W3C WebRTC WG Meeting

November 9, 2016 8 AM PST

Chairs: Harald Alvestrand
Stefan Hakansson

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

Limited editor resources for a period

- During November, December and January the webrtc-pc (and mediacapture-main) editor availability will be lower than normal
- To help out we ask everyone to, when possible, file not only Issues but also proposed solutions (in the form of PRs)

About this Virtual Meeting

Information on the meeting:

- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/November_9_2016
- Link to latest draft:
  - 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

- **Pull Requests**
  - Issue 871/PR 901: What happens when transceiver.stop is called (Bernard)
  - Issue 908/Issue 917/PR 916/PR 918: “Stopped” (Bernard)
  - Issue 714/PR 776: STUN/TURN OAuth token parameter (misi)
  - Issue 822/PR 850: Error Handling (Fluffy)
  - Issue 859/PR 895: Need steps for rollback removing a transceiver (Taylor)
  - Issue 803/PR 913: Rules for negotiation-needed flag need to be updated for transceivers (Taylor)
  - Issue 801/PR 920: Revise ICE Agent/User Agent interactions (Taylor)
- **Issues**
  - Issue 902: Does setLocal/setRemoteDescription modify transceiver.direction? (Taylor)
  - Issue 927: What happens when setDirection is called? (Bernard)
  - Issue 760: Adding ufrag+mid to end-of-candidates (Taylor)
  - Issue 726: Adding ufrag to candidates (Taylor)
  - Issue 849: AllowUnverifiedMedia RTCConfiguration Property (Fluffy)

WebRTC PC Pull Requests

- Issue 878/PR 901: What happens when transceiver.stop is called (Bernard)
- Issue 908/Issue 917/PR 916/PR 918: “Stopped” (Bernard)
- Issue 714/PR 776: STUN/TURN OAuth token parameter (misi)
- Issue 822/PR 850: Error Handling (Fluffy)
- Issue 859/PR 895: Need steps for rollback removing a transceiver (Taylor)
- Issue 803/PR 913: Rules for negotiation-needed flag need to be updated for transceivers (Taylor)
- Issue 801/PR 920: Revise ICE Agent/User Agent interactions (Taylor)

RTCRtpTransceiver methods: stop(), setDirection()

- transceiver.stop():
  - The sender of this transceiver will no longer send, the receiver will no longer receive, and the negotiation-needed flag is set.
  - Irreversible (no way to reset transceiver.stopped to false)
- transceiver.setDirection(*direction*):
  - The `setDirection` method sets the direction of the `RTCRtpTransceiver`. Calls to `setDirection()` do not take effect immediately. Instead, future calls to `createOffer` and `createAnswer` mark the corresponding media description as `sendrecv`, `sendonly`, `recvonly` or `inactive` as defined in [JSEP] … Calling `setDirection()` sets the negotiation-needed flag.

Other methods and attributes: setParameters(), active

- sender.setParameters(*parameters*):
  - does not cause SDP renegotiation and can only be used to change what the media stack is sending or receiving within the envelope negotiated by Offer/Answer.
  - No receiver.setParameters() method
- RTCRtpEncodings.active
  - For an `RTCRtpSender`, indicates that this encoding is actively being sent. Setting it to false causes this encoding to no longer be sent. Setting it to true causes this encoding to be sent.
  - For an `RTCRtpReceiver`, indicates that this encoding is being decoded. *Setting it to false causes this encoding to no longer be decoded. Setting it to true causes this encoding to be decoded.*

PR 926

Issue 871/PR 901: What happens when transceiver.stop is called (Bernard)

Stopping a transceiver is irreversible, and will cause future calls to createOffer() to generate a zero port in the media description for the corresponding transceiver, as defined in [JSEP]. When this method is invoked, the user agent MUST run the following steps:

- Let *transceiver* be the RTCRtpTransceiver to be stopped.
- If *transceiver*.stopped is true, abort these steps.
- Let *connection* be the RTCPeerConnection on which the *transceiver* is to be stopped.
- If *connection*'s [isclosed] slot is true, throw an InvalidStateError exception and abort these steps.
- Stop sending media with *transceiver*.sender
- Stop receiving media with *transceiver*.receiver
- Set *transceiver*.receiver.track.readyState to ended
- Set *transceiver*.stopped to true
- Mark *connection* as needing negotiation

Issue 908/Issue 917: “Stopped” (Bernard)

- In the specification, there is the concept of “stopped” for RtpSenders/Receivers.

  Examples:
  - Section 4.3.2: close()
    - All `RTCRtpSender` s in senders are now considered stopped.
    - All `RTCRtpReceiver` s in receivers are now considered stopped.
  - Section 5.1: removeTrack()
    - If sender is stopped, then abort these steps.
    - Stop sender.
  - Section 5.2: An `RTCRtpSender` can be stopped, which indicates that it will no longer send any media.
  - Section 5.2: replaceTrack()
    - If sender is stopped, return a promise rejected with an `InvalidStateError`.

Issue 908/Issue 917: “Stopped” (cont’d)

- More examples:
  - Section 5.4: stopped
    - It is true if either `stop` has been called or if setting the local or remote description has caused the `RTCRtpReceiver` to be stopped.
  - Section 7.2: insertDTMF
    - If sender has been stopped, throw an `InvalidStateError` exception.
    - If sender has been stopped, abort these steps.

PR 916/PR 918: “Stopped” (Bernard)

- Changes proposed by PRs 916 and 918:
  - Section 4.3.2: close()
    - All `RTCRtpSender`s in senders are now considered stopped.
    - All `RTCRtpReceiver`s in receivers are now considered stopped.
    - New text: Set transceiver.stopped to true.
  - Section 5.1: removeTrack()
    - If sender is stopped, then abort these steps.
    - Stop sender.
    - New text: If transceiver.direction is “recvonly” or “inactive”, then abort these steps.
    - Set transceiver.direction to “recvonly”

PR 916/PR 918: “Stopped” (cont’d)

- More examples:
  - Section 5.2: An `RTCRtpSender` can be stopped, which indicates that it will no longer send any media.
    - Proposal: delete
  - Section 5.2: replaceTrack()
    - If sender is stopped, return a promise rejected with an `InvalidStateError`.
    - New text: If transceiver.stopped is true...
  - Section 5.4: stopped
    - It is true if either `stop` has been called or if setting the local or remote description has caused the `RTCRtpReceiver` to be stopped.
    - New Text: has caused the `RTCRtpTransceiver` to be stopped.

PR 916/PR 918: “Stopped” (cont’d)

- Yet more examples:
  - Section 7.2: insertDTMF
    - If sender has been stopped, throw an `InvalidStateError` exception.
    - If sender has been stopped, abort these steps.
    - New text: If transceiver.stopped is true, throw an `InvalidStateError` exception.
    - If transceiver.stopped is true, abort these steps
    - Problem: What if the sender cannot send (e.g. negotiated direction is ‘recvonly’ or ‘inactive’)?
      - Can’t use transceiver.direction from setDirection() to determine this (does not take effect immediately).
      - Do we need another attribute (or should negotiated direction be reflected in encodings[].active)?

Issue 714/PR 776: STUN/TURN OAuth Token Parameter

- PR updated
- Filed by Misi: How is RFC 7635 (STUN Extension for OAuth 2.0) supported within RTCIceServer? Currently, we have:

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

enum RTCIceCredentialType {
    "password",
    "token"
};
```

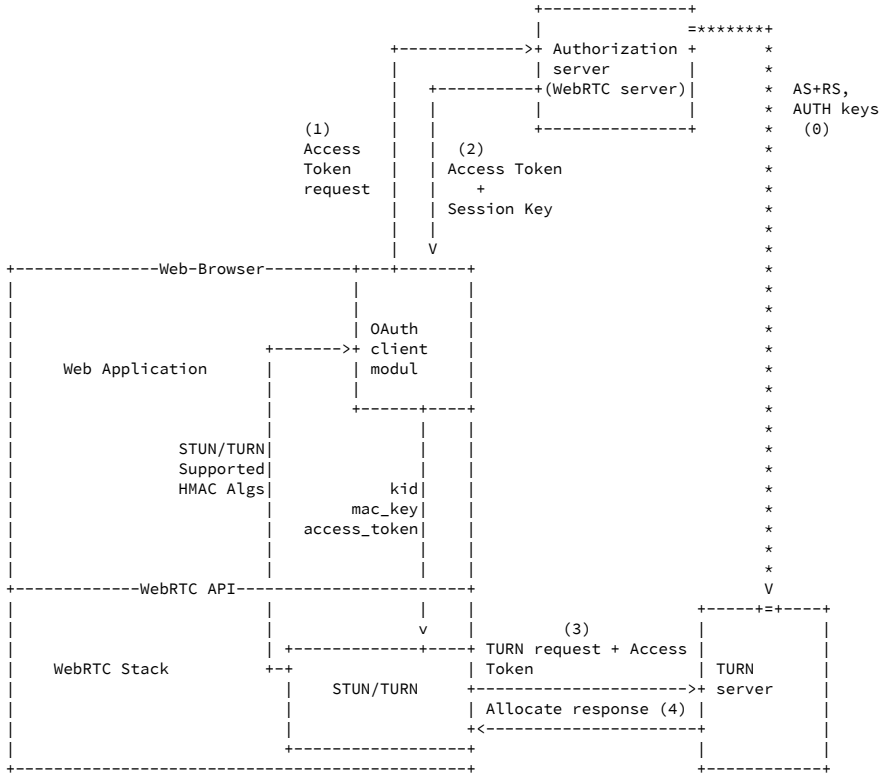
Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- RFC 7635 Appendix B example of a token credential:

```
{
   "access_token":
"U2FsdGVkX18qJK/kkWmRcnfHglrVTJSpS6yU32kmHmOrfGyI3m1gQj1jRPsr0uBb
HctuycAgsfRX7nJW2BdukGyKMXSiNGNnBzigkAofP6+Z3vkJ1Q5pWbfSRroOkWBn",
   "token_type":"pop",
   "expires_in":1800,
   "kid":"22BIjxU93h/IgwEb",
   "key":"v51N62OM65kyMvfTI08O"
   "alg":HMAC-SHA-256-128
}
```

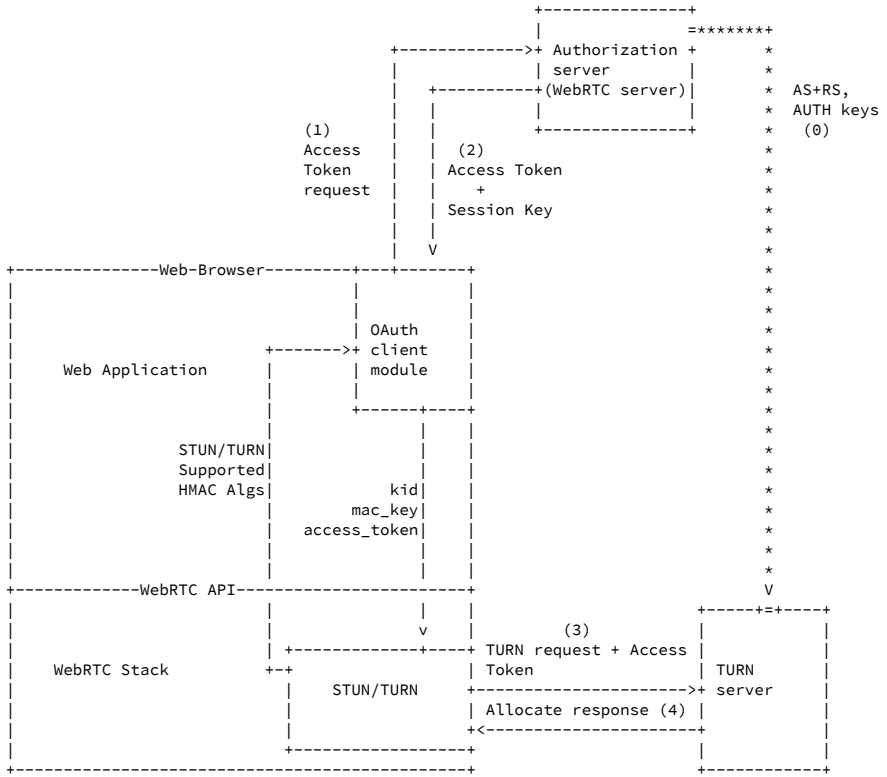
Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- Gap between WebRTC API and RFC7635
  - RFC 7635 defines only external observable behaviours
    - Undefined internal relationship between OAuth client and STUN/TURN client.
  - In WebRTC browser context we need to define (and extend RFC7635)
    - Clarify more OAuth/STUN Client functions duties, interactions
    - define the communication interface between them.
    - More detailed figures, examples
- Is it worth writing about an Informational RFC?
  - E.g.
    "STUN & Third party auth (OAuth) in WebRTC browser context"?



Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

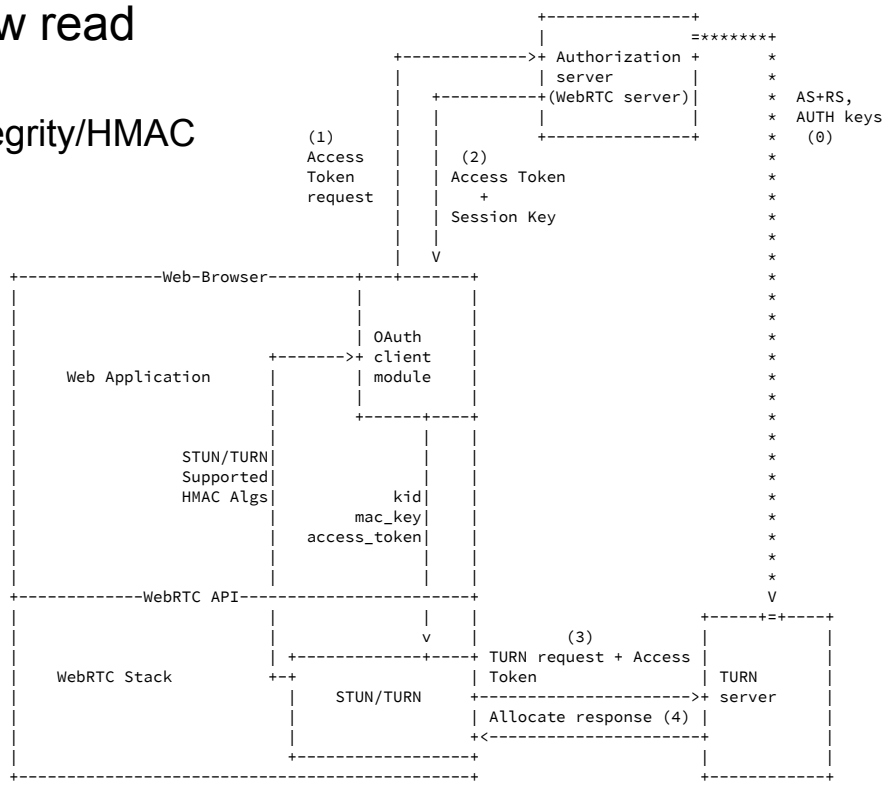
- Explain functionalities
  - OAuth client functionality (probably it is a module in web app)
    - Store OAuth related params
    - Renew access_token and update PC configuration.
  - STUN client functionality Inside ICE Agent
    - Expose during PC creation the supported Message Integrity HMAC Algorithms
    - Get only the needed credentials kid, mac_key, access_token



Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- Proposal part1 (misi):
  Extend RTCPeerConnection with one new read only attribute
- After PC creation extract the supported message integrity/HMAC algorithms from STUN/TURN Client.
  - iceSupportedHMACAlgs
    - Any better name idea?
  - Why?
    Auth Server determines the needed key length
  - Value format ?
    draft-ietf-oauth-pop-key-distribution-01#appendix-A.3
    Examples: HMAC-SHA-1, HMAC-SHA-256-128
- RTCConfiguration extended

```
static readonly attribute DOMString  iceSupportedHMACAlgs;
```



Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- Proposal part2 (misi): Extend RTCIceServer with one new attribute:

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

- Comments
  - Harald: Adding more attributes doesn't seem like the ideal solution.

Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- Concern: Use OAuth & LTC transition & parallel usage
  LTC=Long Term Credential / username and password auth/
  - RTCIceServer is a sequence that can contain multiple times the same url with different credentialType.
  - Order of the same TURN url with different credentialType in "iceservers" could define the preference of credentialType / auth method.

Issue 714/PR 776 STUN/TURN OAuth Token (cont'd)

- Actual text
  - If scheme name is `turn` or `turns`,
    and either of `server.username` or `server.credential` are omitted,
    then throw an `InvalidAccessError` and abort these steps.

- Proposed change:
  - If scheme name is `turn` or `turns` and `server.credentialType` value is `password` ,
    and either of `server.username` or `server.credential` are omitted,
    then throw an `InvalidAccessError` and abort these steps.
  - If scheme name is `turn` or `turns` and `server.credentialType` value is `token` ,
    and either of `server.username` or `server.credential` or `server.access_token` are omitted,
    then throw an `InvalidAccessError` and abort these steps.

Issue 822/PR 850: Error Handling (Fluffy)

- At last TPAC meeting, we agreed we had different errors we needed to recover from and needed more information in the error and agreed to return them in a structure similar to GUM
- Now running into questions about the details of what gets returned on some of the errors.
- Two issues raised covered on next two slides

Issue 822/PR 850: (cont) - SDP Line number (Fluffy)

- JSEP has the browser check the SDP syntax
- If there is an syntax error, the SDP line number is known.
- Question is should we report it in the error or not?
  - A key point in deciding this is the question of it the program would do something with this (other than log it for an operator to read)
- A common approach of programs is to remove the whole m-section by removing all the lines in it and setting the m-line to just a 0 port and trying again.
- This is most likely to happen in the case of a SDP extension that the sender desired to be mandatory to understand

Issue 822/PR 850: (cont) - idpLoginURL (Fluffy)

- Currently draft says
  - Login errors are indicated by rejecting the promise with an object that has a `name` attribute set to "IdpLoginError". If the rejection object also contains a `loginUrl` attribute, this value will be passed to the application in the `idpLoginUrl` attribute.
- The draft is intentionally vague on if one can use multiple identity providers by calling `setIdentityProvider()` multiple times.
- The attribute approach to gettting the `idpLoginUrl` was a simple hack to the lack of being able to pass it in an error that work OK if there is a single identity provider but does not well if there are multiple identity providers.
- Proposal:
  - Add the idpLoginUrl to the error now that the error can carry additional data
  - Have the attribute report the value of last error **or** remove attribute

Issue 859/PR 895: Need steps for rollback removing a transceiver (Taylor)

- Content of PR (already merged):

If description is of type "rollback", then run the following steps:

1. If the `mid` value of an `RTCRtpTransceiver` was set to a non-null value by the `RTCSessionDescription` that is being rolled back, set the `mid` value of that transceiver to null, as described by [JSEP].
2. If an `RTCRtpTransceiver` was created by applying the `RTCSessionDescription` that is being rolled back, and a track has not been attached to it via `addTrack`, remove that transceiver from connection's set of transceivers, as described by [JSEP].

- This behavior is already defined by JSEP.
- Note: this is the *only way* a transceiver is removed; stop() does not remove it.

Issue 803/PR 913: Rules for negotiation-needed flag need to be updated for transceivers (Taylor)

The major changes made by the PR are:

- Negotiation-needed flag sections rewritten to deal with transceivers instead of tracks.
- More thoroughly specify the steps for updating the negotiation-needed flag, and the criteria for determining if negotiation is needed.
- Only update the flag when applying an answer, or modifying the PeerConnection in the "stable" signaling state. Not when creating an answer.
- Removed this: “When attributes on an RTCRtpReceiver are modified, a negotiation is triggered to signal the changes regarding what the application wants to receive to the other side.”

Issue 803/PR 913: Rules for negotiation-needed flag need to be updated for transceivers (cont'd)

Negotiation is needed if any of the following conditions apply:

- Any implementation-specific negotiation is required (this is just carried over from before).
- A data channel was created, and no data section has been negotiated yet.
- For each transceiver:
  - The transceiver isn't yet associated with an m= section.
  - The transceiver is "sendrecv" or "sendonly" and the local description doesn't have an "a=msid" line.

Issue 803/PR 913: Rules for negotiation-needed flag need to be updated for transceivers (cont'd)

- For each transceiver (cont'd)
  - The current local description is an offer, and the transceiver's direction doesn't match the offer or answer.
  - The current local description is an answer, and the direction in the answer doesn't match the transceiver's direction intersected with the direction in the offer.
    - Example: transceiver.direction is "sendrecv", the remote offer is "recvonly", the answer is "sendonly". Even though the answer doesn't match transceiver.direction, it would be fruitless to negotiate again.
  - The transceiver is stopped (and associated with an m= section), but the m= section hasn't yet been rejected.

Issue 801/PR 920: ICE Agent/User Agent interactions (Taylor)

- Descriptions of ICE Agent/User Agent interactions in the RTCPeerConnection section are outdated due to:
  - Addition of RTCIceTransport objects.
  - Additional complexity added to ICE state definitions.
- PR 920 addresses this by:
  - Redefining interactions in terms of an RTCIceTransport.
  - Relying on the definitions of RTCIceTransportState/ RTCIceConnectionState, instead of documenting every possible state transition.

Issue 801/PR 920: ICE Agent/User Agent interactions (cont'd)

Example from PR:

When the ICE Agent indicates that the `RTCIceTransportState` for an `RTCIceTransport` has changed, the User Agent must queue a task that runs the following steps:

1. Let *connection* be the `RTCPeerConnection` object associated with this ICE Agent.
2. If *connection*'s [[isClosed]] slot is `true`, abort these steps.
3. Let *transport* be the `RTCIceTransport` whose state is changing.
4. Let *newState* be the new indicated `RTCIceTransportState`.
5. Set *transport*'s `state` to *newState*.
6. Fire a simple event named `statechange` at *transport*.
7. Update the ICE connection state of *connection*.
8. Update the connection state of *connection*.

WebRTC PC Issues

- **Issue 902**: Does setLocal/setRemoteDescription modify transceiver.direction? (Taylor)
- **Issue 927**: What happens when setDirection is called? (Bernard)
- **Issue 760**: Adding ufrag+mid to end-of-candidates (Taylor)
- **Issue 726**: Adding ufrag to candidates (Taylor)
- **Issue 849**: AllowUnverifiedMedia RTCConfiguration Property (Fluffy)

Issue 902: Does setLocal/setRemoteDescription modify transceiver.direction? (Taylor)

- **Example 1 (Offerer):**

```
transceiver.setDirection("sendrecv");
// ...
pc.setLocalDescription(sendRecvOffer);
// ...
pc.setRemoteDescription(recvOnlyAnswer);
// What is transceiver.direction now? "sendrecv" or "sendonly"?
```

- **Example 2 (Answerer):**

```
transceiver.setDirection("sendrecv");
// ...
pc.setRemoteDescription(sendOnlyOffer);
// ...
pc.setLocalDescription(recvOnlyAnswer);  //Mandated by JSEP Section 5.3.1
// What is transceiver.direction now? "sendrecv" or "recvonly"?
```

Issue 902: Does setLocal/setRemoteDescription modify transceiver.direction? (cont'd)

Pros of the direction changing:

- Can see the negotiated direction without parsing SDP.
  - Though we could add a "negotiatedDirection" attribute to accomplish this.
- Makes the handling of "negotiated-needed" really simple; either the direction matches, or it doesn't.

Cons of the direction changing:

- May force application to call setDirection in between each offer/answer.
- Breaks the common addTrack-type use cases, unless we add even more special rules for addTrack.
- The direction now serves a double purpose: "direction the application wants to negotiate" as well as "direction that was last negotiated".

Issue 902: Does setLocal/setRemoteDescription modify transceiver.direction? (cont'd)

Recommendation: setLocal/setRemoteDescription does not change transceiver.*direction*

- transceiver.*direction* reflects last call to transceiver.setDirection(*direction*)
- Example 1: transceiver.*direction* ***always*** reflects the SDP m-line direction created by createOffer()
  - After setRemoteDescription(recvOnlyAnswer), *direction* is still "sendrecv"
- Example 2: transceiver.*direction* ***not always*** the same as the SDP m-line direction created by createAnswer()
  - After setRemoteDescription(sendOnlyOffer), createAnswer() generates recvOnlyAnswer.
  - But after setLocalDescription(recvOnlyAnswer), transceiver.*direction* is still "sendrecv"
  - To deduce that createAnswer() generated recvOnlyAnswer, need to parse SDP

Question: How do we determine the "negotiated direction" (e.g. can sender send? Can receiver receive?)

Issue 927: What happens when setDirection() is called (Bernard)?

- **Existing text:**
  - The `setDirection` method sets the direction of the `RTCRtpTransceiver`. Calls to `setDirection()` do not take effect immediately. Instead, 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.
- **Strawman proposal:**
  - Let *transceiver* be the RTCRtpTransceiver whose direction is to be set.
  - Let *newDirection* be the argument to setDirection().
  - If *newDirection* is equal to *transceiver*.direction, abort these steps.
  - If *newDirection* has a value other than "sendrecv", "sendonly", "recvonly" or "inactive", throw an InvalidParameters exception and abort these steps.
  - Set *transceiver*.direction to *newDirection*.
  - Let *connection* be the RTCPeerConnection corresponding to transceiver.
  - Mark *connection* as needing negotiation.

Issue 760/Issue 726: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (Taylor)

- **Background:** We should have the ufrag included with ICE candidates so you could disambiguate candidates and end-of-candidates from different ICE generations (restarts). We need to:
  - Add ufrag to ICE candidate event and addIceCandidate
  - Also add ufrag and mid to the “done gathering” indication
- Since end-of-candidates currently works by setting the RTCIceCandidate to “null”, there’s a question of “where should the ufrag/mid go”?

Issue 760/Issue 726: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (cont'd)

- Previously discussed approaches:
  - PR 757:
    - Adds ufrag to RTCIceCandidate.
    - end-of-candidates is an RTCIceCandidate with a null "candidate" string.
  - PR 819:
    - Adds ufrag at a higher level: an optional argument to addIceCandidate, and another member of RTCPeerConnectionIceEventInit.
    - end-of-candidates is a null RTCIceCandidate. It doesn't get a mid.

Issue 760/Issue 726: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (cont'd)

Current best idea - replace "ICE candidate" with "ICE action" (credit to Adam Bergkvist):

```
partial interface RTCPeerConnection {
    attribute EventHandler oniceaction;
    Promise<void> handleIceAction(RTCIceAction action);
}

dictionary RTCIceAction {
    // not all members are defined for all types of actions
    required RTCIceActionType type;
    DOMString sdpMid;
    DOMString sdpMLineIndex;
    DOMString ufrag;
    DOMString candidate;  // not defined if type is "end-of-candidates"
}
```

Issue 760/Issue 726: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (cont'd)

Current best idea - replace "ICE candidate" with "ICE action" (credit to Adam Bergkvist):

```
// Would be easy to extend if new action types are added to ICE.
enum RTCIceActionType { "add-candidate", "end-of-candidates" };

interface RTCPeerConnectionIceEvent : Event {
    RTCIceAction getAction();
    RTCIceCandidate? getCandidate();  // To inspect candidate attributes.
    readonly attribute DOMString? url;
};
```

Issue 760/Issue 726: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (cont'd)

Usage:

```
pc.oniceaction = evt => {
    signalingChannel.send(JSON.stringify({ iceAction: evt.getAction() }));
};

signalingChannel.onmessage = evt => {
    // ...
    if (message.iceAction)
        pc.handleIceAction(message.iceAction).catch(logError);
};
```

Issue 760/Issue 726/Issue 811: Adding ufrag to candidates, and ufrag+mid to end-of-candidates (cont'd)

If we do this, should the new API go on RTCPeerConnection or RTCIceTransport?

Pros of putting it on RTCPeerConnection:

- Just as simple to use as before. No need to wait for transports to be created and hook up events at the right point in time.

Cons of putting it on RTCPeerConnection:

- Requires an extra field (sdpMid) that wouldn't be necessary if it was on RTCIceTransport.
- Doesn't match the object model. Unless you view it as handing an event to the per-PeerConnection "ICE agent", in which case we're fine.

Issue 849: AllowUnverifiedMedia RTCConfiguration Property 1/2 (Fluffy)

- RFC 4572 Section 6.2:
  - [the server endpoint] MUST NOT assume that the data transmitted over the TLS connection is valid until it has received a matching fingerprint in an SDP answer. If the fingerprint, once it arrives, does not match the client's certificate, the server endpoint MUST terminate the media connection with a bad_certificate error, as stated in the previous paragraph.
- The default behavior needs to be not to render this data

(more on next slide)

Issue 849: AllowUnverifiedMedia RTCConfiguration Property (cont'd)

- Alice calls Bob. Bob's browser does ICE while ringing. Once Bob answers, Bob does have alice's fingerprint and can immediately say "hello". Alice will likely receive the RTP before the fingerprint. For applications that display to alice that the speaker is not known and the connection is not secure, the risk of Alice hearing hello is not a big deal. Once Alice's application receives Bob's fingerprint, the application can enable outbound media from Alice and display the call as secure to Bob.
- To support this:
    - transceiver.receiver returned by addTransceiver can immediately be hooked up to an audio or video tag
    - A DtlsTransport can provide a limited buffer for unverified media (so as to prevent loss of packets in a key frame)

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs