

W3C ORTC Community Group Meeting

April 17, 2014 10:00am-11:30am PDT

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-ortc
 - CG members who have signed CLA can post to public-ortc-contrib
 - Editor should ensure that spec includes only “contributions”, CC-ing public-ortc-contrib makes that easier on the editor.

Welcome!

- Welcome to the 2nd meeting of the W3C [ORTC Community Group](#)!
(renamed from “ORCA Community Group”)
- During this meeting, we hope to:
 - Bring you up to date on the status of the ORTC specification.
 - Discuss ORTC priorities
 - Make progress on outstanding issues.

About this Virtual Meeting

Information on the meeting

- Dial-in Number: 585-627-0587 PIN: 10059
- Link to Slides will be published on CG home page & ORTC.org

CG Chair

Robin Raymond, Chief Architect - Hookflash Inc.

robin@hookflash.com

W3C ORTC Community Group Basics

- W3C ORTC CG website:
 - <http://www.w3.org/community/ortc/>
- Public mailing list: public-ortc@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-ortc-contrib@w3.org
 - Join [Here](#) - link on the right hand side
 - Members only, preferred list for contributions to the specification.

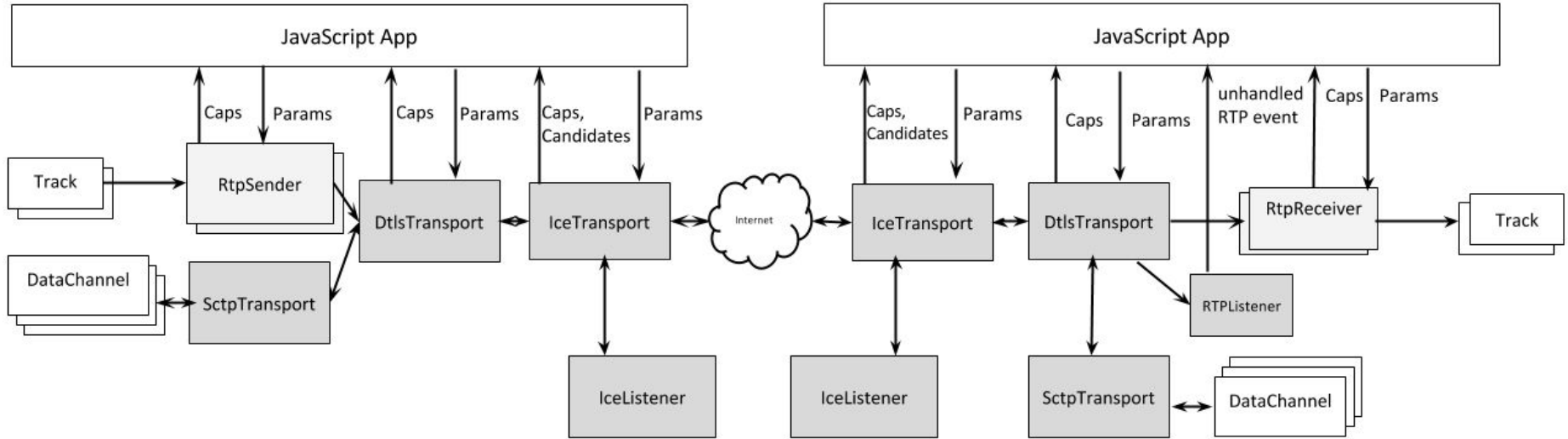
Associated Sites

- 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>

The Way Forward

- ORTC Big Picture
- ORTC Value Proposition
- ORTC Goals
- Assessment of priorities for the future work

The (Revised) Big Picture



ORTC Main Value Proposition

Areas that provide clear value over 1.0:

- Granular / object level control over RTC behaviour without tying to a bigger state machine
- Not tied to a specific blob legacy format (SDP)
- Layering / simulcast with per-layer control

ORTC Goals

- Support RFCs / functionality / capabilities already in WebRTC 1.0
- Basic ORTC 1.0 API to start, improve later as problem space / requirements are understood
- Clear API rule sets and behaviours for clear implementation guidelines
- Cover reasonable set of CG use cases (e.g. mobility, simulcasting, layering)

High Priority Issues

- ICE TCP
- ICE restart
- ICE candidate gather policy
- ICE freezing
- API modeling (eg. factory vs ctor)
- RTP simulcast /layering
- Demux / latching rules for RtcRtpReceiver
- Stats
- Error handling

Nice to Haves

- ICE candidate packaging
- ICE candidate flushing
- ICE mobility
- RTP contributing sources

Out of Scope for ORTC 1

- ICE candidate priority changing
- ICE aggressive changing knob
- ICE warmth

TBD

- ICE Pacing
- Run-time Changing capabilities
- Special case codec parameters

ORTC 1.0 Criteria

- Already supported in WebRTC 1.0
- API usability
- Problem space / use cases well defined
- Needed for compatibility
- Provides clear value proposition for web developers / applications

Questions for the CG

- Do you agree with the stated goals?
- Do you agree with the priorities outlined?

Editor's Draft Changes

12 April 2014 Editor's draft:

- <http://ortc.org/wp-content/uploads/2014/04/ortc.html>

Changes since 13 February 2014 Editor's draft:

- Support for control of quality, resolution, framerate and layering, as described in Issue [31](#).
- More support for RTP and codec parameters, as described in Issue [33](#).
- ICE issues [ICE TCP ([41](#)), acquisition of local candidates ([43](#)), onlocalcandidate definition ([44](#)), gather policy ([47](#))] addressed.
- RTPListener object added, as described in Issue [32](#).
- Initial stab at a Stats API, as requested in Issue [46](#).
- Support for contributing sources added, as requested in Issue [27](#).
- Default values added in some cases, to partially address Issue [39](#).
- Various NITs fixed, as requested in Issues [34](#), [37](#), [38](#).

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 general aspects of the spec?

Coming Attractions

- Broken up "big proposal"
 - layering/simulcast by itself (posted to ortc mail list)
 - <https://github.com/openpeer/ortc/issues/61>
 - quality knobs stuff by itself (posted to ortc mail list)
 - <https://github.com/openpeer/ortc/issues/62>
- Ideas for non-muxed RTCP
- Minor DataChannel cleanup (posted to mail list)
 - <https://github.com/openpeer/ortc/issues/60>

Issues For Discussion Today

- Stats
- ICE TCP
- ICE Gather Policy
- ICE Freezing
- Factory Method Pattern

Stats

- Concept reused from WebRTC 1.0.
- Stats returned are within the context of what each object tracks, no difference otherwise.
- Include the existing stats from <http://www.w3.org/2011/04/webrtc/wiki/Stats>
- Do we need any additional stats? Use cases?
 - For Receiver/Sender:
[draft-singh-xrblock-webrtc-additional-stats](#)
 - Anything for DtlsTransport, IceTransport and SctpTransport?

ICE TCP Proposal (Active / Passive)

- At IETF 89, consensus was to require ICE-TCP support ([RFC 6544](#)).
- Added TCP candidate type:

```
enum RTCIceProtocol {  
    "udp",  
    "tcp"  
};
```

- Offered TCP candidates “passive” or “active”

```
enum RTCIceTcpType {  
    "active",  
    "passive"  
};
```

ICE Gather Policy

- Added from WebRTC 1.0:

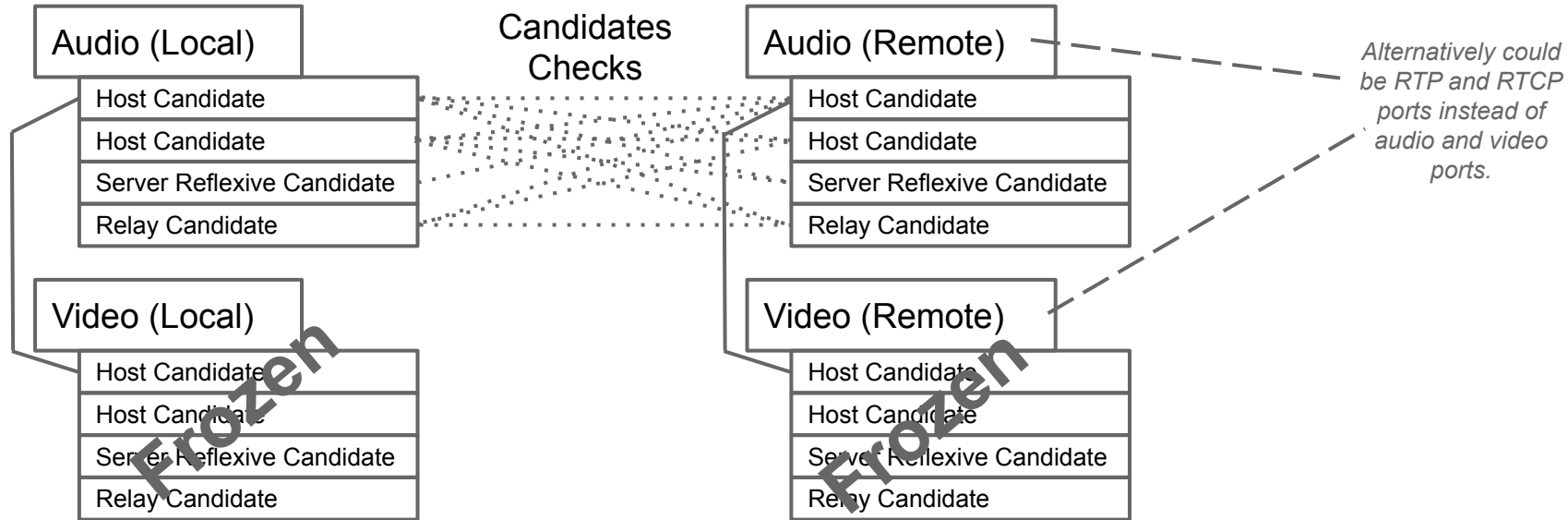
```
enum RTCIceGatherPolicy {  
    "all",  
    "nohost",  
    "relayonly"  
};
```

Does this address the needs of the CG?

Use cases?

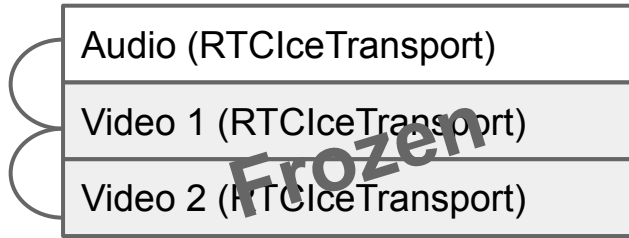
ICE Freezing

- Needed for RTP vs RTCP non-muxed
- Needed for audio / video candidate searches



ICE Freezing Implicit vs Explicit

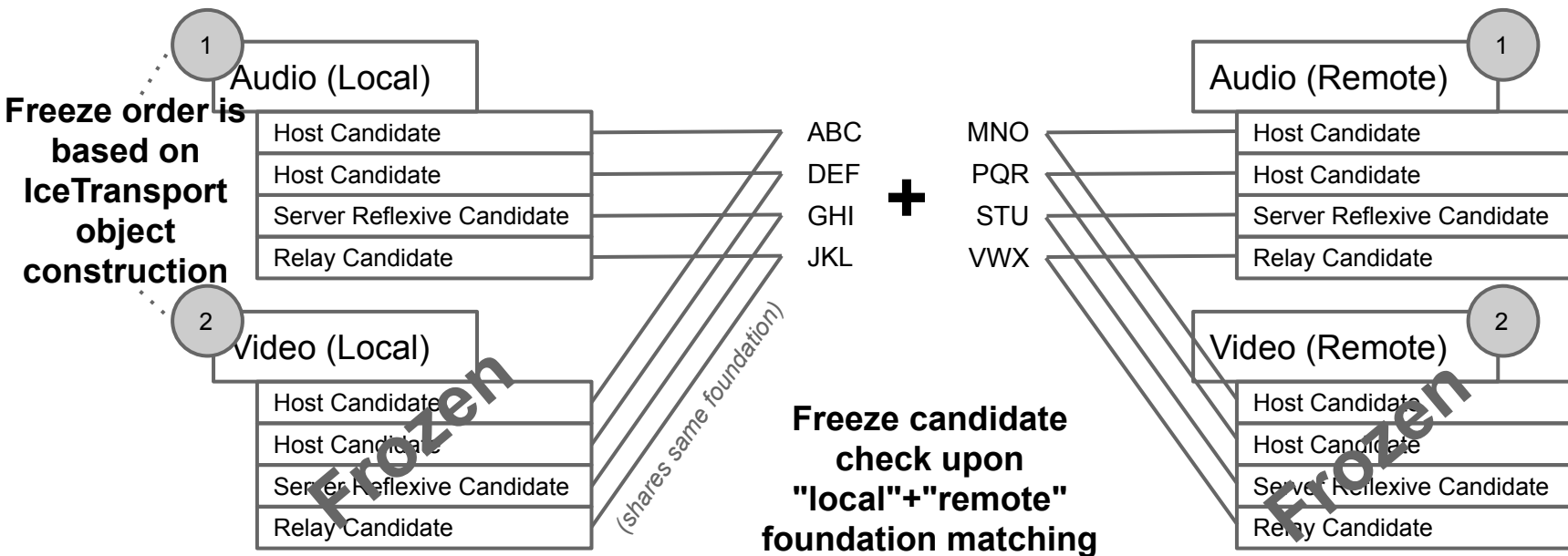
- How is relationship between RTCIceTransport candidates known?



ICE Freezing Implicitly

Each candidate has a unique “foundation” based on:

- type (e.g. host vs server reflexive)
- base IP
- connecting server IP (relay only)



ICE Freezing Explicit

- Relationships expressed in code:

```
function initiate(signaller) {  
  var iceOptions = ...;  
  var iceAudio = new RTCIceTransport(RTCIceRole.controlling, iceOptions);  
  var iceVideo1 = new RTCIceTransport(iceAudio, RTCIceRole.controlling, iceOptions);  
  var iceVideo2 = new RTCIceTransport(iceVideo1, RTCIceRole.controlling, iceOptions);  
}
```

- Can we do implicit relationships?
- Does it cover all needed use cases?
- Do we need explicit RTCIceTransport relationships?

Factory Method Pattern

Pros

- Instantiate abstract interfaces
- Meaningful method signatures:
SomeObject createWithFoo(...)
- Easier to add singletons / static helper methods
- Better encapsulation
- Weak coupling
- Possible to add customization hooks

Cons

- “new” is clearly creating a specific object type
- Consistency with other API(s)?
which? Do we care?
- More methods inside interface vs outside in ctor

Factory Method Pattern Works like ctor

```
[Constructor(
    RTCIceRole role,
    optional RTCIceListener iceListener
),
Constructor(
    RTCIceRole role,
    RTCIceOptions options
)]
interface RTCIceTransport {
    readonly attribute RTCIceRole role;
    readonly attribute RTCIceTransportState state;

    [...]
};
```

```
function initiate(signaller) {
    var iceOptions = ...;
    var ice = new RTCIceTransport(RTCIceRole.controlling,
    iceOptions);
}
```

VS

```
interface RTCIceTransport {
    readonly attribute RTCIceRole role;
    readonly attribute RTCIceTransportState state;

    static RTCIceTransport create(
        RTCIceRole role,
        optional RTCIceListener iceListener
    );

    static RTCIceTransport create(
        RTCIceRole role,
        RTCIceOptions options
    );

    [...]
};
```

```
function initiate(signaller) {
    var iceOptions = ...;
    var ice = RTCIceTransport.create(RTCIceRole.controlling,
    iceOptions);
}
```

Allows For Meaningful Method Name Signatures

```
[Constructor(  
  RTCDATATransport transport,  
  RTCDATAChannelParameters params)]  
interface RTCDATAChannel : EventTarget {  
  readonly attribute RTCDATATransport transport;  
  readonly attribute RTCDATAChannelParameters parameters;  
  
  //...  
};
```

```
interface RTCDATAChannel : EventTarget {  
  readonly attribute RTCDATATransport transport;  
  readonly attribute RTCDATAChannelParameters parameters;  
  
  static RTCDATAChannel open(  
    RTCDATATransport transport,  
    RTCDATAChannelParameters params  
  );  
  
  //...  
};
```

VS

```
function initiate(transport) {  
  var params = ...;  
  var channel = new RTCDATAChannel(transport, params);  
}
```

```
function initiate(transport) {  
  var params = ...;  
  var channel = RTCDATAChannel.open(transport, params);  
}
```

Instantiate Default Derived Interface Types

```
interface RTCDATATransport {  
  
}  
  
[Constructor(RTCDtlsTransport transport)]  
interface RTCSctpTransport : RTCDATATransport {  
  
  // ...  
}
```

VS

```
interface RTCDATATransport {  
  
  static RTCDATATransport create(RTCDtlsTransport transport);  
  
}  
  
interface RTCSctpTransport : RTCDATATransport {  
  
  // ...  
}
```

```
function initiate(transport) {  
  var channel = new RTCSctpTransport(transport);  
}
```

```
function initiate(transport) {  
  var channel = RTCDATATransport.create(transport);  
}
```

Thank you

Special thanks to:

Bernard Aboba - Microsoft

Michael Champion - MS Open Tech

Justin Uberti - Google

Peter Thatcher - Google

Martin Thomson - Self

Robin Raymond - Hookflash

Erik Lagerway - Hookflash

For More Information

ORTC Community Group

<http://www.w3.org/community/ortc/>

ORTC website

<http://ortc.org>