WebRTC 1.0 objects

at 2015 f2f

Done: DtlsTransport

```
partial interface RTCRtpSender {
  readonly attribute RTCDtlsTransport transport;  // And rtcpTransport
}

partial interface RTCRtpReceiver {
  readonly attribute RTCDtlsTransport transport;  // And rtcpTransport
}

interface RTCDtlsTransport {
  readonly attribute RTCIceTransport transport;
  readonly attribute RTCDtlsTransportState state;
  sequence<ArrayBuffer> getRemoteCertificates();
  attribute EventHandler onstatechange;
}
```

Done

Done: initial IceTransport

```
interface RTCIceTransport {
  readonly attribute RTCIceConnectionState state;
  RTCIceCandidatePair? getSelectedCandidatePair();
  attribute EventHandler onstatechange;
  attribute EventHandler onselectedcandidatepairchange
}

dictionary RTCIceCandidatePair {
  RTCIceCandidate local;
  RTCIceCandidate remote;
}
```

Done

Done: RtpSender/RtpReceiver parameters

```
partial interface RTCRtpSender {
  RTCRtpParameters getParameters();
  void setParameters(RTCRtpParameters parameters);
}


dictionary RTCRtpParameters {
  sequence<RTCRtpEncodingParameters> encodings;
}

dictionary RTCRtpEncodingParameters {
  boolean active;
  RTCPriorityType priority;  // high, medium, low, very-low
  unsigned long maxBitrate;
}
```

Done

Pending: RtpSender/RtpReceiver capabilities (PR 269)

```
partial interface RTCRtpSender {  // Same as RTCRtpReceiver
  static RTCRtpCapabilities getCapabilities(DOMString kind)
}

dictionary RTCRtpCapabilities {
  sequence<RTCRtpCodecCapability> codecs;
  sequence<RTCRtpHeaderExtensionCapability> headerExtensions;
}

dictionary RTCRtpCodecCapability {
  DOMString mimeType;
}

dictionary RTCRtpCodecCapability {
  DOMString uri;
}
```

Can we merge?

Pending: SctpTransport (PR 270)

```
partial interface RTCPeerConnection {
  readonly attribute RTCSctpTransport? sctp
}

interface RTCSctpTransport {
  readonly attribute RTCDtlsTransport transport;
  readonly attribute unsigned long maxMessageSize;
}
```

Can we merge?

Needs discussion: IceTransport more readonly info: (PR 280)

```
partial interface RTCIceTransport {
  readonly attribute RTCIceRole role;
  readonly attribute RTCIceComponent component;
  readonly attribute RTCIceGatheringState state;
  RTCIceParameters? getLocalParameters();
  RTCIceParameters? getRemoteParameters();
  sequence<RTCIceCandidate> getLocalCandidates();
  sequence<RTCIceCandidate> getRemoteCandidates();
  attribute EventHandler ongatheringstatechange;
}

dictionary RTCIceParameters {
  DOMString usernameFragment;
  DOMString password;
}
```

Discussion:

Worth Having?

Needs discussion: RtpSender more readonly info (PR 273)

```
dictionary RTCRtpParameters {
  // ...
  sequence<RTCRtpHeaderExtensionParameters> headerExtensions;
}

dictionary RTCRtpHeaderExtensionParameters {
  DOMString uri;
  unsigned short id;
  boolean encrypted;
}

dictionary RTCRtpEncodingParameters {
  unsigned long ssrc;
  RTCRtxParameters rtx;    // dictionary { unsigned long ssrc; }
  RTCFecParameters fec;    // dictionary { unsigned long ssrc; }
  RTCRtcpParaemeters rtcp; // dictionary { DOMString cname; boolean reducedSize; }
}
```

Discussion:
Worth Having?

Needs discussion: RtpSender codec selection (PR 258)

```
dictionary RTCRtpEncodingParameters {
  unsigned short payloadType;
}

dictionary RTCRtpParameters {
  // ...
  sequence<RTCRtpCodecParameters> codecs;  // These are all read-only
}

dictionary RTCRtpCodecParameters {
  // These are all read-only.
  DOMString mimeType;
  unsigned long clockRate;
  unsigned short channels;
  DOMString sdpFmtpLine;
}
```

Discussion:

Worth Having?

RtpSender codec selection example

```
// Pick dalaa if it's available
var params = sender.getParameters();
for (codec of params.codecs) {
  if (codec.mimeType == "video/dalaa") {
    params.encodings[0].payloadType = codec.payloadType;
    sender.setParameters(params);
  }
}
```

Needs discussion: RtpSender more parameters (PR 273)

```
dictionary RTCRtpEncodingParameters {
  double resolutionScale;  // 1 == full, 2 == half, 4 == quarter
  double framerateScale;   // 1 == full, 2 == half, 4 == quarter
  double framerateBias;    // 1.0 == framerate, 0.0 == resolution, 0.5 == default
}
```

Discussion:
Worth Having?

Needs Discussion: PC.connectionState (PR 291)

```
partial interface PeerConnection {
  // disconnected, connecting, connected, failed
  readonly attribute PeerConnectionState connectionState;
  attribute EventHandler onconnedctionstatechange;
}

pc.onconnectionstatechange = function() {
  if (pc.connectionstate == "failed") {
    // Uh-oh!
  } else if (pc.connectionstate == "connected") {
    // Great!
  }
}
```

Needs Discussion

More slides: https://docs.google.com/presentation/d/1vHTlEof4dV12iAtZ7SrchrtVK1o239pbrPi9fAutnb4/edit#slide=id.

Needs Discussion: PC.onwarning/onfatalerror (PR 292)

```
partial interface PeerConnection {
  // The PeerConnection can't continue.
  attribute EventHandler onfatalerror;
  // The PeerConnection can continue.
  attribute EventHandler onwarning;
}

pc.onwarning = function(evt) {
  console.log(evt.message);
}

pc.onfatalerror = function(evt) {
  console.log(evt.message);
  goToBrokedUI();
}
```

Needs Discussion

Needs Discussion: PeerConnection warmup (PR 271)

```
partial interface PeerConnection {
  RtpSender createRtpSender(DOMString kind);
}
```

For more details, this deserves its own slide deck:

[PeerConnection ICE/DTLS warmup]

Needs Discussion

PeerConnection warmup example

```
// Offer side
var sender = pc.createRtpSender("audio");  // Adds sendrecv m-line in createOffer
// ... otherwise normal offer/answer/SLD/SRD ...
// Wait for the "I really answered" bit in signalling
sender.replaceTrack(track);
// Hookup pc.getReceivers()[0].track

// Answer side
var sender = pc.createRtpSender("audio");  // Uses existing m-line in createAnswer
// ... otherwise normal offer/answer/SLD/SRD ...
// Wait for the user to really answer
// Send the "I really answered" bit
sender.replaceTrack(track);
// Hookup pc.getReceivers()[0].track
```

PeerConnection warmup questions

- What's the "track ID"/MSID/MID? A random ID
- Do we do anything with track-specific hardware codecs? No
- Does this create an m-line with sendrecv, sendonly, recvonly, or inactive? sendrecv, just like addTrack
- Can we rename replaceTrack to setTrack? replaceTrack doesn't make sense when there isn't one yet.
- Should replaceTrack cause renegotiation? If so, I vote we add another method: setTrackWithoutRenegotiation and use that for ICE/DTLS warmup.
- Do we need to worry about changing from sendonly/inactive to sendrecv/recvonly? I don't think we do.

Needs Discussion: Replace offerToReceiveX (PR 279)

```
partial interface PeerConnection {
  // Note: We need to change createRtpSender to create a
  // sendonly m-line unless an RtpReceiver is also created
  // This changes the warmup example.
  RtpSender createRtpSender(DOMString kind);
  RtpReceiver createRtpReceiver(DOMString kind);
}
```

For more details, this deserves its own slide deck

Remaining issues with RtpSenders/RtpReceviers and SDP

Needs Discussion

Example of createRtpReceiver()

```
var sender = pc.addTrack(audioTrack);  // audio sendrecv
var sender = pc.createRtpSender();  // video sendonly
sender.replaceTrack(videoTrack);
var receiver2 = pc.createRtpReceiver("audio");  // audio recvonly
var receiver3 = pc.createRtpReceiver("audio");  // audio recvonly
// Generates like {offerToRecevieAudio: 3, offerToReceiveVideo: 0}
//   1 audio sendrecv line
//   1 video sendonly line
//   2 audio recvonly lines
pc.createOffer();
```

Did you notice a problem?

While discussing RtpSender/RtpReceiver, did you notice that it's very complicated to reason about when they are created, how they map to SDP, and what state they are in. With our current model of things, I found that things get hairy rather quickly and there are lots of remaining issues not address in the spec. I have documented many of those and created a proposal to fix them here:

Remaining issues with RtpSenders/RtpReceviers and SDP and a proposal to resolve them

Proposal: RTCRtpTransceiver objects

```
partial interface PeerConnection {
  RTCRtpTransceiver addMedia(DOMString kind or MediaStreamTrack, RTCRtpTransceiverInit dict);
  sequence<RTCRtpTransceiver> getMedia();
}

dictionary RTCRtpTransceiverInit {
  bool send = true;
  bool receive = true;
}

interface RTCRtpTransceiver {
  readonly attribute mid;  // Chosen when addMedia is called.
  // These are non-nullable.  You get one, even if it isn't actively sending/receiving.
  readonly attribute RtpSender sender;
  readonly attribute RtpReceiver receiver;
  readonly attribute bool stopped;
  void stop();
}
```

or Proposal: RTCSdpMediaSection objects

```
partial interface PeerConnection {
  RTCSdpMediaSection addMedia(DOMString kind or MediaStreamTrack, RTCSdpMediaSectionInit dict);
  sequence<RTCSdpMediaSection> getSdpMediaSections();
}

dictionary RTCSdpMediaSectionInit {
  bool send = true;
  bool receive = true;
}

interface RTCSdpMediaSection {
  readonly attribute mid;  // Chosen when createRtpPair is called.
  // These are non-nullable.  You get one, even if it isn't actively sending/receiving.
  readonly attribute RtpSender sender;
  readonly attribute RtpReceiver receiver;
  readonly attribute bool close;
  void close();
}
```

Examples of RTCSdpMediaSection objects

```
// Replaces createRtpSender, and createRtpReceiver, and offerToReceiveX
pc.addMedia(track);
pc.addMedia("audio", {send: false, recv: true});
pc.addMedia("audio", {send: false, recv: true});
pc.addMedia("video", {send: true, recv: false));
// Adds 1 sendrecv audio, 2 recvonly audio, and 1 sendonly "warmup" video
pc.createOffer();

// On answer side, replace track.stop()
pc.setRemoteDescription(offer);
for (media of pc.getSdpMediaSections()) {
  if (media.receiver.kind == "video") {
    media.close();
  }
}
pc.addTrack(audioTrack)
// Send back one audio, leave two extra audio inactive, reject the video warmup
```

Needs lots of Discussion