdpFmptLine* attribute, providing “rtxtime” and “apt” parameters.

Thank you

Special thanks to:

Google - for the Hangout

Cisco for WebEx

WG Participants, Editors & Chairs