

W3C WebRTC WG Meeting

January 17, 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 January 2023 interim meeting of the W3C WebRTC WG, at which we will cover:
 - WebRTC-PC, WebRTC-Extensions, Encoded Transform, Capture Handle, Auto-pause Capture, MediaStreamTrack frame rates
- Future meetings:
 - February 21
 - March 21
 - April 18
 - May 16
 - June 27

About this Virtual Meeting



- Meeting info:
 - https://www.w3.org/2011/04/webrtc/wiki/January_17_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.

Call for Consensus (CfC) Status

- [WebRTC-NV “Low Latency Streaming” Use Cases \(Section 3.2\)](#)
 - Responses received:
 - Support: Bernard, Tim Panton, Harald, Jan-Ivar
 - Abstain: Youenn
 - Issues referred to: 80, 85, 86, 91, 94, 95
- [Face Detection API \(PR 78\)](#)
 - Responses received:
 - Support: Silvia, Harald, Tim Panton, Youenn, Jan-Ivar
 - Object: Bernard
 - Issues filed: PR 78 comments, 79

Issues for Discussion Today

- 08:10 - 08:30 AM WebRTC-PC (Fippo & Jan-Ivar)
- 08:30 - 08:50 AM WebRTC-Extensions (Florent & Fippo)
- 08:50 - 09:10 AM Encoded-Transform (Harald)
- 09:10 - 09:15 AM Screen Capture Community Group (Elad Alon)
- 09:15 - 09:25 AM De-adopting Capture Handle Identity (Elad Alon)
- 09:25 - 09:40 AM Auto-pause Capture (Elad Alon)
- 09:40 - 09:50 AM MediaStreamTrack frame rates (Henrik)
- 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.

WebRTC-PC (Fippo & Jan-Ivar)

Start Time: 08:10 AM





End Time: 08:30 AM

For Discussion Today

- [Issue 2795](#): Missing URL in RTCIceCandidateInit
- [Issue 2780](#): duplicate rids in sRD underspecified
- [PR 2801](#): Prune createAnswer()'s encodings and `[[SendEncodings]]` in sLD(answer).

Issue 2795: Missing URL in RTCIceCandidateInit

- Added url and relayProtocol to RTCIceCandidate
 - These are not possible to reconstruct and only available for local candidates
 - Not serialized by toJSON, not to be signaled
- [4.8.1](#): “...the remaining attributes are derived from parsing the candidate”
 - Not updated when adding the new properties
- Proposal:
 - update description in 4.8.1
 - Write more tests!

Subtest	 Chrome 109 Linux 20.04 c88ed1f Dec 1, 2022	 Edge 109 Windows 10.0 c88ed1f Dec 1, 2022	 Firefox 109 Linux 20.04 c88ed1f Dec 1, 2022	 Safari 159 preview macOS 12.6 c88ed1f Dec 1, 2022
Harness status	OK	OK	OK	OK
Duration	0.346 seconds	0.49 seconds	0.247 seconds	0.569 seconds
RTCIceCandidate.toJSON serializes only speci...	FAIL	FAIL	PASS	FAIL
Subtest Total	0/1	0/1	1/1	0/1

Issue 2780 / PR 2800: duplicate rids in sRD underspecified

Proposal: Remove duplicate rids in `proposedSendEncodings`:

2. For each encoding, *encoding*, in *proposedSendEncodings* in reverse order, if *encoding*'s rid matches that of another encoding in *proposedSendEncodings*, remove *encoding* from *proposedSendEncodings*.

PR 2801: Prune createAnswer()'s encodings and `[[SendEncodings]]` in `sLD(answer)`.

A follow-up to [#2758](#) whose intent was to defer pruning of `[[SendEncodings]]` to `sLD(answer)`, but mistakenly relied on the spec's existing pruning language which only applies to `sRD(answer)`.

Add similar language to `sLD(answer)`:

7. If *description* is of type "[answer](#)" or "[pranswer](#)", then run the following steps:

1. If *transceiver*.`[[Sender]].[[SendEncodings]]` .length is greater than 1, then run the following steps:

1. If *description* is missing all of the previously negotiated layers, then remove all dictionaries in *transceiver*.`[[Sender]].[[SendEncodings]]` except the first one, and skip the next step.
2. If *description* is missing any of the previously negotiated layers, then remove the dictionaries that correspond to the missing layers from *transceiver*.`[[Sender]].[[SendEncodings]]`.

Next, we need to touch where this *description* comes from (next slide: createAnswer)

PR 2801: Prune createAnswer()'s encodings and [[SendEncodings]] in sLD(answer).

Fix [final steps to create an answer](#) to prune based on JSEP's answer \cap [[Sender]].[[SendEncodings]], instead of parroting create offer:

2. If the length of the [\[\[SendEncodings\]\]](#) slot of the [RTCRtpSender](#) is larger than 1, then for each encoding given in [\[\[SendEncodings\]\]](#) of the [RTCRtpSender](#), add an `a=rid` send line to the corresponding media section, and add an `a=simulcast:send` line giving the RIDs in the same order as given in the [encodings](#) field. No RID restrictions are set.

2. If this is an answer to an offer to receive simulcast, then for each media section requesting to receive simulcast, exclude from the media section in the answer any RID not found in the corresponding transceiver's [\[\[Sender\]\].\[\[SendEncodings\]\]](#). If there are any identically named RIDs in the `a=simulcast` attribute, remove all but the first one. No RID restrictions are set.

NOTE

When a [setRemoteDescription\(offer\)](#) establishes a transceiver's [simulcast envelope](#), the transceiver's [\[\[Sender\]\].\[\[SendEncodings\]\]](#) is updated in "[have-remote-offer](#)". However, once a simulcast envelope has been established for the transceiver, subsequent pruning of the transceiver's [\[\[Sender\]\].\[\[SendEncodings\]\]](#) happen when this answer is set with [setLocalDescription](#).

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

-

WebRTC-Extensions

Start Time: 08:30 AM

End Time: 08:50 AM

For Discussion Today

- [Issue 43](#): Mixed Codec Simulcast (Florent & Bernard)
- [Issue 126](#): Negotiationless codec selection (Florent)
- [Issue 127](#): How to deal with encoder errors? (Henrik)
- [Issue 130](#): how does setOfferedRtpHeaderExtensions work? (Fippo)

[Issue 43/PR 139](#): Mixed Codec Simulcast

- Florent presented Issue 43 (mixed codec simulcast) at the July 08, 2020 WebRTC WG meeting
 - Use case: hw codec for some (low res) layers, software codec for other layers.
 - Example: a low resolution simulcast stream encoded with AV1, and higher resolution layers encoded with VP8/VP9.
 - Advantages
 - Doesn't require multiple senders, can provide rids.
 - Compatible with `addTransceiver()`.
 - Allows resource allocation and graceful degradation between layers as opposed to having multiple senders with different codecs.
 - Allows switching codecs without O/A, using `setParameters()`.
- [Resolution](#): “Florent to submit a PR”
 - [PR 139](#) under development.

[Issue 43/PR 139](#): Mixed Codec Simulcast (cont'd)

- The original (ORTC) mechanism was:

```
partial dictionary RTCRtpCodingParameters {  
    payloadtype          codecPayloadType;  
};
```

- WebRTC Issue: when `addTransceiver()` is called prior to `createOffer()`, the codec payload type is not known (since it's negotiated via Offer/Answer).
- Alternative in [PR 139](#):

```
partial dictionary RTCRtpEncodingParameters {  
    RTCRtpCodecCapability          codec;  
};
```

- Codecs come from capabilities such as `RTCRtpSender.getCapabilities('video').codecs`.
- Doesn't use payload types (only known after O/A), so compatible with `addTransceiver()`.

PR 139: sendEncodings Example

```
sendEncodings: [  
  {  
    rid: 'q',  
    scaleResolutionDownBy: 4.0,  
    codec: [  
      {clockRate: 90000, mimeType: "video/AV1"},  
      {clockRate: 90000, mimeType: "video/VP8"}  
    ],  
  },  
  {  
    rid: 'f',  
    scaleResolutionDownBy: 1.0,  
    codec: [  
      {clockRate: 90000, mimeType: "video/VP8"}  
    ],  
  },  
]
```

5.2.10 RTCRtpCodecCapability Dictionary

WebIDL



```
dictionary RTCRtpCodecCapability {  
  required DOMString mimeType;  
  required unsigned long clockRate;  
  unsigned short channels;  
  DOMString sdpFmtpLine;  
};
```

Issue 126: Negotiation-less Codec Control: Problem

- Applications want to be able to select which codec is used.
- Currently only possible to achieve this with a renegotiation reordering or removing the unwanted codecs:
 - Call `setCodecPreferences()` with the new codec
 - Do a multi steps negotiation (`sLD()`, send offer, wait for answer or fake it and `sRD()`)
 - Update the `scalabilityMode` to be adequate with the new codec capabilities
- This process is heavy (need to do a round-trip with the server), racy (so many things could happen between each step) and not optimal (the SVC mode after `sRD` may not be relevant and may take time to change)

Issue 126: Negotiation-less Codec Control: Proposal

- Leverage solution in [PR 139](#): Add a codec member to `RTCRtpCodingParameters` of type `RTCRtpCodecCapability`.

- **Example:**

```
// Changing current codec
parameters.encodings[0].codec = {clockRate: 90000, mimeType: 'video/AV1'};
  // Equivalent to: RTCRtpSender.getCapabilities('video')
  //           .codecs.filter(c => c.mimeType == 'video/AV1')[0];
parameters.encodings[0].scalabilityMode = 'L3T3';

// Mixed-codec simulcast (Issue 43)
pc.addTransceiver(stream.getVideoTracks()[0], {
  sendEncodings: [
    {rid: 'q', codec: {clockRate: 90000, mimeType: 'video/AV1'},
      scaleResolutionDownBy: 4.0}
    {rid: 'f': {clockRate: 90000, mimeType: 'video/VP8'}},
  ]
});
```

Issue 126: Negotiation-less Codec Control: Questions

- Behavior on renegotiation to be defined. What should happen if the selected codec is removed from the SDP?
 - Proposal 1: Error in sLD/sRD
 - Proposal 2: Remove codec value, use first negotiated codec
- Should codec be a single value of `RTCRtpCodecCapability` or should it be an array?

Neither proposal help with selecting a `scalabilityMode` per codec automatically in case of fallbacks during renegotiation.

 - Proposal 1: Keep it simple with a single value
 - Proposal 2: Make it an array to simplify the renegotiation problem.
- Is renegotiation a problem for application developers with such an API?

Issue 127: How to deal with encoder errors? (Henrik)

When `setParameters()` fails, we can reject the promise and revert behavior.

But what should happen if encoder errors occurs later?

The specification doesn't say.

- The app may desire different (codec, scalabilityMode, #active layers) for different codecs, so arguably there is no “sensible default”.
 - *A list of codecs does not work either, these are combinations.*
- The browser doesn't know what the app wants, so why delegate this to the browser? The app can decide...
 - APIs already exist for taking action: call `setParameters()` again.
 - What is missing is an error callback. (This can be an opt-in.)
In WebCodecs, there is a [WebCodecsErrorCallback](#) for encoder or decoder errors.

Issue 127: How to deal with encoder errors? (Henrik)

Proposal:

- Set `active=false` on all layers and notify the app via callback.
- Let's handle all errors this way. Including if a codec previously selected via `setParameters()` is removed from the codec list in a negotiation.

```
sender.onencodeerror = async (/*DOMException*/ error) => {  
    await sender.setParameters(  
        /* app-specific combinations... */  
    );  
};
```

- Let's also add `RTCRtpReceiver.ondecodeerror`.

BONUS: We can `setParameters()` with a codec *prior to first negotiation*, because if negotiation fails we know how handle that.

Issue 130: how does setOfferedRtpHeaderExtensions work?

- Misunderstanding how header extension API is supposed to work
 - Due to lack of example
- Harald:
 - Call with a array of modified extensions with different direction
 - No need to pass all supported ones, not transactional
- Philipp:
 - Modify and filter from headerExtensionsToOffer
 - Similar to setCodecPreferences
- What model does the working group prefer?
 - (will resolve the other issues after the direction is clear)

Discussion (**End Time: 08:50**)

-

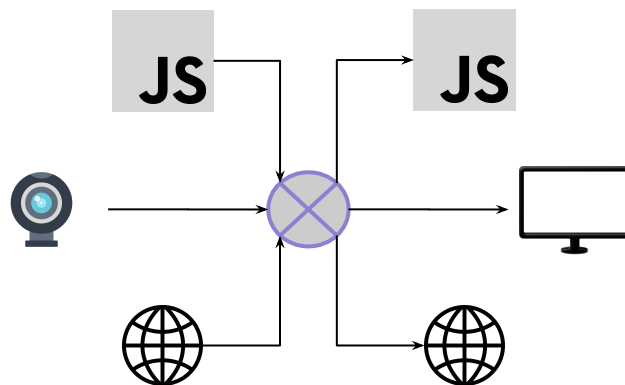
Encoded Transform (Harald)

Start Time: 08:50 AM

End Time: 09:10 AM

Encoded Media Manipulation - beyond Bump-In-Stack

- Unlocks several interesting use cases
 - Relay without decoding/decrypting
 - Send over non-RTP transports
 - Send pre-encoded media
- [Issues](#) / [PR](#) on nv-use-cases
- [PR](#) for explainer to encoded-media



API design - Use cases

- Use cases added to [nv-use-cases](#)
 - 3.10.1 [Live encoded media](#) - example: Non-WebRTC cameras
 - 3.10.2 [Transmitting stored encoded media](#) - example: “Music-on-hold”
 - 3.10.3 [Decoding pre-encoded media](#) - example: adding media over datachannels to webrtc-based apps

API design - requirements

Additional requirements: N40-N47

- Create senders without encoders, and receivers without decoders (N40, N45)
- Create frames and enqueue them (N41, N46)
- Handle control signals (N42, N47)
- Modify frame metadata to allow “splice” (N43, N44)

API design - Frame handling

- Creating and modifying frames (N41, N46, N43, N44)
- `New Frame(metadata, data)`
- `frame.getMetadata()` → Modify → `frame.setMetadata`
- Data modification is asynchronous
- Open question: When is data/metadata consistency checked? (if ever?)

API design - Frame queues and controls

- Using streams for the frames themselves has been successful so far
- Reconfiguration requests are event-like
- Would like symmetry between app-defined and platform-defined objects
- Proposal: Define an interface

API design - interface object

From proposal at the [IETF hackathon](#) (November 2022):

```
interface EncodedFrameSource {  
    ReadableStream source;  
    undefined requestKeyframe();  
    undefined requestBandwidth(BandwidthEvent);  
    undefined requestResolution(ResolutionEvent);  
};
```

This API can be offered by a platform object or by a Javascript object.

For interfacing to the existing objects:

```
partial interface RTCRtpSender { // similar for receiver  
    EncodedFrameSource DivertFrames(); // decouples encoder from packetizer  
    undefined insertFrames(EncodedFrameSource); // couples source object to packetizer  
};
```

Any object that receives frames will have an insertFrames() function, and will call the “request” functions on the EncodedFrameSource object when appropriate.

Comparison with other models

- Chrome stream creation
 - Source can easily be emulated using these functions
 - Explicit sink is not exposed in this proposal.
Needed?
 - Signals go from implicit to explicit - relay is easy
- Apple stream creation
 - No required interaction with workers
 - Can possibly emulate using transferable streams

Maybe: Sender/Receiver redefinition

- Redefine RTPSender and RTPReceiver as box chains
- RTPSender = RTPEncoder + RTPPacketizer + packetizer.insertFrames(encoder)
- RTPReceiver = RTPDepacketizer + RTPDecoder + decoder.insertFrames(packetizer)
- If done right, roles of the components should be clearer, and compatible with current model

WG decisions requested

- Agree that addressing these use cases is within the scope of the WG (more CfCs)
- Agree to accept proposals for APIs that support these use cases' requirements

Discussion (**End Time: 09:10**)

-

Screen Capture Community Group (Elad)

Start Time: 09:10 AM

End Time: 09:15 AM

Screen Capture Community Group

- A new community group has been formed, focusing on screen-capture.
 - <https://www.w3.org/community/sccg/>
- Driven by the needs of Web developers.
- Screen-capture is distinct from WebRTC. Captured content might even stay on the local machine, or might be transmitted remotely using non-WebRTC means.
- We hope to make rapid progress on new APIs and unblock new use-cases:
 - Better integration of capturing and captured applications.
 - [Element Capture](#)
 - Recording all screens
 - Your own topics!
- You are all hereby cordially invited.



Discussion (**End Time: 09:15**)



De-adopting Capture Handle Identity (Elad)

Start Time: 09:15 AM

End Time: 09:25 AM

State of the Art

- Capture Handle Identity allows captured apps to self-declare an identity to a capturer.
- Implemented in Chromium.
- Gainfully employed by multiple Web apps.
- Multiple extensions proposed, but progressing very slowly through standardization.
- Disagreements over vision between current participants in the discussion.
- Insufficient representation of the interests of developers.

Proposal

Stop investing WebRTC WG time and resources in this topic.

Parties that do not share a vision had better work on separate proposals. Allow them to incubate their ideas independently.

These proposals can later be formally presented to a group, which can either choose between the proposals or synthesize them.

Discussion (End Time: 09:25)



Auto-pause Capture (Elad)

Start Time: 09:25 AM

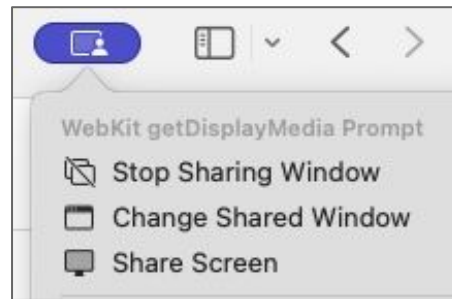
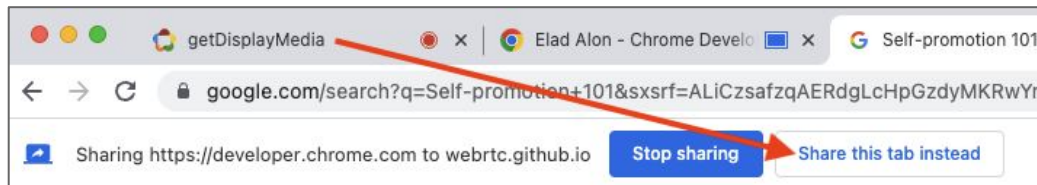
End Time: 09:40 AM



Reminder

A captured surface could change at any moment.

- User navigates a captured tab's top-level document
- Captured app navigates its top-level document (potentially cross-site)
- Dynamic surface switching (deployed by Safari and Chromium)



Problem (Issue #255)

- Apps might wish to adapt themselves to the shared content.
 - Re-prompt the user
 - Adjust cropping parameters
 - Adjust constraints (resolution, frame-rate)
 - Adjust encoding parameters
 - Pipe frames to a new file
- Actions by the app could lag behind production of new frames; possibly even their remote transmission.

Proposal

Add the concept of a paused stream.

- Like muted, it is triggered by something other than the app.
- Like enabled, it can be changed by the app.

(Alternatives reusing muted and/or enabled are possible and may be discussed after we align on the bigger picture.)

Proposal (continued)

```
enum PauseReason {  
    "top-level-navigation",  
    "surface-switch",  
    "config-change" // Foreshadowing  
};  
  
interface PauseEvent : Event {  
    readonly attribute PauseReason reason;  
};
```

```
interface MediaStreamTrack : EventTarget {  
    readonly attribute boolean paused;  
    attribute EventHandler onpause;  
    undefined unpaused();  
};
```

Sample Use

```
track.onpause = (event) => {  
  const track = event.target;  
  if (!AdjustmentNecessary(track) ||  
      Adjust(track)) {  
    track.unpause();  
  }  
};
```

Plugging in the right `AdjustmentNecessary()` and `Adjust()`, an app could tackle any of the previously discussed issues.

Default Behavior

Some options for the default behavior (when there is no explicit event handler set):

- `Pause` (not backwards compatible)
- Default event handler unpauses unconditionally
- `SetPauseHandler()`, to get around the guidance that setting an event handler should not have side effects.

Prior Art

A similar issue was recently tackled as onconfigurationchange.

```
partial interface MediaStreamTrack {  
    attribute EventHandler onconfigurationchange;  
};
```

Possibly one solution can tackle both issues. If we add “config-change” to the PauseReason enum, would that not tickle two birds with one feather?

Discussion (**End Time: 09:40**)



MediaStreamTrack Frame Rates (Henrik)

Start Time: 09:40 AM

End Time: 09:50 AM

PR#77 **MediaStreamTrack.getFrameStats()**

Problem: When FPS is low, we can't determine why...

- Is the camera (or other source) not producing frames?
- Is the user agent dropping frames? Are they dropped or decimated?

Slow progress on PR... balancing act:

- Need to be specific enough that we all agree on what is measured.
- But need to be vague enough to allow different implementations.

Proposal:

- Phrase as requirements, not specific steps.
 - [VideoPlaybackQuality](#) is an example of this.
- Require each frame is *distinctly categorized*. Counter per category.

PR#77 `MediaStreamTrack.getFrameStats()`

If a `MediaStreamTrack` supports `frameRate` as a setting, the user agent is required to categorize each frame originating from its source into one of the following categories:

- A frame is considered **delivered** if it either was delivered to a sink or would have been delivered to a sink, if one was connected.
- A frame is considered **decimated** if it was discarded in order to achieve the target `frameRate`.
- A frame is considered **dropped** if the frame was dropped for any other reason, such as if the system is under heavy load.

WebIDL



```
partial interface MediaStreamTrack {  
  Promise<MediaTrackFrameStats> getFrameStats();  
};
```

WebIDL



```
dictionary MediaTrackFrameStats {  
  DOMHighResTimeStamp timestamp;  
  unsigned long long framesDelivered;  
  unsigned long long framesDecimated;  
  unsigned long long framesDropped;  
};
```

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

-

Wrapup and Next Steps

Start Time: 09:50 AM

End Time: 10:00 AM

Wrap Up and Next Steps

- Step1
- Step2

Name that Butterfly



Thank you

Special thanks to:

WG Participants, Editors & Chairs