W3C WebRTC WG Meeting

TPAC 2022

Chairs: Bernard Aboba

Harald Alvestrand

Jan-Ivar Bruaroey

W3C WG IPR Policy

- This group abides by the W3C Patent Policy https://www.w3.org/Consortium/Patent-Policy/
- 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

TPAC Health Rules

While physically attending TPAC, follow the Health Rules:

- Authorized masks are required indoors at all time (no exceptions)
  If you need to remove your mask during a meeting, keep it short (but keep enjoying that water/coffee)
- Daily COVID19 self-test is expected

Accommodate participants' needs for physical distancing and other accommodations or precautions due to health concerns

Lunch boxes will be provided for inside or outside. There is limited outdoor seating at the hotel but there are public parks within a 10-minute walk.

TPAC Meeting Schedule

[September 2022 - Web Real-Time Communications Working Group Wiki (w3.org)]

- WebRTC WG: September 12, 2022
  - 9 AM - 12 noon Pacific Time
- WebRTC WG: September 13, 2022
  - 9 AM - 12 noon Pacific Time
- Joint WebRTC/MEDIA/MEIG: September 15, 2022
  - 3:30 PM - 5:30 PM Pacific Time
  - Slides

Welcome!

- Welcome to the TPAC 2022 meetings of the W3C WebRTC WG.
- Future meetings:
  - October 18 2022
  - November 15 2022

About TPAC 2022 Meetings

- TPAC Schedule:
  - https://docs.google.com/spreadsheets/d/1GNwrHKI06ftyCldJIkxxYF1Sq1m-5uf4rM5bkresfD0/
- WEBRTC WG Meeting info:
  - September 2022 - Web Real-Time Communications Working Group Wiki (w3.org)
- Link to slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- Will we be recording the session?
- Volunteers for note taking?

W3C Code of Conduct

- This meeting operates under W3C Code of Ethics and Professional Conduct

- We're all passionate about improving WebRTC and the Web, but let's all keep the conversations cordial and professional

Understanding Document Status

- Hosting within the W3C repo does ***not*** imply adoption by the WG.
    - WG adoption requires a Call for Adoption (CfA) on the mailing list.
- Editor’s drafts do ***not*** represent WG consensus.
    - WG drafts ***do*** imply consensus, once they’re confirmed by a Call for Consensus (CfC) on the mailing list.
    - Possible to merge PRs that may lack consensus, if a note is attached indicating controversy.

Virtual Meeting Tips (Zoom)

- **Both local and remote participants need to be on irc.w3.org channel #webrtc.**
- **Type q+ and q- in IRC to get into and out of the speaker queue.**
- **To try out WebCodecs over RTCDatachannel (not RTP!) join using a Browser.**
- **Please use headphones when speaking to avoid echo.**
- **Please wait for microphone access to be granted before speaking.**
- **Please state your full name before speaking.**

September 12 Agenda

- 09:10 - 09:30 WebRTC WG: State of the Union (Harald, 20 mins)
- 09:30 - 09:45 WebRTC-NV Use Cases (Bernard, 15 mins)
- 09:45 - 10:00 Developer Engagement (Tim Panton, 15 mins)
- 10:00 - 10:45 WebRTC-PC (Dom & Jan-Ivar, 45 mins)
- 10:45 - 11:15 Break (30 minutes)
- 11:15 - 11:45 WebRTC-Stats (Henrik & Varun, 30 minutes)
- 11:45 - 12:00 WebRTC-Extensions (Fippo, 15 mins)

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

September 13 Agenda

- 09:10 - 09:40 Encoded-Transform Challenges (Harald, 30 mins)
- 09:40 - 10:10 Encoded-Transform Issues (Youenn, 30 mins)
- 10:10 - 10:40 Conditional Focus (Elad, 30 mins)
- 10:40 - 11:10 Break (30 minutes)
- 11: 10 - 11:30 Screen-sharing Next Steps (Elad, 20 mins)
- 11:30 - 12:00 Topic TBD (30 mins)

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

WebRTC WG: State of the Union (Harald)

Start Time: 09:10

End Time: 09:30

WEBRTC State of the Union

TPAC 2022, Vancouver

What we are chartered to do

Our charter - started in 2011 and unchanged for the last 2 cycles - states that the mission of the WG is “to define client-side APIs to enable Real-Time Communications in Web browsers”.

Our list of documents is long - some are “functionally stable”, others are still “works in progress” - but only one (WebRTC-PC) has achieved REC status

What we have done

WebRTC 1.0 went to REC

Mediacapture-Main “close to ready” to move to PR

Lots of extensions collected in mediacapture-extensions, webrtc-extensions, new docs - some tidying needed here

- transforms
- capture manipulation
- new media functionality
- Workers interaction

Not much work on redesigning current concepts (“good enough, don’t change”?)

What we are currently working on

- New features for capture: gaze correction, face detection and so on
- New features for capture handling: Crop, select, prefer
- Getting Mediacapture-main and Webrtc-stats out the door
- Challenging the model
    - Integration with WebCodecs
    - Stream transforms of media (unencoded and encoded)

What we are not doing

Actively decided to not pursue

- Mediacapture-depth (contributions have lapsed)

No proposals to address

- 3D, spatial awareness

Being done elsewhere (?)

- Codec management (media)
- encrypted media (EME)
- stored media being streamed (MOQ)

Lots of other groups are working on tools related to media rendering / processing.

Changes since last year

We have some more clarity on what we’re doing, what we’re not doing, and how we move things between the two.

We don’t have many new resources - either for old work or for new work.

We do have a great deal of usage - both inside and outside browsers; WebRTC is THE p2p stack.

Discussion (End Time: 9:30 AM)

-

WebRTC-NV Use Cases (Bernard)

Start Time: 09:30

End Time: 09:45

“Would you tell me, please, which way I ought to go from here?”
“That depends a good deal on where you want to get to,” said the Cat.
“I don’t much care where–” said Alice.
“Then it doesn’t matter which way you go,” said the Cat.

-- Lewis Carroll, “Alice in Wonderland”

History of WebRTC-NV Use Cases

- FPWD: December 11, 2018
  - 27 months before the pandemic.
  - 28 months before FPWD of WebCodecs.
- Technologies that reached the mass market during the pandemic:
  - Podcasting
  - Video conferencing
  - Video streaming services (live, video-on-demand)
  - Game streaming
  - IoT devices (doorbells/security, exercise equipment, robots, smart speakers)
  - Large scale webinars, classes, “town hall” meetings (100K+ viewers)
  - Online events (auctions, conferences, sporting events, concerts)
- After discussion at TPAC 2021, use cases added in November 2021:
  - Section 3.2: Low latency broadcast
  - Section 3.3: Internet of Things
  - Section 3.4: Decentralized messaging
  - Section 3.9: Reduced complexity signaling

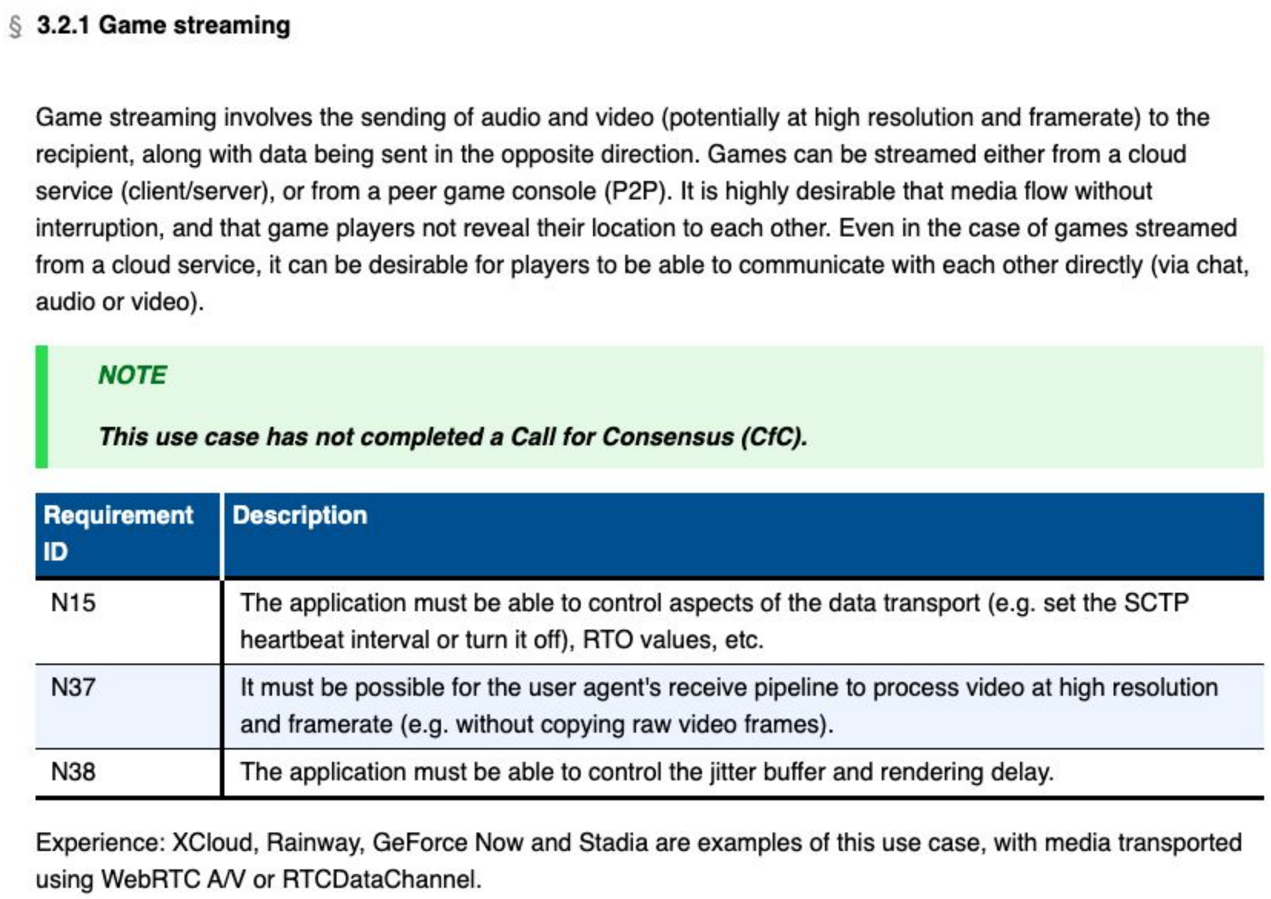
Are We Enabling These Use Cases?

- Simple answer: no!
  - No use cases have ***all*** requirements met by API proposals.
  - Only 4 of 11 use cases have ***any*** API proposals.
  - Some use cases may not have sufficient interest and/or consensus
    - Call for Consensus (CfC) once requirements are filled in.
  - Substantial gaps in data transport (required by 7 of 11 use cases)
    - Access to low-latency transport
    - Need for low-level control
    - Performance (worker support, copy minimization)
- What is the approach to satisfying the use cases?
  - “Extend WebRTC”?
  - WebCodecs over RTCDataChannel?
  - WebCodecs over WebTransport?
  - WebCodecs over RTP transport?

Issue 62/PR 75: Requirements for low latency streaming use cases

- PR 75 under development to add requirements.
- PR splits use case into two sub-sections:
  - Game streaming (Section 3.2.1)
    - Requires ultra-low latency (< 100 ms)
    - Experience: Many (all?) game streaming services now using WebRTC (some transporting media over RTP, others using data channel)
  - Low latency broadcast (Section 3.2.2)
    - Covers webinars, classes, events, “town hall” meetings
    - Goal is both large scale (1M participants) and low latency (< 1 second)
      - CDN/eCDN support for scalability and cost control
    - IETF WISH WG standardizing ingestion via WebRTC
    - Experience: pipe, Peer5, Dolby

Issue 62/PR 75: Requirements for low latency streaming use cases (cont’d)



§ 3.2.1 Game streaming

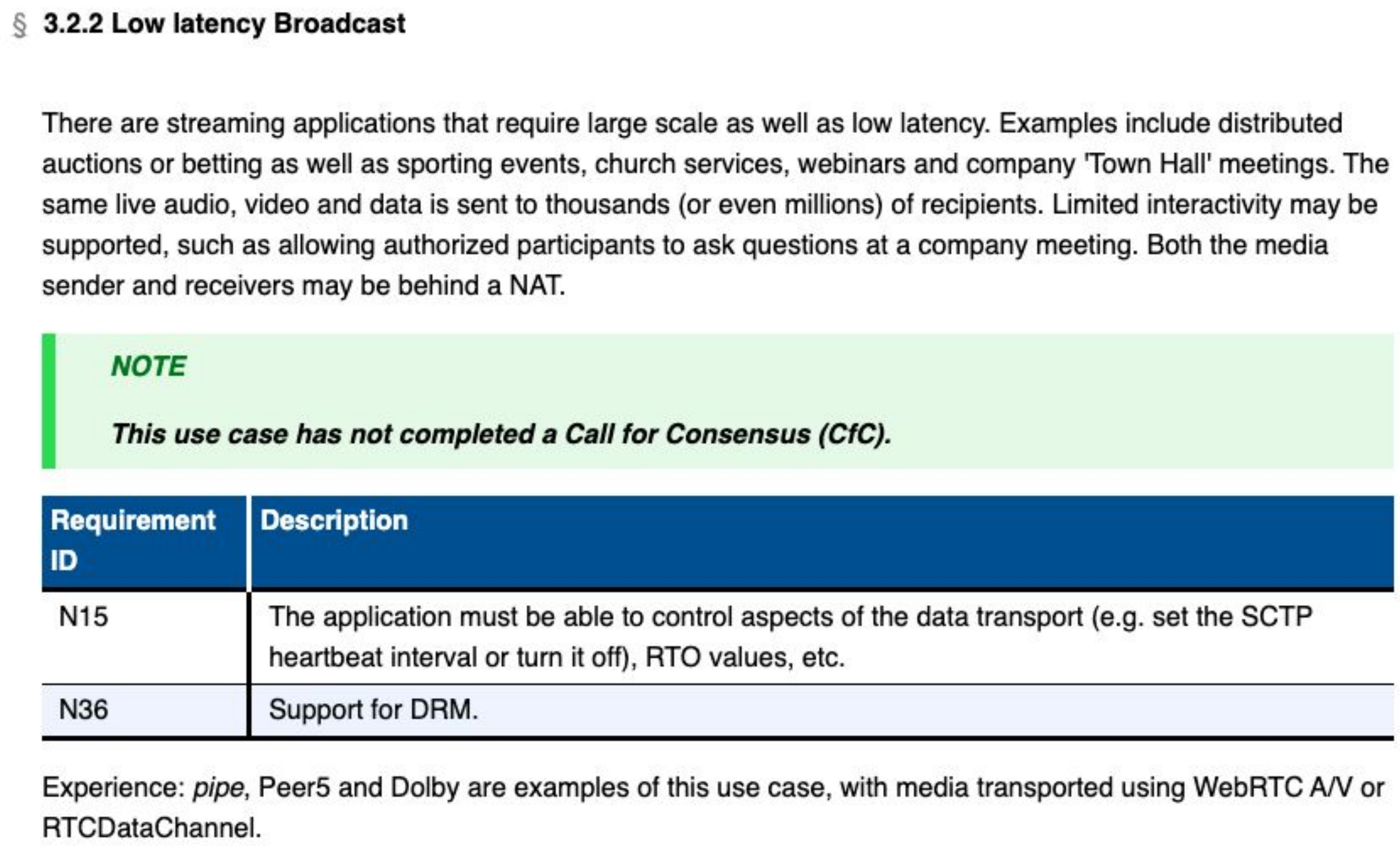
Game streaming involves the sending of audio and video (potentially at high resolution and framerate) to the recipient, along with data being sent in the opposite direction. Games can be streamed either from a cloud service (client/server), or from a peer game console (P2P). It is highly desirable that media flow without interruption, and that game players not reveal their location to each other. Even in the case of games streamed from a cloud service, it can be desirable for players to be able to communicate with each other directly (via chat, audio or video).

> ***NOTE***
>
> ***This use case has not completed a Call for Consensus (CfC).***

Requirement ID	Description
N15	The application must be able to control aspects of the data transport (e.g. set the SCTP heartbeat interval or turn it off), RTO values, etc.
N37	It must be possible for the user agent's receive pipeline to process video at high resolution and framerate (e.g. without copying raw video frames).
N38	The application must be able to control the jitter buffer and rendering delay.

Experience: XCloud, Rainway, GeForce Now and Stadia are examples of this use case, with media transported using WebRTC A/V or RTCDataChannel.

Issue 62/PR 75: Requirements for low latency streaming use cases (cont'd)



§ 3.2.2 Low latency Broadcast

There are streaming applications that require large scale as well as low latency. Examples include distributed auctions or betting as well as sporting events, church services, webinars and company 'Town Hall' meetings. The same live audio, video and data is sent to thousands (or even millions) of recipients. Limited interactivity may be supported, such as allowing authorized participants to ask questions at a company meeting. Both the media sender and receivers may be behind a NAT.

NOTE

This use case has not completed a Call for Consensus (CfC).

Requirement ID	Description
N15	The application must be able to control aspects of the data transport (e.g. set the SCTP heartbeat interval or turn it off), RTO values, etc.
N36	Support for DRM.

Experience: *pipe*, Peer5 and Dolby are examples of this use case, with media transported using WebRTC A/V or RTCDataChannel.

Discussion (End Time: 10:00)

-

Developer Engagement (Tim Panton)

Start Time: 09:45

End Time: 10:00

How to get more resources (i.e. people involved)

Ask a dev-rel expert:

“… growing a community … look at what's valuable to the people who are showing up, what's valuable and unaddressed to the people who aren't showing up, what the blockers are to people who aren't showing up.

… how do I lower barriers?”

Jessica Rose - distributedfutu.re/#episode78

What’s valuable to the people who are showing up

Standards grow the market

Standards avoid some legal ‘complexities’

Standards provide certainty to developers

Shared expertise

As a group we can achieve more than any single effort

Valuable and unaddressed to the people who aren't showing up

I’m not sure we know….

because we haven’t asked?

Who would we like to show up?

What would we like them to contribute?

Blockers for people who aren't showing up:

informally:
legal - several former contributors changed roles and now can’t attend.

irrelevance - slow or zero progress on issues that ‘matter’ * to them…

cumbersome process, hostile atmosphere

Only tackle half the problems (IETF rtcweb is inactive)

Problem summary

WebRTC developer when asked what it would take for them to contribute:
"To not feel like it would be a waste of time ? Ref the w3c standards group) maybe I’m being too harsh but the times I’ve offered my opinion outside of the group…. I don’t feel like I’ve been listened to so (shrug)”

Possible solutions:

Create a ‘WebRTC Network of Users’
Quarterly meetings
Used as input to the W3C process
Chatham house rules
Acts as an ‘invited expert’ to this group

Broaden our view

Take input from other non-browser projects

With related APIs (e.g. Pion)

Use them as sandboxes for experimenting with changes that can’t safely or easily be done in a browser.

Focus remains on browser APIs, just leverage outside info better.

Discussion (End Time: 10:00 AM)

-

WebRTC-PC (Dom, Jan-Ivar)

Start Time: 10:00

End Time: 10:45

For Discussion Today

- WebRTC-PC Revision Process (Dom)
- PR 2763: add relayProtocol to RTCIceCandidate (Fippo)
- Issue 2732: Inconsistent rules for rid in RTCRtpEncodingParameters (jib)
- Issue 2733: addTransceiver does not check for uniqueness of rid (Jan-Ivar)
- Issue 2734: addTransceiver does not check for missing rid properties (jib)
- Issue 2762: Simulcast: Implementations do not fail (Jan-Ivar)
- Issue 2764: What is the intended behavior of rollback of remote simulcast offer? (Jan-Ivar)
- Issue 2736: webrtc-pc does not say to clear RTCRtpCodingParameters.rid when sRD rejects simulcast (Jan-Ivar)
- Issue 2737: Modifications to [[SendEncodings]] from setParameters and sRD can be racy (bonus issue) (Jan-Ivar)

WebRTC revision process

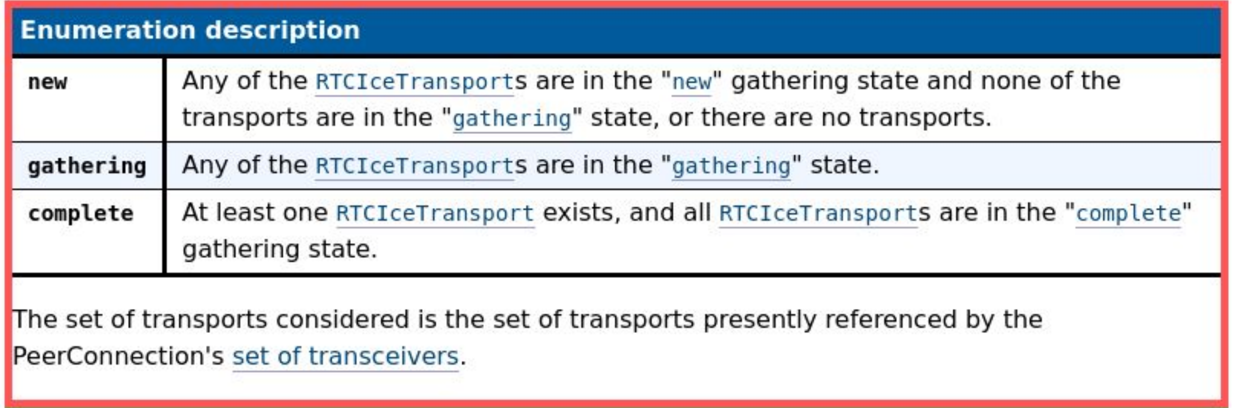
- WebRTC published as a Rec in Jan 2021
- 19 normative-change pull requests merged since, visible in the Editors draft
- We can and should update the Recommendation to reflect these corrections

Updating WebRTC Rec

- Proposed tooling to help align with W3C Process Requirements
- Once merged, we can automatically publish corrections as they land in the Editors Draft
- Some small additional work for editors

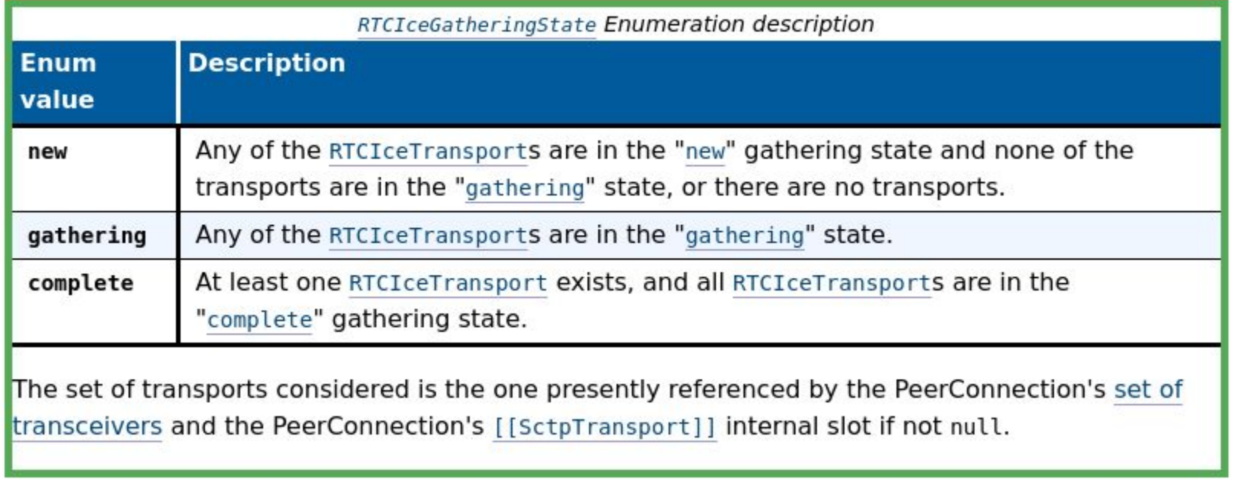
CANDIDATE CORRECTION 3: Update RTCIceGatheringState to clarify the relevant transport it represents (PR #2680)

[✓ Show Current and Future] Show Current Show Future



Enumeration description	
new	Any of the `RTCIceTransport`s are in the "`new`" gathering state and none of the transports are in the "`gathering`" state, or there are no transports.
gathering	Any of the `RTCIceTransport`s are in the "`gathering`" state.
complete	At least one `RTCIceTransport` exists, and all `RTCIceTransport`s are in the "`complete`" gathering state.

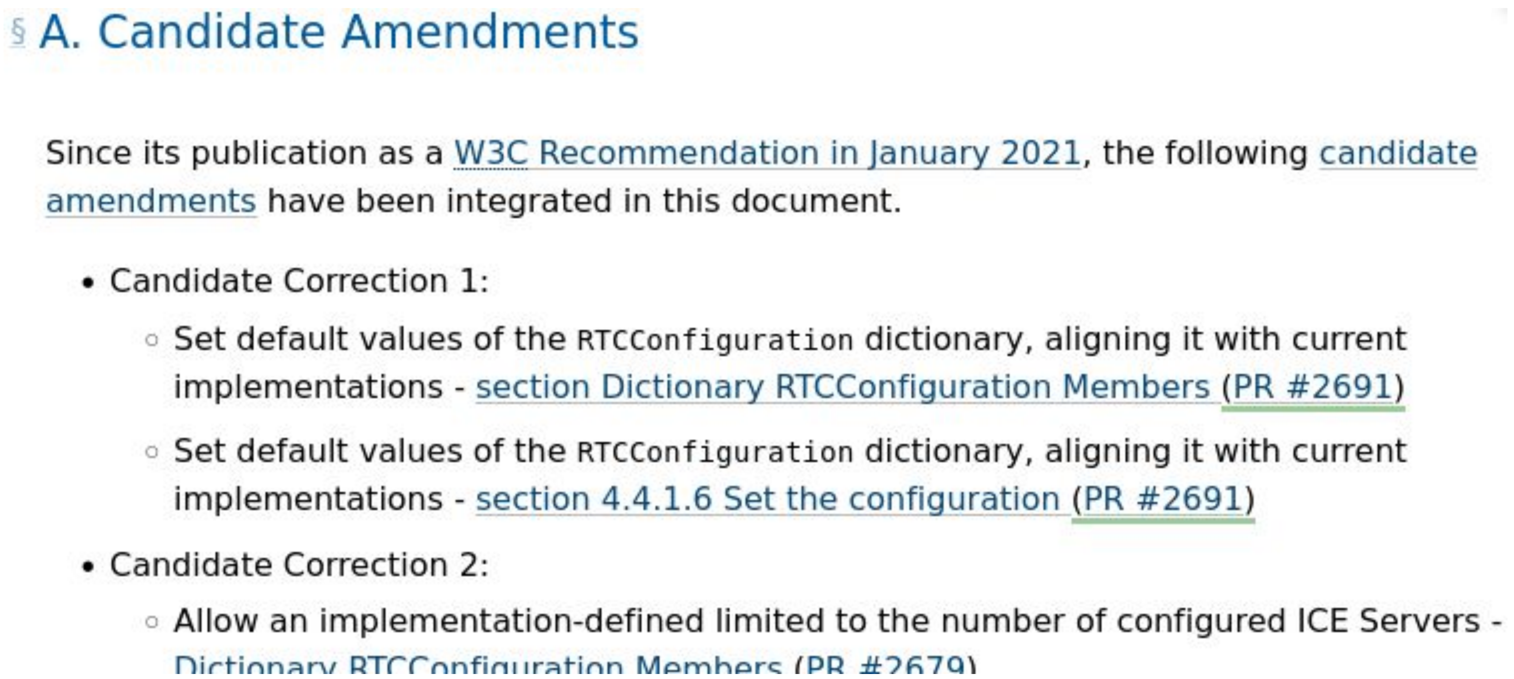
The set of transports considered is the set of transports presently referenced by the PeerConnection's set of transceivers.



`RTCIceGatheringState` Enumeration description

Enum value	Description
new	Any of the `RTCIceTransport`s are in the "`new`" gathering state and none of the transports are in the "`gathering`" state, or there are no transports.
gathering	Any of the `RTCIceTransport`s are in the "`gathering`" state.
complete	At least one `RTCIceTransport` exists, and all `RTCIceTransport`s are in the "`complete`" gathering state.

The set of transports considered is the one presently referenced by the PeerConnection's set of transceivers and the PeerConnection's `[[SctpTransport]]` internal slot if not `null`.



§ A. Candidate Amendments

Since its publication as a W3C Recommendation in January 2021, the following candidate amendments have been integrated in this document.

- Candidate Correction 1:
  - Set default values of the `RTCConfiguration` dictionary, aligning it with current implementations - section Dictionary RTCConfiguration Members (PR #2691)
  - Set default values of the `RTCConfiguration` dictionary, aligning it with current implementations - section 4.4.1.6 Set the configuration (PR #2691)
- Candidate Correction 2:
  - Allow an implementation-defined limited to the number of configured ICE Servers - Dictionary RTCConfiguration Members (PR #2679)

Moving away from webrtc-extensions?

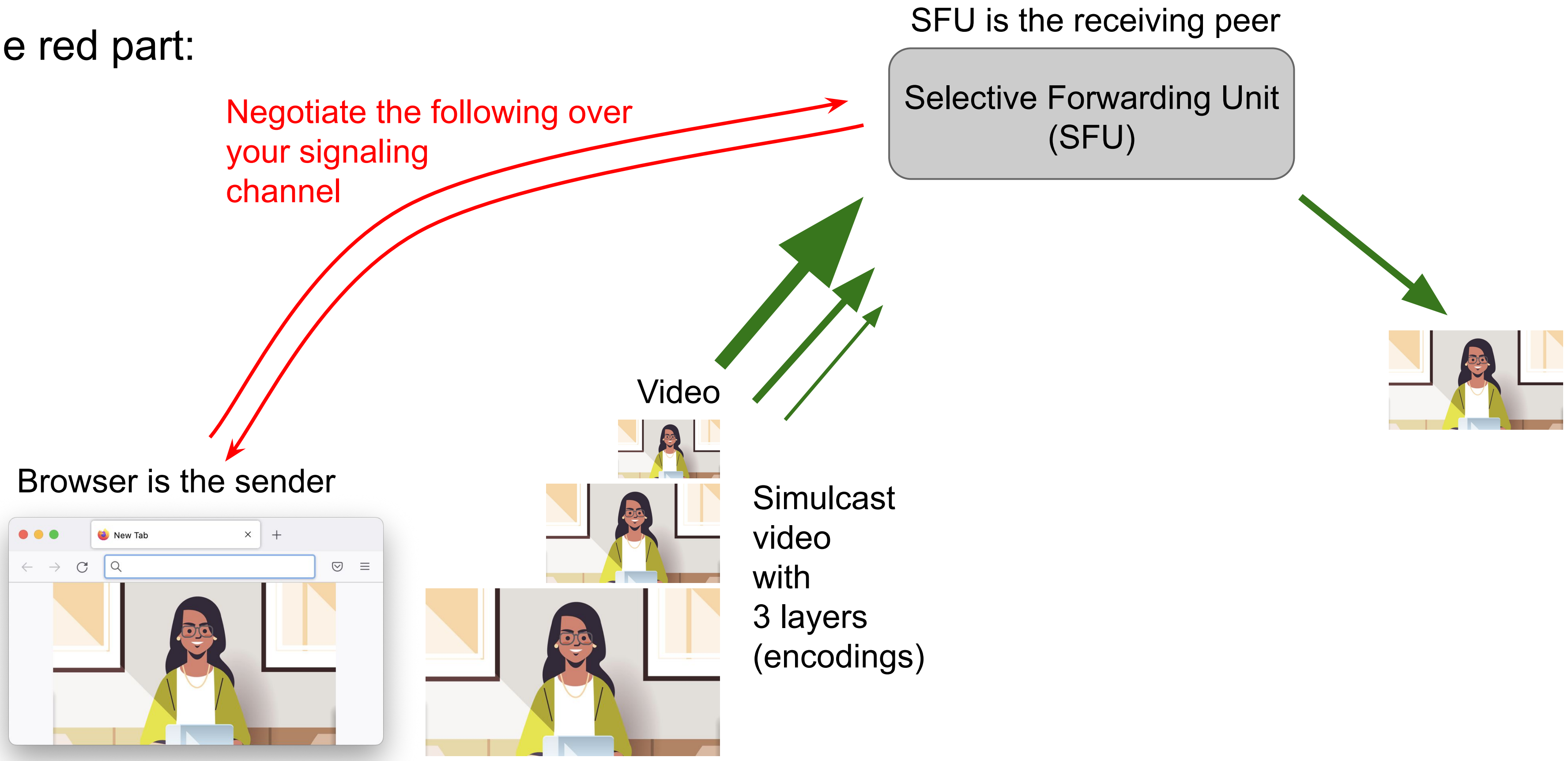
- We could use the same tool to document less mature changes in-line in the editors draft
- This would allow avoiding the split between webrtc-pc and webrtc-extensions
- (the same could apply to mediacapture-main once it reaches Recommendation status)

PR 2763: add relayProtocol to RTCIceCandidate (fippo)

- inconsistency between candidate object and stats
  - candidate is missing relayProtocol
  - stats is missing foundation, component, relatedAddress, relatedPort, usernameFragment
- url is property of RTCPeerConnectionIceEvent
  - has anyone implemented this yet?
  - Chrome implementation gets proxied by candidate
  - expose on candidate, remove on event?

Welcome! It’s all about simulcast negotiation! (Jan-Ivar)

The red part:


SFU is the receiving peer
Selective Forwarding Unit (SFU)
Negotiate the following over your signaling channel
Video
Browser is the sender
New Tab
Simulcast video with 3 layers (encodings)

Issue 2732: Inconsistent rules for rid in RTCRtpEncodingParameters 1/2 (Jan-Ivar)

The following succeeds in Firefox, but Chrome/Edge & Safari throw TypeError: Illegal length of rid (or similar message)

```
const transceiver = pc.addTransceiver("video", {
  sendEncodings: [{rid: "thequickbrownfoxj"}] // > 16 characters
});
```

The spec relies on RFC 8851 which allows 256 octets, but in practice, people use single-character RIDs, because anything bigger bloats the RTP header.

Proposal: limit rids to 16 characters for web compat.

Issue 2732: Inconsistent rules for rid in RTCRtpEncodingParameters 2/2 (Jan-Ivar)

Chrome/Edge and Safari similarly throw on `"-"` and `"_"` (which the spec and RFC 8851 say are valid rids), and throw on `""` (which RFC 8852 says is). Firefox doesn't.

Align spec with Chrome and Safari for web compat by doing either of:

Proposal A: switch to RFC 8852 (`"-"` and `"_"` invalid), but add `""` is invalid

Proposal B: add `""`, `"-"` are invalid (with note that `""` is invalid already)

Issue 2734: addTransceiver does not check for missing rid properties (Jan-Ivar)

This succeeds in Firefox, but Chrome/Edge and Safari throw InvalidAccessError: RIDs must be provided for either all or none of the send encodings. (or similar message)

```
const transceiver = pc.addTransceiver("video", {
  sendEncodings: [{rid: "a"}, {}]
});
```

Proposal: Clarify spec to align with Chrome and Safari on this app mistake.

There's some evidence this was the spec's intent all along, but it's using "verify each rid value" on a dictionary which technically means only on values that exist. The absence of the (faulty) checks against undefined to determine existence seen on adjacent members, suggests the intent here was for the bad grammar checks (RFC8851) to have caught rid absence as `""`.

Issue 2733: addTransceiver does not check for uniqueness of rid (Jan-Ivar)

Proposal: this, which works in all browsers today, must throw `TypeError` (which is what we throw today for bad grammar rids in RFC 8851):

```
const transceiver = pc.addTransceiver("video", {
  sendEncodings: [{rid: "a"}, {rid: "a"}]
});
```

It’s a silly application mistake to have two layers with identical rid names.

Curious asymmetry: RFC 8853 doesn’t seem to prohibit a=simulcast:recv a;a in SRD(offerToReceiveSimulcast), and browsers allow it. How to treat this?

Issue 2762: Simulcast: Implementations do not fail (Jan-Ivar)

Problem: The spec is overly strict about the number of rids in SFU reoffers: *“If [SRD(offer)] contains a request to receive simulcast, and applying description leads to modifying a transceiver transceiver, and transceiver.[[Sender]].[[SendEncodings]] is … not equal to the encodings that would result from processing description, the process of applying description* ***fails****. This specification does not allow remotely initiated RID renegotiation.”*

Here, [[SendEncodings]] reflects previous *answer* not previous offer, which is onerous: Asking SFUs to keep track of previous *answers* is impractical & unnecessary in negotiation

Initiating a renegotiation is harmless when answer is the same (a feature of negotiation):

- *“I want 5 layers”*, “I’ll send you 2”, … *“How about 10?”*, “Just 2” // no “RudeError” 🙂

SFUs re-offer all the time to add/drop participants, and today’s browsers aren’t failing them:

1. remote reoffers can reduce the number of simulcast layers (works in browsers)
2. remote reanswers can reduce the number of simulcast layers (works in browsers)
3. remote reoffers increasing the number of simulcast layers (are no-op in browsers)
4. Once reduced down, the number of simulcast layers doesn’t go back up (in browsers)

Issue 2762: Simulcast: Implementations do not fail (Jan-Ivar)

Proposal: amend the spec to not fail SRD in these cases, to align w/implementations:

1. remote reoffers reducing the number of simulcast layers (may reduce encodings)
2. remote reanswers reducing the number of simulcast layers (reduce encodings)
3. remote reoffers increasing the number of simulcast layers (keep encodings)
   a. The UA must keep `sender.getParameters().encodings` unchanged, and the UA's answer must keep the number of layers the same

Editorially, we can also remove language that SRD must fail in these cases:

1. remote answers increasing the number of offered simulcast layers (invalid JSEP)
2. remote answer altering RID names (invalid in JSEP)

…not because they shouldn't fail but because this is covered by IETF specs.
Browsers should update to follow spec on the last two.

Issue 2764: What is the intended behavior of rollback of remote simulcast offer? (Jan-Ivar)

```
await pc.setRemoteDescription(offerToReceiveSimulcast_1mline_3rids);
const [{sender}] = pc.getTransceivers();
console .log(sender.getParameters().encodings.length); // 3
await pc.setRemoteDescription({type: "rollback"});
console.log(sender.getParameters().encodings.length); // 0 or 3?
```

The spec fails to undo changes to `sender.getParameters().encodings` in this case. In tests, Chrome undoes the encodings changes while Safari doesn't.

Proposal: Fix the spec to undo the encodings changes in keeping with RFC 8829 (section 4.1.10.2.):

*“it may be desirable to "undo" a change made to … setRemoteDescription … we introduce the concept of "rollback", which **discards any proposed changes** to the session, returning the state machine to the "stable" state”*

Issue 2736: RTCRtpCodingParameters.rid isn't cleared when sRD rejects simulcast (Jan-Ivar)

Contrasting examples:

1.
```
pc.addTransceiver("video", {sendEncodings: [{rid: "a"]});
// rid gets cleared. getParameters().encodings is [{scaleResolutionDownBy: 1}]
```

2.
```
pc.addTransceiver("video", {sendEncodings: [{rid: "a"}, {rid: "b"}]});
// [{rid: "a", scaleResolutionDownBy: 2}, {rid: "b", scaleResolutionDownBy: 1}]}
await negotiate(); // in this example, remote answer is 1 layer (rejects simulcast)
// We end up with [{rid: "a", scaleResolutionDownBy: 2}]}, a thumbnail
```

In 1 and 2, the spec says to truncate [[SendEncodings]] to size 1, but in 2 it doesn't say anything about clearing the rid on that last encoding like it does in 1.

Maybe this is harmless? Seems a bit weird to leave rid in there when nothing is using it, but maybe having a stable identifier is worth it?

Also, since scaleResolutionDownBy was auto-filled with small to large powers of 2, truncating to size 1 leaves us with the thumbnail encoding in 2, which also may be a little surprising.

Proposal: Leave as is?

Issue 2737: Modifications to [[SendEncodings]] from setParameters and sRD can be racy (bonus issue) (Jan-Ivar)

What to do when setParameters races against sRD(offerToReceiveSimulcast) depends on what the expected behavior is in the non-racy case:

***Q:** "I call setParameters ahead of SRD(offerToReceiveSimulcast) because"*:

1. I don't know what I'm doing
2. I'm operating on the assumption of unicast and am not prepared to send simulcast
3. Some other reason?

Proposal A: Early setParameters() locks in unicast answers from the browser

Proposal B: SRD(offerToReceiveSimulcast) erases early setParameters()

Proposal C: SRD(offerToReceiveSimulcast) causes racy error from setParameters()

Proposal D: Attempt a merge/extrapolation of intent to other layers (here be dragons)

Discussion (End Time: 10:45 AM)

-

Break (End Time: 11:15 AM)

TPAC Health Rules

While physically attending TPAC, follow the Health Rules:

- Authorized masks are required indoors at all time (no exceptions)
  If you need to remove your mask during a meeting, keep it short (but keep enjoying that water/coffee)
- Daily COVID19 self-test is expected

Accommodate participants' needs for physical distancing and other accommodations or precautions due to health concerns

Lunch boxes will be provided for inside or outside. There is limited outdoor seating at the hotel but there are public parks within a 10-minute walk.

WebRTC-Stats (Henrik & Varun)

Start Time: 11:15

End Time: 11:45

For Discussion Today

Status

- **Towards PR: we've moved unimplemented metrics to webrtc-provisional-spec.**
- **PR blocker: spec/implementations don't align on RTP lifetime.**

Today

- **Issue 667 - When are RTP stream stats created?**
- **Issue 668 - When are RTP streams destroyed?**
- **Issue 643 - Do we agree removing "sender", "receiver" and "transceiver" stats is a good idea?**
- **Issue 666 - powerEfficientEncoder/powerEfficientDecoder**
- **Issue 662 - Don't expose so many RTCCodecStats!**

Issue 667 - When are RTP stream stats created?

Spec says: “When the first RTP packet is sent or received on the SSRC…”

Implementations:

- Chromium: When the SSRC is first *known*.
    - If SSRC is ***not*** signalled, SSRC is known on the first recv packet.
    - Chrome signals SSRC, leading to RTP stats appearing ***before*** the first packet. *(Fun fact: some WPTs depend on this bug.)*
- Firefox: correct implementation for outbound-rtp, but not inbound-rtp?

Update spec or update implementation?

- **Proposal A:** No spec change. Add “if packets > 0” to implementations.
- **Proposal B:** Expose RTP stats if *either* SSRC or MID is known.
    - SSRC no longer `required` (could be missing)!
    - Different lifetimes depending if SSRC is signalled :(

Issue 668 - When are RTP streams destroyed?

Spec says: Never delete RTP stats.

Requires recording SSRC history (cache on SSRC removal). ***Memory leak!***

Chromium only exposes “current RTP streams”.

PR 672: Delete when the sender/receiver is “reconfigured”, meaning…

- The transceiver is `stopped`, or…
- `The ssrc` changes, or…
- The `codecId` or number of simulcast layers are renegotiated.

If the RTP stream is used again, a new RTP stats object is created.

- Packet counters are reset (no SSRC history needed).
- New object has new `id`, so counters never decrease on a given object.

Issue 643 - Do we agree removing "sender", "receiver" and "transceiver" stats is a good idea?

Sender/receiver/transceiver objects were moved to webrtc-provisional-stats due to lack of implementation (part of the webrtc-stats → PR effort).

Q: Was removing them a good idea?

Issue 643 - Do we agree removing "sender", "receiver" and "transceiver" stats is a good idea?

Sender/receiver/transceiver objects were moved to webrtc-provisional-stats due to lack of implementation (part of the webrtc-stats → PR effort).

Q: Was removing them a good idea?

A: Yes.

- Bloat! 50 transceivers = 150 additional stats objects…
- They add no new information.
  - MID? See inbound-rtp or outbound-rtp (which also has RID).
  - What if RTP don't exist because packets == 0? Just use pc.getTransceivers()!

Proposal: Don't revive these objects unless we have a better use case.

Issue 666 - powerEfficientEncoder/powerEfficientDecoder

People want to know if they are are using an efficient encoder/decoder.

Power efficient ≈ HW encoder/decoder.

Not added in the past because MediaCapabilities has powerEfficient.

Problem:

- If you *actually* got HW is a separate question than if there is HW *support*.
  - E.g. HW encoder already in use, SW fallback due to decode error.
- Ergonomics and API compatibility.
  - MediaCapabilities and RTCCodecStats don't map 1:1. FMTP line?
- ***Already exposed*** *as a "hack"*: People now depend on implementation-specific encoderImplementation to deduce isHW… 🙈

Proposal (PR 670): Add powerEfficient[En/de]coder to RTP stats (CL ready).

Issue 662 - Don't expose so many RTCCodecStats!

Without BUNDLE, 872 codec stats objects can be expected in a 50p meeting.

With BUNDLE, we're down to "only" 43 codec stats.

- You *probably* only care about codecs currently in use.

Proposal (PR 669):

- Only expose codecs currently in use, i.e. referenced by an RTP stream.
- ≤ 4 codecs in most cases.

"But I want to know which other codecs were negotiated!"

- See `pc.localDescription` / `pc.remoteDescription`.

Discussion (End Time: 11:45 AM)

WebRTC-Extensions (Fippo)

Start Time: 11:45

End Time: 12:00

For Discussion Today

- Issue 98: Disabling hardware encoding/decoding (Fippo)
- Issue 112: header extension API: do we need enabled in setParameters?(Fippo)
- Issue 111: Integration of congestion control across SCTP and media (Harald)

Issue 98: Disabling hardware encoding/decoding

- Hardware encoding and decoding for WebRTC is generally desirable
- But there are operational challenges
  - HW encoding/decoding issue list of doom (20+)
- Chrome flags for disabling HW encoder/decoding
  - useful for support and debugging
  - but not available to JS application
- Can we change that?
  - while still somewhat discouraging usage of the new API
  - while avoiding increase in fingerprint surface
- At March 15 meeting, discussed extending setCodecPreferences() by adding hardwareAcceleration attribute from WebCodecs to RTCRtpCodecCapability dictionary
  - Is there a simpler and easier to implement way to do this?

Issue 98: Disabling hardware encoding/decoding

- partial interface RTCRtpSender {
  ```
  static void disableHardwareEncoding();
  ```
  }
  - One-way operation, unpleasant to use
  - changing your opinion requires a page reload
  - MUST be called before any peerconnection is created or fails silently
  - but does allow opt-out and A/B experimentation
- Should be covered by appropriate usage metrics
  - persistent large scale usage means either fingerprinting or a bug which should be fixed instead

Issue 112: header extension API: do we need enabled in setParameters? (fippo)

- setOfferedHeaderExtension API solves the problem of enabling non-default extensions (similar to issue 100)
- Can use setParameters to enable/disable (within the envelope) on the fly
  - What extension would **you** like to turn off?
  - Not implemented by Chrome implementation (rest is behind flag)
  - Possible to achieve with (local) renegotiation
- Remove until value is proven?

Issue 111: Integration of congestion control across SCTP and media (harald)

- It seems logical that one sending app should only get one shot at congesting a link, and should never self-congest
- Still, the SCTP (NewReno) and media (transport-cc) components of a PeerConnection are independent, and will fight
- Can we fix this?
- Is it important enough to fix?

Suggested answers: Maybe, and probably not.

Issue 111: Integration of congestion control across SCTP and media (continued)

An integrated CC could look like this:

- An “envelope generator” that uses both SCTP and RTCP feedback to estimate bandwidth
- A “prioritizer” that allocates bandwidth and/or congestion window to the two users according to priorities set

Solution will be invasive, and will need testing.

Apps will have to compete against TCP anyway.

Is it worth pursuing?

Discussion (End Time: 12:00 Noon)

-

Virtual Meeting Tips (Zoom)

- **Both local and remote participants need to be on irc.w3.org channel #webrtc.**
- **Type q+ and q- in IRC to get into and out of the speaker queue.**
- **To try out WebCodecs over RTCDatachannel (not RTP!) join using a Browser.**
- **Please use headphones when speaking to avoid echo.**
- **Please wait for microphone access to be granted before speaking.**
- **Please state your full name before speaking.**

September 13 Agenda

- 09:10 - 09:40 Encoded-Transform Challenges (Harald, 30 mins)
- 09:40 - 10:10 Encoded-Transform Issues (Youenn, 30 mins)
- 10:10 - 10:40 Conditional Focus (Elad, 30 mins)
- 10:40 - 11:10 Break (30 minutes)
- 11: 10 - 11:30 Screen-sharing Next Steps (Elad, 20 mins)
- 11:30 - 12:00 Topic TBD (30 mins)

Time control:

- A warning will be given 2 minutes before time is up.
- Once time has elapsed we will move on to the next item.

TPAC Health Rules

While physically attending TPAC, follow the Health Rules:

- Authorized masks are required indoors at all time (no exceptions)
  If you need to remove your mask during a meeting, keep it short (but keep enjoying that water/coffee)
- Daily COVID19 self-test is expected

Accommodate participants' needs for physical distancing and other accommodations or precautions due to health concerns

Lunch boxes will be provided for inside or outside. There is limited outdoor seating at the hotel but there are public parks within a 10-minute walk.

WebRTC Encoded Transform Challenges (Harald)

Start Time: 09:10

End Time: 09:40

For Discussion Today

- Beyond transform: Breaking open the pipeline
  - Issue 106: Add use cases that require one-ended encoded streams
- Pluggable codecs
  - Issue 90: Add feedback streams in RTCRtpScriptTransformer
- Congestion control
  - Issue 31 & Issue 50: Interaction with congestion control
- Packetization API
  - Issue 131: Should we expose packetization level API to RTCRtpScriptTransform?
- Depacketization
  - Issue 109 & Issue 119: Revisit question of whether frames need to be ordered on incoming

Issue 106: Add use cases that require one-ended encoded streams

- End to end encryption works with the stream processing model (mostly)
- Diverting the stream to other purposes is not possible in the current model
- Use cases abound:
    - SFU-in-the-browser
    - Alternative transports (WS, WebTransport)
    - Alternative encoders/decoders (WebCodecs)
    - Sending & receiving metadata along with frames (faces, AR…)
- Requirements rather than API surface should be a good starting point

Issue 90: Pluggable codecs

- This involves using WebCodecs, ECE or other processing mechanisms instead of WebRTC encoding/decoding to generate and parse data
- When integrating with WebRTC, it is important to ensure that SDP negotiation tells the truth of what is being sent across the wire
- This means that codec descriptions must be under application control.

Suggestion: We should allow codec descriptions to be injected into the SDP negotiation machinery, invoking known packetization/depacketization functions, and allow the app to figure out if the injected codec is acceptable to the remote party.

Transforms should operate in terms of codec descriptions, not PT.

Issue 31 & Issue 50: Congestion Control

- Possibility to append metadata in encoded stream on sender side
    - AR/VR may append large data
        - Potential bitrate increase

- User Agent target bitrate computation may not account for it
    - Is it just a User Agent implementation problem?

- Use cases where frames are injected in encoded form will generate congestion control confusion (stream of frame loss indicators, no bitrate measurements)

- What if the encoded transform could modulate metadata bitrate
    - Should the encoded transform be made aware of the target bitrate?
    - Should the encoded transform be able to hint at reducing the encoder bitrate?

Issue 131: Packetization API

- Use cases where WebRTC encoded transform is missing support
  - RED transform that hits the MTU limit
    - Expose MTU value (plus event) in transformer
  - Use SFrame at the packet level (SPacket) instead of frame level
    - Extension to RTCEncodedVideoFrame on sender side
      - Provide RTP packet payloads instead of encoded frame data
  - Read/Write access to RTP headers like voice activity
    - Extend RTCEncodedAudioFrame & RTCEncodedVideoFrame
      - Dedicated getters/setters of useful supported RTP headers

- Do we have enough of a use case to do any of this?

Issue 109 & Issue 119: Depacketization order

- With packet loss and retransmission, packets (and frames) may be reordered
- When decoding, frames must be in order (or be a keyframe)
- When transforming, it is simplest to do it in decode order
- This means jitter buffer happens before transforming, which does not allow compensation for app-induced jitter

Should we allow apps that receive frames out-of-order?

Discussion (End Time: 09:40)

-

WebRTC Encoded Transform Issues (Youenn)

Start Time: 09:40

End Time: 10:10

For Discussion Today

- Interaction with WebCodecs
  - Issue 99: Encoded frame IDLs share definitions with WebCodecs
  - Issue 141: Relationship to WebCodecs
  - Issue 70: FaceDetection metadata & VideoFrame

- generateKeyFrame()
  - Issue 143: generateKeyFrame takes a “rid” argument, but is invoked with “rids”

Issue 99 & Issue 141: WebCodecs & WebRTC

- Similarities between some structures
  - RTCEncodedAudioFrame vs. EncodedAudioChunk
  - RTCEncodedVideoFrame vs. EncodedVideoChunk

- Some differences
  - Data is mutable in WebRTC, not WebCodecs
  - WebRTC specific metadata needed

- Proposal: do not use WebCodec main types
  - Reuse WebCodec subtypes when feasible
    - RTCEncodedVideoFrameType ↔ EncodedVideoChunkType
    - Metadata

Issue 70: WebCodecs & MediaStream transform

- Desire for WebRTC pipeline to manipulate VideoFrames + metadata
  - Face detection metadata from camera up to encoder, via transforms
    - OS based Object detection?
  - AR/VR metadata generated by WebRTC encoded transform
    - Attached to VideoFrame by WebRTC decoders

- Requirements
  - Metadata readable/writable by WebRTC encoders/decoders
  - Metadata persisting VideoFrame cloning/transfer/serialization

- VideoFrame does not have (yet) a mechanism for generic metadata
  - What should we do?

Issue 70: WebCodecs & MediaStream transform

- Proposal 1: wrap VideoFrame around our own WebRTC object

- Proposal 2: design a generic mechanism API with WebCodecs WG
  - Add specific face detection metadata API in VideoFrame
    - In WebCodecs or in WebRTC specs
  - Metadata is mutable
    - Cloned when VideoFrame is cloned or when sent to encoder
  - Metadata is any JS that can be structure cloned

- Proposal 3: add specific metadata definition in WebRTC spec
  - Start with face detection
  - 2a: Add generic VideoFrame metadata mechanism in WebCodec spec
    - Do not expose it to the Web
  - 2b: Add specific VideoFrame metadata mechanism in WebRTC spec

Issue 143: generateKeyFrame

- Current specification is inconsistent
  - One generateKeyFrame targets a specific RID and returns a timestamp
  - Another takes a list of RIDs
- Problem
  - Some encoders will trigger key frames for all rids
  - Some applications might want to trigger key frame for some rids

- Proposal 1
  - Revert to pass an array of rids and not a single rid
    - Return no timestamp or timestamp of the lowest rid?
- Proposal 2
  - Keep existing, Let the UA deal with concatenating key frame requests
- Proposal 3
  - Providing no rid → Trigger key frames for all rids

Discussion (End Time: 10:10)

-

Conditional Focus (Elad)

Start Time: 10:10

End Time: 10:40

Recap

- Conditional Focus originally proposed in September 2021.
- API to conditionally focus captured tab/window when capture starts. (Why conditionally? Answered shortly.)
- Chrome has an active Origin Trial with satisfied partners.

Why Conditionally?

Applications come in many flavors. Some…

- … always never to focus the capturee. Some never.
- … want to focus all tabs, but no windows. Or the inverse.
- … always/never want to focus capturees of a specific origin.
- … want to make the focus decision depending on the contents of the Capture Handle exposed by the capturee.

All of these use-cases are legitimate.

The API should serve them all.

Recent Developments

Mozilla previously objected to exposing the API on the track.

They have proposed a controller object to be returned by gDM, on which multiple APIs could theoretically be exposed, Conditional Focus being one. It is backwards-compatible.

```
let controller = new CaptureController;
let stream = await
navigator.mediaDevices.getDisplayMedia({
  video: video_constraints,
  controller: controller
});
```

Default Behavior

When gDM is called without a controller - state of the art - focus behavior is unspecified.

- Chromium focuses
- Firefox focuses
- Safari has not yet implemented tab/window-capture

Open issues:

- Let’s keep this unspecified (gDM-no-controller focus).
- Shall we specify the default behavior if a controller is provided?
- Mozilla was interested in “unspecified behavior that matches in both cases” - does that make sense to the WG? (Contradictory with a later issue; continued in next slides.)

Focus at an Arbitrary Time

Developers have expressed interest in being able to focus at an arbitrary time. This is too dangerous. By switching focus at just the right time, this would allow capturing apps to trick the user into producing a specific action in the captured app (Web-based or native).

It is clear that the focus decision must be limited to the beginning of the capture-session.

Limiting Time to Focus, Option 1

Option #1: Purely timer-based

Idea:

- Calls to focus() are only valid up to N milliseconds after gDM resolves.

Analysis:

- Simple to specify and implement
- Allows advanced behavior, such as communicating with the captured content to determine whether to focus (if we add a message port to the Capture Handle - which we should anyway, see future presentations).
- Mutually-exclusive with default-no-focus (see next slides).

Limiting Time to Focus, Option 2 - Idea

Option #2: Window-of-Opportunity to focus closed by a scheduled task

Idea:

- Following a call to gDM:
  a) Schedule a task to resolve the gDM promise.
  b) Open the window-of-opportunity for a focus-decision.
  c) Schedule a task to close the window-of-opportunity.
- Calls to focus() are only valid from inside the window.
  a) Known-failures raise an exception. E.g. calling from outside the window.
  b) Unpredictable failures are no-op. E.g. if the call was made late due to a timer running out in another process.

(Analysis follows.)

Limiting Time to Focus, Option 2 - Analysis

Option #2: Window-of-Opportunity to focus closed by a scheduled task

Analysis:

- Forces developers to call focus() early - encourages less flaky code.
- In apps that don’t call controller.setFocusBehavior(), the default had better be no-focus, or else the users will always experience the focus-change after N seconds.

Bikeshedding

- If the default is to not focus, then just call it focus().
- If the default is unspecified or not-focus, then name along the lines of setFocusBehavior(enum).
- Or does this WG have other ideas?

Discussion (End Time: 10:40)

Break (End Time: 11:10 AM)

TPAC Health Rules

While physically attending TPAC, follow the Health Rules:

- Authorized masks are required indoors at all time (no exceptions)
  If you need to remove your mask during a meeting, keep it short (but keep enjoying that water/coffee)
- Daily COVID19 self-test is expected

Accommodate participants' needs for physical distancing and other accommodations or precautions due to health concerns

Lunch boxes will be provided for inside or outside. There is limited outdoor seating at the hotel but there are public parks within a 10-minute walk.

Screen-sharing Next Steps (Elad)

Start Time: 11:10

End Time: 11:30

Recently Introduced

- Region Capture:
  - But only specified for the current tab.
  - Can’t yet expose beyond the current tab.
- Capture Handle Identity, but

Discussion (End Time: 11:30)

Topic TBD

Start Time: 11:30 AM

End Time: 12 noon

Title Goes Here

- <Content goes here>

Discussion (End Time: noon)

-