# W3C ORCA Community Group Meeting

February 20, 2014

# Welcome!

- Welcome to the first meeting of the W3C ORCA community group!
- During this meeting, we hope to:
  - Introduce you to the community group.
  - Bring you up to date on the status of the ORTC specification.
  - Make progress on outstanding issues.

# About this Virtual Meeting

Information on the meeting

- Hangout URL
- Link to Slides will be published on CG home page & ORTC.org

## CG Chair

Robin Raymond, Chief Architect - Hookflash Inc.

robin@hookflash.com

# W3C CG IPR Policy

- See the Community License Agreement for details.
- Goals are
  - Enable rapid spec development
  - Safe to implement via royalty-free commitments from participants+employers
  - Comfort for committers by limiting scope to OWN contributions
  - Transparency about who is making commitments
- How it works in practice
  - Anyone can post to public-orca
  - CG members who have signed CLA can post to public-orca-contrib
  - Editor should ensure that spec includes only “contributions”, CC-ing public-orca-contrib makes that easier on the editor.

# Introduction:

## W3C ORCA Community Group

- W3C ORCA CG website:
  - http://www.w3.org/community/orca/
- Public mailing list: public-orca@w3.org
  - Join Here - link on the right hand side
  - Non-members can post to this list.
  - Non-member contributions are problematic.
- Contributor's mailing list: public-orca-contrib@w3.org
  - Join Here - link on the right hand side
  - Members only, preferred list for contributions to the specification.

# Associated Sites

- ORTC website: http://ortc.org/
  - Editor’s drafts, pointers to github repos, etc.
- ORTC API Issues List: https://github.com/openpeer/ortc/issues?state=open

# Status of the ORTC API Spec

## 13 February 2014 Editor's draft:

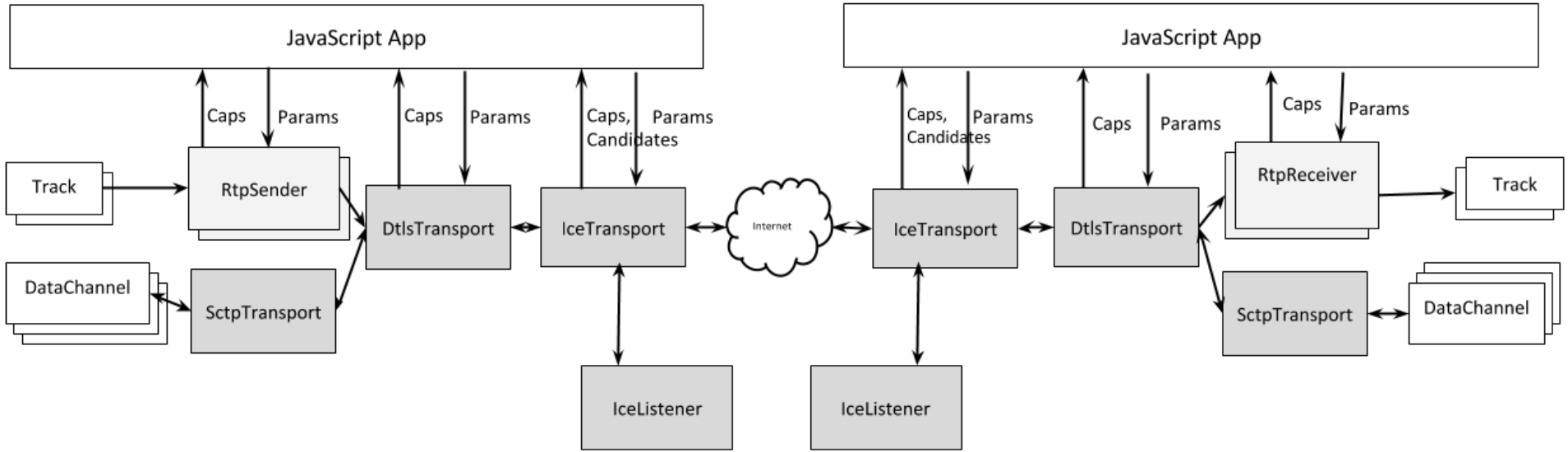
- http://ortc.org/wp-content/uploads/2014/02/ortc.html

## Changes since 07 November 2013:

1. RTCTrack split into RTCRtpSender and RTCRtpReceiver objects, as proposed on 06 January 2014.
2. RTCConnection split into RTCIceTransport and RTCDtlsTransport objects, as proposed on 09 January 2014.
3. RTCSctpTransport object added, as described in Issue 25
4. RTCRtpHeaderExtensionParameters added, as described in Issue 28
5. RTCIceListener added, in order to support parallel forking, as described in Issue 29
6. DTMF support added, as described in Issue 30

Progress rate (and number of changes) is substantial, so there are some loose ends...

# The Big Picture



# Some Loose Ends

- Section 9 (Data Channel) indicates that the Data Channel object is constructed from an RTCDtlsTransport object (Issue # 34).
  - Contradicts the figure in Section 1.
- Examples 1, 7, 11 are blank
- DTLS fingerprints / identity needs to be added
- Any other glaring errors that you noticed?

# Questions for the CG

- Is the CG generally OK with the direction in which the Editor’s draft is headed?
- Do you have questions about the “Big Picture” or other general aspects of the spec?

# Discussions in Progress

- Error handling (Issue #26)
  - Need specific proposals for fixes.
- Active speaker information in mixed (RFC 6465) streams (Issue #27)
  - Current proposal: list of CSRCs with (averaged) levels and (maybe) timestamp when it last appeared.
  - Ongoing discussion: is it accessed via polling, an event handler, or a callback function?
  - Discussion to continue on the list.

# Discussions in Progress (cont'd)

- RTPListener for unsignaled SSRCs (Issue #32)
  - Ongoing discussions relating when the event fires, buffering, role of recv-AppId, etc.
  - Proposal to be refined on the list.
- A more full RTCRtpParameters (Issue #33)
  - Goal: configuration of advanced functionality such as AVPF feedback messages (RFC 4585 & 5107), RTX, FEC, etc.
  - Discussion to continue on the list.

# Issues For Discussion Today

Issue 30: DTMF support

Issue 31: A Big Proposal (leaving for last)

Issue 34: Data Channel (Section 9)

Issue 35: RTCRtpSender/Receiver transport

# Issue 30: DTMF Support

- Editor's draft incorporates Peter Thatcher's proposal to construct an RTCDtmfSender object from an RTCRtpSender object:
  - http://lists.w3.org/Archives/Public/public-orca/2014Feb/0015.html
  - Most of the text in Section 8 copied from WebRTC 1.0 specification.
- Post of Issue 30 to Github generated an (off list) discussion on DTMF support.
  - Approach to DTMF (e.g. SIP INFO vs. RFC 4733)
  - Need for WebRTC 1.0 compatibility
- Is the CG OK with the proposed resolution?

# Issue 34: Data Channel (Section 9)

- What is the problem?
  - Data Channel is constructed from an RTCDtlsTransport object.
  - This contradicts the figure in Section 1 and also bypasses the RTCSctpTransport object (Section 10).
- Proposed fix
  - RTCDataChannel constructed from RTCSctpTransport object
  - ondatachannel on RTCSctpTransport for incoming data channel

# Issue 35: RTCRtpSender/Receiver transport

- What is the problem?
  - RTCRtpSender and RTCRtpReceiver objects are directly constructed from RTCDtlsTransport objects.
- Proposed fix
  - In future (when alternative transports exist) will derive RTCDtlsTransport from base transport interface, RTCRtpSender / RTCRtpReceiver will take base transport as constructor parameter.

# Issue 31: A Big Proposal (Quality+Simulcast)

- Focus on use cases
- This is only the first proposal.
  - Control resolution + framerate with tracks
  - Configure layers/encodings explicitly
  - Configure bitrates explicitly
  - Give quality/priority/bias policy
  - Let more complex things be controlled by JS.
  - If we need to add more, provide more use cases
- Other proposals are welcome

# "Big Proposal" structure

```
dictionary RTCRtpEncodingParameters {

  // Existing codecs, header extensions, etc ...

  // Control of layer definition and dependencies
  bool active;  // send this layer now?
  int layerId;  // numeric layer ID
  sequence<int> layerDependencies;  // Just the IDs
  double scale;  // spatial dimension control

  // For each layer, control what to do as more bits or fewer bits are available.
  // either set quality threshold or bitrate thresholds, not both
  // by default a general-purpose min/maxQuality are used
  double priority;    // how bits are allocated, using weighted fair queueing
  double maxQuality;  // analogous to min QP
  double minQuality;  // analogous to max QP
  double maxBitrate;  // never send more bits than this
  double minBitrate;  // don't send this layer if this many bits aren't available

  DOMString bias; // "resolution" or "framerate"
}
```

# Examples: Single Layer

```
// Normal 1:1 video
var encodings = [{
  ssrc: 1,
}];

// Spend more bits to get higher quality than usual
var encodings = [{
  ssrc: 1, maxQuality: 11.0
}];

// Sign Language - trade off resolution for framerate
var encodings = [{
  ssrc: 1, bias: "framerate"
}];

// Screencast - trade off framerate for resolution
var encodings = [{
  ssrc: 1, bias: "resolution"
}];
```

# Examples: Simulcast, SVC

```
// Simulcast
var encodings =[{
  ssrc: 1, layerId: 0, scale: 0.25,
}, {
  ssrc: 2, layerId: 1, scale: 0.5,
}, {
  ssrc: 3, layerId: 2, scale: 1.0,
}]


// SVC SST; for MST, send each layer from a separate RtpSender
var encodings =[{
  ssrc: 1, layerId: 0, scale: 0.25,
}, {
  ssrc: 1, layerId: 1, scale: 0.5, layerDependencies: [0]
}, {
  ssrc: 1, layerId: 2, scale: 1.0, layerDependencies: [0, 1]
}]
```

# Request to change CG Name

A member of our CG has made a request that we discuss changing the CG name from ORCA to ORTC. The following reasons were given:

- W3C members are actively contributing to ORCA Screenreader project
- ORCAjs.org is also in market now
- ORTC is the specification name

# Thank you


Special thanks to:

Bernard Aboba - Microsoft

Michael Champion - MS Open Tech

Justin Uberti - Google

Peter Thatcher - Google

Martin Thomson - Self

Erik Lagerway - Hookflash