

# **W3C WebRTC WG Meeting**

June 27, 2023  
8 AM - 10 AM

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

# Welcome!

- Welcome to the June 2023 interim meeting of the W3C WebRTC WG, at which we will cover:
  - Use Cases, IceController
- Future meetings:
  - July 18
  - September 12 (TPAC)
  - September 14 (SCCG Joint Meeting)
  - September 21 (MEDIA WG Joint Meeting)
  - October 17
  - November 21
  - December 12

# About this Virtual Meeting



- Meeting info:
  - [https://www.w3.org/2011/04/webrtc/wiki/June\\_27\\_2023](https://www.w3.org/2011/04/webrtc/wiki/June_27_2023)
- Link to latest drafts:
  - <https://w3c.github.io/mediacapture-main/>
  - <https://w3c.github.io/mediacapture-extensions/>
  - <https://w3c.github.io/mediacapture-image/>
  - <https://w3c.github.io/mediacapture-output/>
  - <https://w3c.github.io/mediacapture-screen-share/>
  - <https://w3c.github.io/mediacapture-record/>
  - <https://w3c.github.io/webrtc-pc/>
  - <https://w3c.github.io/webrtc-extensions/>
  - <https://w3c.github.io/webrtc-stats/>
  - <https://w3c.github.io/mst-content-hint/>
  - <https://w3c.github.io/webrtc-priority/>
  - <https://w3c.github.io/webrtc-nv-use-cases/>
  - <https://github.com/w3c/webrtc-encoded-transform>
  - <https://github.com/w3c/mediacapture-transform>
  - <https://github.com/w3c/webrtc-svc>
  - <https://github.com/w3c/webrtc-ice>
- Link to Slides has been published on [WG wiki](#)
- Scribe? IRC <http://irc.w3.org/> Channel: [#webrtc](#)
- The meeting is (still) being recorded. The recording will be public.
- 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

# Virtual Interim Meeting Tips

**This session is (still) being recorded**

- Click  Raise hand to get into the speaker queue.
- Click  Lower hand to get out of the speaker queue.
- Please wait for microphone access to be granted before speaking.
- If you jump the speaker queue, you will be muted.
- Please use headphones when speaking to avoid echo.
- Please state your full name before speaking.
- Poll mechanism may be used to gauge the “sense of the room”.

# 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.

# Issues for Discussion Today

- 08:10 - 08:30 AM Mediacapture-screen-share (Elad)
- 08:30 - 08:40 AM A message from our sponsor (Fippo)
- 08:40 - 09:00 AM WebRTC NV-Use Cases (Bernard & Tim)
- 09:00 - 09:20 AM IceController (Sameer & Peter)
- 09:20 - 09:50 AM Encoded Transform Codec Negotiation (Harald)
- 09:50 - 10:00 AM Wrapup and Next Steps (Chairs)

## 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.



# **Mediacapture-screen-share (Elad)**

**Start Time: 08:10 AM**

**End Time: 08:30 AM**

# For Discussion Today

- Make CaptureController inherit from EventTarget ([Issue 268](#))
- Improve upon CaptureStartFocusBehavior.no-focus-change ([Issue 263](#))
- Allow apps to avoid riskier display-surface types ([Issue 261](#))

# Désolé

“The present letter is a very long one,  
simply because I had no leisure to make it shorter.”

– Blaise Pascal

## Issue 268: Make CaptureController inherit from EventTarget

### CaptureController...

- ...was recently introduced.
- ...is optional.
- ...is immutably associated with a “capture session.”
- ...is only currently used to expose setFocusBehavior().

Inheriting from EventTarget can only be properly done in the original spec.

- Immediate potential usage by [Screen-Capture Mouse Events](#).
- Potential usage for other hypothetical uses:
  - Events when captured surface changes.
  - Events when the user pauses the capture through the UA or OS.
  - Events if the app wants to register a handler for a capture initiated from the captured surface, as macOS Sonoma allows.

## Issue 263: Improve upon `CaptureStartFocusBehavior.no-focus-change`

`CaptureController.serFocusBehavior(enum)` accepts two possible enum values:

Enumeration description	
<code>focus-captured-surface</code>	The application prefers that the <a href="#">display surface</a> associated with this <code>CaptureController</code> 's <a href="#">capture-session</a> be focused.
<code>no-focus-change</code>	The application prefers that the user agent not change focus.

- The former essential means “focus the captured tab or window.”
- The latter...
  - Intended - “keep the capturing application focused.”
  - Result - unclarity in the case of Safari, because the user-flow of picking is composed of interaction with the to-be-captured window.

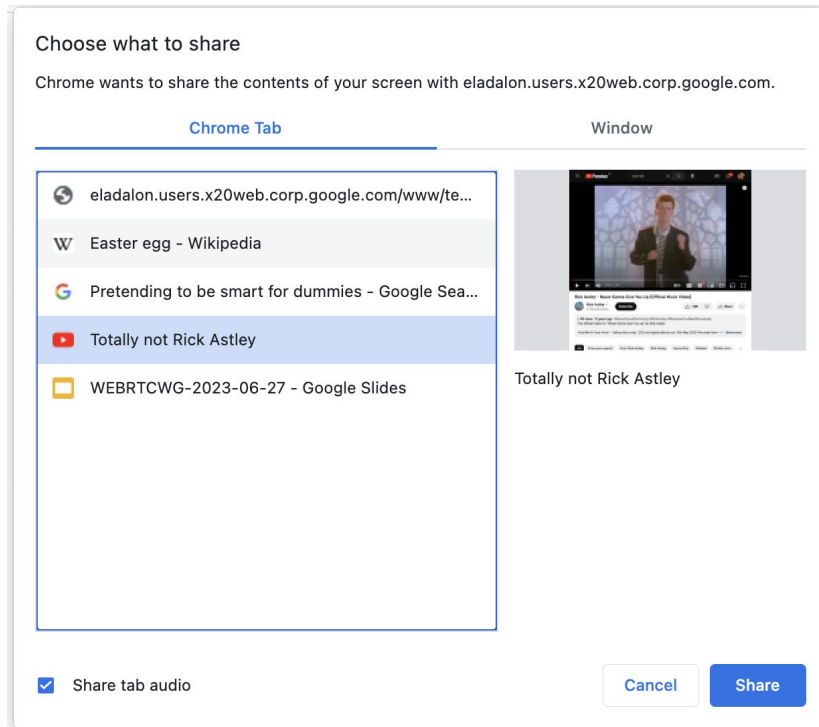
## Issue 263: Proposal

1. Add “focus-capturing-application”
2. Existing implementations retain “no-focus-change” and it’s redefined as “keep in focus whichever surface was last focused following the user’s interaction with the UA and/or OS.”
3. Longer-term, deprecating “no-focus-change” is on the table.

# Issue 261: Allow apps to avoid riskier display-surface types

Proposal - allow applications calling `getDisplayMedia()` to request the user agent to exclude monitors from the selection offered to the user.

```
enum MonitorTypeSurfacesEnum {  
  "include",  
  "exclude"  
};  
  
dictionary DisplayMediaStreamOptions {  
  MonitorTypeSurfacesEnum monitorTypeSurfaces;  
};
```



## Issue 261: Motivation

A hypothetical company called HypComp uses a hypothetical video-conferencing application called VC-app.

For simplicity, let's assume that the list of users can only be set before the call. Now:

- All participants are HypComp employees? Allow them to share anything.
- Externals present? Minimize risk to company IP by preventing full-screen sharing.

Considered alternatives:

- Can't use a policy; users will join multiple calls throughout their day.
- VC-app could call `getDisplayMedia()` again, but that will perplex and frustrate users.

Added benefit:

- If the UA allows dynamic switching between windows and screens, as Safari does, then it's helpful to suppress this option. VC-app cannot react dynamic switching in time and pause the capture otherwise (at least not without [auto-pause](#); shameless plug).



## Issue 261: Security

- The spec disallows constraining the choice offered to the user.
- The rationale is that it protects the user from being nudged towards sharing something risky with a malicious application.
- The riskiest option is the current screen:
  - Contains the current tab, which is under the capturer's control.
  - Contains the maximum other information.
- A precedent where we allowed constraining the selection is [selfBrowserSurface](#), after which the current proposal is modelled. The rationale employed there applies here too.

# Discussion (**End Time: 08:30**)

**A message from our sponsor (Fippo)**

**Start Time: 08:30 AM**

**End Time: 08:40 AM**

# Requesting keyframes via setParameters [#167](#)

- 🙅 on adding to encoding parameters
  - Not persistent so not a parameter
- Proposal: add a second parameter to setParameters
  - `setParameters(RTCRtpSendParameters parameters, optional sequence<VideoEncoderEncodeOptions> encodeOptions);`
  - Still explicit and in sync with other setParameters changes
  - Sequence of the same length as parameters.encodings
  - [VideoEncoderEncodeOptions](#) from WebCodecs
  - First time we reuse WebCodecs IDL in WebRTC directly?
  - Do we want to go in this direction?

```
dictionary VideoEncoderEncodeOptions {  
    boolean keyFrame = false;  
};
```

# Discussion (**End Time: 08:40**)

# **WebRTC-NV Use Cases (Bernard & Tim)**

**Start Time: 08:40 AM**

**End Time: 09:00 AM**

# Proposals from the May Meeting

- **Rename it. Proposal: “WebRTC Extended Use Cases”. Done.**
- Focus on things that can only/best be done by WebRTC (p2p etc)
- Remove use cases that are now met by other standards
- Include use cases that have no requirements but extend RFC 7478
- Remove use cases that don't get consensus within a few months
- Remove requirements that don't get consensus within a few months
- **Remove use cases that don't add new requirements**
- Proposed API changes should include changes to the use-case doc
- Define the relationship between this doc and explainers
- Broaden the input somehow - perhaps via webrtc.nu ?

# Remove Use Cases That Don't Add New Req'ts

- [PR 112](#): Remove Section 3.9: Reduced Complexity Signaling
- [PR 113](#): Remove Machine Learning Use Case (Section 3.7)



# PR 112: Remove Section 3.9: Reduced Complexity Signaling

- Use case has no requirements.
- Use case has been partially addressed via the WHIP protocol developed within the IETF WISH WG.

## § 3.9 Reduced complexity signalling

Some simpler media/data sources/sinks require only a subset of WebRTC functionality.

In such use cases the full SDP O/A could be replaced with a server generated token at runtime.

A goal might be to set the source attribute of a video tag to point at a WebRTC video source.

Care needs to be taken to avoid introducing security vulnerabilities.

### **NOTE**

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

Requirement ID	Description
	TBD

Experience: Both *pipe* and *pion* have built systems of this sort.

## PR 113: Remove Machine Learning Use Case (Section 3.7)

- Use case does not add any requirements beyond those for the Funny Hats use case (Section 3.6).
- This use case includes Requirement N22 (efficient media manipulation via GPUs) that seems questionable:
  - Efficient media manipulation != efficient machine learning
  - GPUs are not the only way to accelerate Machine learning
    - WASM SIMD?
    - NPUs?
  - Machine learning performance is not in scope for WEBRTC WG:
    - Machine Learning APIs are developed in the [Machine Learning WG](#)
    - [WebGPU WG](#) owns APIs relating to GPUs.
    - WASM is developed in the [WASM WG](#).
- Can we remove requirement N22 entirely?
  - Req't makes more sense for Funny Hats, but is it actionable in this WG?

# PR 113: Remove Machine Learning Use Case (cont'd)

## 3.7 Machine Learning

In a web game called “NameTheBird.com” participants use their devices to provide audio and video observations of birds to the service along with identifications for training purposes, allowing the service to identify birds from the provided audio and video and returning this information to the users in real-time.

The web application has a site specific federated learning-based classifier for contextual object detection, user intent prediction and media manipulation, allowing it to augment the streams it receives and inject identifying or other supplemental information into the streams sent or received.

The shared classification models are trained on the birds found by the participants and are based on the feedback of the participants. Each device client updates of the model are up-streamed to a shared model server that pushes updates of the global model to the clients.

# PR 113: Remove Machine Learning Use Case (cont'd)

Implementation outline:

1. Originating media (raw) streams are cloned for inference and training purposes, denoted “inference stream” and “training stream”, with the inference stream also being the media stream shared with peer(s). The cloning can occur any time during a session.
2. Inference stream: A web site specific classifier acts on the raw inference stream, with the result used to guide a custom encoder in the sender device and send metadata to the server and peer devices outside the media stream. The encoder adds proper augmentation, e.g. sign with “name this bird” hovering over the enlarged bird in case of video enrichment, or enhanced bird song if audio.
3. Training stream: Model in training classifies the raw data and evaluate the classification using user feedback, said feedback loop being web site specific. The evaluation may be “online” or “offline”, offline meaning the training is done at a later stage on the recorded encoded media set.
4. Both inference stream and training streams may use payload protection depending on trust model on compute resources for optional intermedia server side of app.
5. Both inference stream and training streams use transport object for communicating with peers or servers, the communication in some cases can be a site specific QUIC based transport solution, in others RTP based.

## PR 113: Remove Machine Learning Use Case (cont'd)

This use case adds the following requirements:

Requirement ID	Description
N18	The application must be able to obtain raw media from the capture device in desired formats.
N19	The application must be able to insert processed frames into the outgoing media path.
N20	The application must be able to obtain decoded media from the remote party.
N21	It must be possible to efficiently share media between the main thread and worker threads.
N22	It must be possible to do efficient media manipulation in worker threads by utilizing the GPU.
N24	Content Security Policy (CSP) support for WebRTC.

# Proposals from the May Meeting

- Rename it. Proposal: “WebRTC Extended Use Cases”. **Done.**
- Focus on things that can only/best be done by WebRTC (p2p etc)
- Remove use cases that are now met by other standards
- Include use cases that have no requirements but extend RFC 7478
- Remove use cases that don't get consensus within a few months
- Remove requirements that don't get consensus within a few months
- Remove use cases that don't add new requirements
- **Proposed API changes should include changes to the use-case doc**
- **Define the relationship between this doc and explainers**
- Broaden the input somehow - perhaps via webrtc.nu ?

# Process changes

- **Proposed API changes should include changes to the use-case doc**
- **Define the relationship between this doc and explainers**
  
- Do we agree in principle to these ?
  
- If so where should a PR describing them reside?

# Discussion (**End Time: 09:00**)



**IceController (Sameer & Peter)**

**Start Time: 09:00 AM**

**End Time: 09:20 AM**

# Issue 166 - prevent candidate pair removal

## Use case: connection redundancy

Keep one or more backup connections around for if/when the active connection deteriorates

Extend with further improvements - switch to a backup connection without an ICE restart or waiting for an ICE disconnect

Now: PR 168

# PR 168 - prevent candidate pair removal

## WebIDL

```
partial interface RTCIceTransport {  
  attribute EventHandler onicecandidatepair;  
  attribute EventHandler onicecandidatepairprune;  
};
```

## WebIDL

```
[Exposed=Window]  
interface RTCIceCandidatePairEvent : Event {  
  constructor(DOMString type, RTCIceCandidatePairEventInit eventInitDict);  
  readonly attribute RTCIceCandidate local;  
  readonly attribute RTCIceCandidate remote;  
};
```

Event name	Interface	Fired when...
<i>icecandidatepair</i>	<a href="#">RTCIceCandidatePairEvent</a>	The ICE agent has formed a candidate pair and is making it available to the script.
<i>icecandidatepairprune</i>	<a href="#">RTCIceCandidatePairEvent</a>	The ICE agent has selected a candidate pair to prune, which will be pruned unless the operation is canceled by invoking the <code>preventDefault()</code> method on the event.

# WebRTC ICE incremental improvements

- Prevent removal of candidate pairs - [Issue 166](#)
- Remove candidate pairs - [Issue 170](#)
- Control selection of candidate pair - [Issue 171](#)
- (?) Observe candidate pair states
- Observe result/RTT of outgoing checks
- Control frequency of outgoing checks of particular candidate pairs
- Prevent outgoing checks of particular candidate pairs
- Control order and timing of outgoing checks
- Observe presence of not of incoming checks or media for particular candidate pairs
- Gather local candidates for new network interfaces
- Re-gather local candidates of previously failed network interfaces
- Prevent removal of local candidates
- Remove local candidates
- Construct IceTransport without PeerConnection
- Support forking

# Issue 170 - remove candidate pairs

## Use case: clean up redundant connections

- Release local and network resources taken up by unnecessary candidates
- Stop sending STUN checks on the removed candidates

```
partial interface RTCIceTransport {  
    undefined pruneCandidatePairs(sequence<RTCIceCandidatePair> pairs);  
}
```

# Issue 171 - select a candidate pair

## Use case: switch to a redundant connection

- switch to a better connection without an ICE restart, or without waiting for an ICE disconnect

```
partial interface RTCIceTransport {  
    Promise<undefined> setSelectedCandidatePair(RTCIceCandidatePair pair);  
}
```

# Discussion (End Time: 09:20)

- Add RTCIceCandidatePair interface
  - breaking change dictionary→interface
    - local / remote are not constants in dictionary
  - Alternatively, interface by a new identifier?
  - Alternatively, type any for candidatePair attributes
- Return promise or undefined from prune() / setSelected()
  - prune() isn't async, so return immediately
  - setSelected() asynchronously indicates completion, so promise
    - Alternatively, return undefined, and let selectedCandidatePairChange event indicate completion

# Encoded Transform Codec Negotiation (Harald)

**Start Time: 09:20 AM**

**End Time: 09:50 AM**



# Negotiating Custom Codecs

(recap from March)

Transforming content and being truthful  
about it

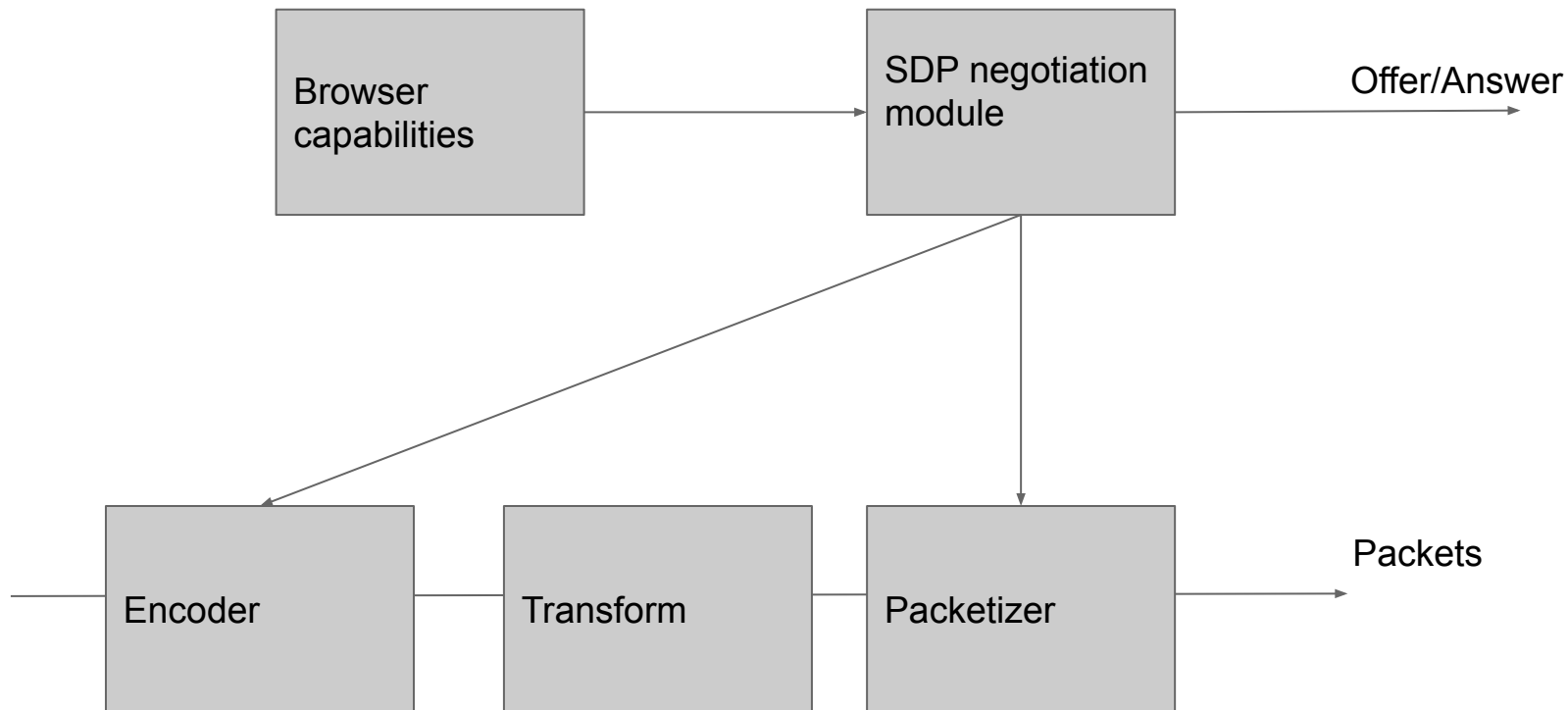
[Issue #172](#)

# The Problem with Encoded Transform

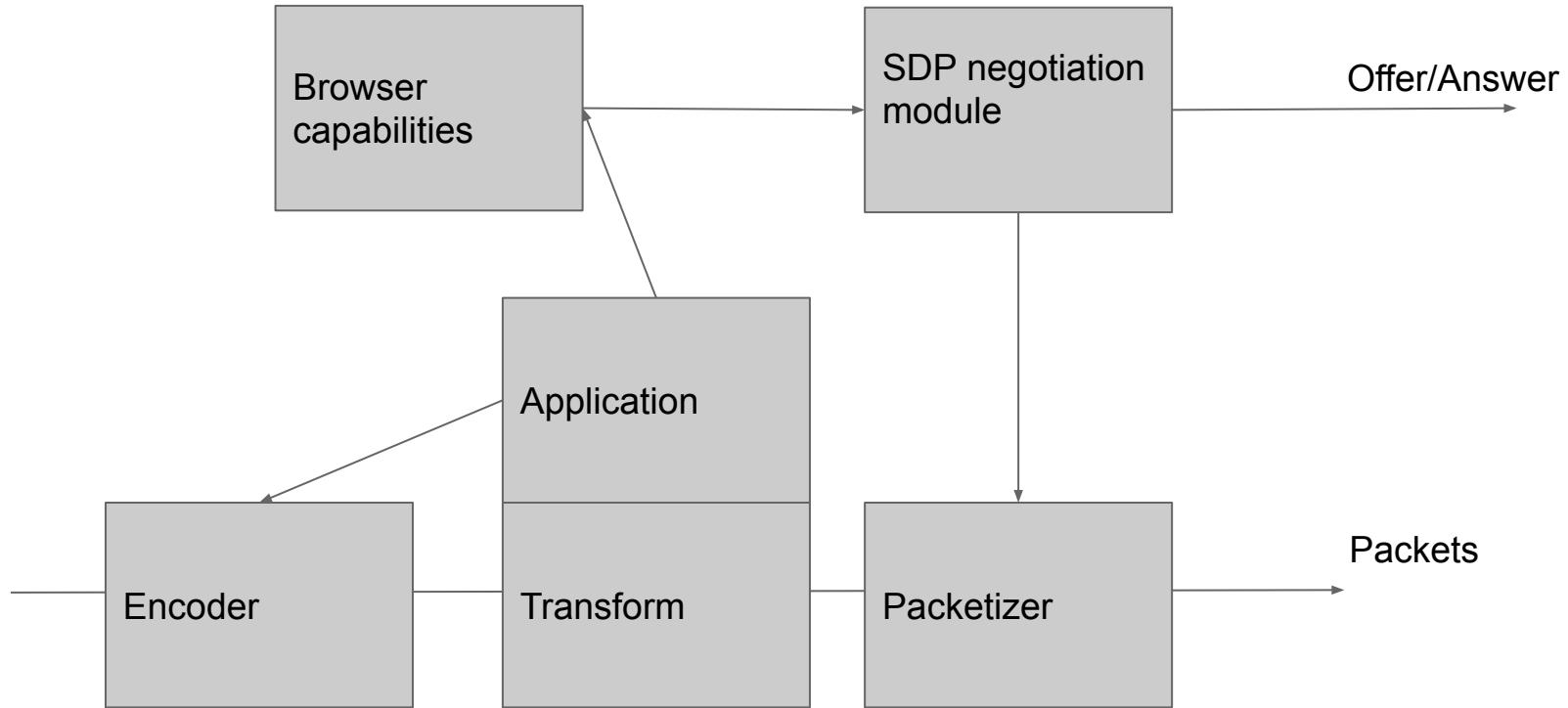
- An app sets up a connection, negotiating a set of codecs
- The app sender inserts a transform, changing the format on the wire
- The app receiver reverses the transform
- Problem: Elements on the way (SFUs, packetizers) expect the negotiated format, not the transformed format
- This complicates things. Complexity is bad.

Solution: Negotiate what you send.

# The Encoded Transform model



# The Enhanced Encoded Transform model



# Operations in the Enhanced Encoded Transform

- The application **tells the SDP module about new formats**
- The application **tells the encoder what format to encode to**
- The other modules of the system **operate as before**

In particular:

- The SDP module negotiates over the set of known formats, with the normal controls over what format to select
- The encoder encodes to a supported format
- The packetizer is configured by the SDP module as before

(The receiving side functions similarly)

# New APIs needed to achieve this functionality

New information needed about codecs - this allows SDP to configure the packetizer

```
partial dictionary RTCRtpCodecCapability {  
    DOMString packetizationMode;  
}
```

Pre-negotiation calls - these allow SDP to negotiate support of “custom” codecs

```
PeerConnection.addSendCodecCapability(DOMString kind, CodecCapability capability)  
PeerConnection.addReceiverCodecCapability(DOMString kind, CodecCapability capability)
```

After creating senders and receivers - these allow the app to select the encoder and PT->decoder mapping

```
RTCRtpSender.setEncodingCodec(RTCCodecParameters parameters) // Alternatively, extensions PR #147  
RTCRtpReceiver.addDecodingCodec(CodecParameters parameters)
```

# Example Code - Sender

```
customCodec = {  
  mimeType: "video/acme-encrypted",  
  clockRate: 90000,  
  sdpFmtpLine = "encapsulated-codec=vp8",  
  packetizationMode = "video/vp8",  
};
```

```
pc.addSenderCodecCapability('video', customCodec);  
sender = pc.AddTrack(videotrack);  
// Negotiate as usual  
for (codec in sender.getParameters().codecs) {  
  if (codec.mimeType == "video/acme-encrypted") {  
    encryptedPT = codec.payloadType;  
  }  
}  
if (!encryptedPT) { /* failure; don't encrypt */ return; }  
(readable, writable) = sender.getEncodedStreams();  
  
readable.pipeThrough(new TransformStream(  
  transform: (frame) => {  
    metadata = frame.metadata();  
    if (metadata.payloadType == expectedPT) {  
      encryptBody(frame);  
      metadata = frame.metadata();  
      metadata.pt = encryptedPT;  
      frame.setMetadata(metadata);  
      writable.write(frame);  
    } // "Else" branch depends on application  
  }  
}).pipeTo(writable);
```

# Example Code - Receiver

```
pc.AddReceiverCodecCapability(customCodec);
// Negotiation goes here
pc.ontrack = (receiver) => {

    for (codec in receiver.getParameters().codecs) {
        if (codec.mimeType == "video/acme-encrypted") {
            encryptedPT = codec.payloadType;
        }
    }
    if (!encryptedPT) { /* Failure, don't decrypt */ return; }
    receiver.addDecodingCodec({mimeType: video/vp8, payloadType=208});
    (readable, writable) = receiver.getEncodedStreams();
    readable.pipeThrough(new TransformStream(
        transform: (frame) => {
            metadata = frame.metadata();
            if (metadata.payloadType == encryptedPT) {
                decryptBody(frame);
                metadata.payloadType = 208;
            } // "Else" branch will depend on application
            writable.write(frame);
        }
    ).pipeTo(writable);
};
```



## New: [PR 186](#)

- Closely follows proposal from March
- Adds an explainer and some API changes
- Needs some exports from webrtc-pc
- Implementation started in March at IETF hackathon; not yet functional, but no showstoppers found
- Proposing this API for adoption.

# Discussion (**End Time: 09:50**)

# Wrap Up and Next Steps

**(End Time: 10:00)**

- Next step 1
- Next step 2

# Name that Bird

# Thank you

Special thanks to:

WG Participants, Editors & Chairs