Stream:	Internet Engineering Task Force (IETF)
RFC:	0000
Category:	Standards Track
Published:	December 2019
ISSN:	2070-1721
Author:	H. T. Alvestrand
	Google

RFC 0000
Overview: Real-Time Protocols for Browser-Based Applications

Abstract

This document gives an overview and context of a protocol suite intended for use with real-time applications that can be deployed in browsers -- "real-time communication on the Web".

It intends to serve as a starting and coordination point to make sure that (1) all the parts that are needed to achieve this goal are findable and (2) the parts that belong in the Internet protocol suite are fully specified and on the right publication track.

This document is an applicability statement -- it does not itself specify any protocol, but it specifies which other specifications WebRTC-compliant implementations are supposed to follow.

This document is a work item of the RTCWEB Working Group.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc0000.

Copyright Notice

Table of Contents

1. Introduction

The Internet was, from very early in its lifetime, considered a possible vehicle for the deployment of real-time, interactive applications -- with the most easily imaginable being audio conversations (aka "Internet telephony") and video conferencing.

The first attempts to build this were dependent on special networks, special hardware, and custom-built software, often at very high prices or of low quality, placing great demands on the infrastructure.

As the available bandwidth has increased, and as processors and other hardware have become ever faster, the barriers to participation have decreased, and it has become possible to deliver a satisfactory experience on commonly available computing hardware.

Still, there are a number of barriers to the ability to communicate universally -- one of these is that there is, as of yet, no single set of communication protocols that all agree should be made available for communication; another is the sheer lack of universal identification systems (such as is served by telephone numbers or email addresses in other communications systems).

Development of The Universal Solution has, however, proved hard.

The last few years have also seen a new platform rise for deployment of services: the browser-embedded application, or "Web application". It turns out that as long as the browser platform has the necessary interfaces, it is possible to deliver almost any kind of service on it.

Traditionally, these interfaces have been delivered by plugins, which had to be downloaded and installed separately from the browser; in the development of HTML5, application developers see much promise in the possibility of making those interfaces available in a standardized way within the browser.

This memo describes a set of building blocks that (1) can be made accessible and controllable through a JavaScript API in a browser and (2) together form a sufficient set of functions to allow the use of interactive audio and video in applications that communicate directly between browsers across the Internet. The resulting protocol suite is intended to enable all the applications that are described as required scenarios in the use cases document [RFC7478].

Other efforts -- for instance, the W3C Web Real-Time Communications, Web Applications Security, and Device and Sensor Working Groups -- focus on making standardized APIs and interfaces available, within or alongside the HTML5 effort, for those functions. This memo concentrates on specifying the protocols and subprotocols that are needed to specify the interactions over the network.

Operators should note that deployment of WebRTC will result in a change in the nature of signaling for real-time media on the network and may result in a shift in the kinds of devices used to create and consume such media. In the case of signaling, WebRTC session setup will typically occur over TLS-secured web technologies using application-specific protocols. Operational techniques that involve inserting network elements to interpret the Session Description Protocol (SDP) -- through either endpoint cooperation [RFC3361] or the transparent insertion of SIP Application Level Gateways (ALGs) -- will not work with such signaling. In the case of networks using cooperative endpoints, the approaches defined in [RFC8155] may serve as a suitable replacement for [RFC3361]. The increase in browser-based communications may also lead to a shift away from dedicated real-time-communications hardware, such as SIP desk phones. This will diminish the efficacy of operational techniques that place dedicated real-time

devices on their own network segment, address range, or VLAN for purposes such as applying traffic filtering and QoS. Applying the markings described in [RFCFFFF] may be appropriate replacements for such techniques.

While this document formally relies on [RFC8445], at the time of its publication, the majority of WebRTC implementations support the version of Interactive Connectivity Establishment (ICE) that is described in [RFC5245] and use a pre-standard version of the Trickle ICE mechanism described in [RFCGGGG]. The use of the "ice2" attribute defined in [RFC8445] can be used to detect the version in use by a remote endpoint and to provide a smooth transition from the older specification to the newer one.

This memo uses the term "WebRTC" (note the case used) to refer to the overall effort consisting of both IETF and W3C efforts.

2. Principles and Terminology

2.1. Goals of This Document

The goal of the WebRTC protocol specification is to specify a set of protocols that, if all are implemented, will allow an implementation to communicate with another implementation using audio, video, and data sent along the most direct possible path between the participants.

This document is intended to serve as the roadmap to the WebRTC specifications. It defines terms used by other parts of the WebRTC protocol specifications, lists references to other specifications that don't need further elaboration in the WebRTC context, and gives pointers to other documents that form part of the WebRTC suite.

By reading this document and the documents it refers to, it should be possible to have all information needed to implement a WebRTC-compatible implementation.

2.2. Relationship between API and Protocol

The total WebRTC effort consists of two major parts, each consisting of multiple documents:

- A protocol specification, done in the IETF
- A JavaScript API specification, defined in a series of W3C documents [W3C.WD-webrtc-20120209] [W3C.WD-mediacapture-streams-20120628]

Together, these two specifications aim to provide an environment where JavaScript embedded in any page, when suitably authorized by its user, is able to set up communication using audio, video, and auxiliary data, as long as the browser supports this specification. The browser environment does not constrain the types of application in which this functionality can be used.

The protocol specification does not assume that all implementations implement this API; it is not intended to be necessary for interoperation to know whether the entity one is communicating with is a browser or another device implementing this specification.

The goal of cooperation between the protocol specification and the API specification is that for all options and features of the protocol specification, it should be clear which API calls to make to exercise that option or feature; similarly, for any sequence of API calls, it should be clear which protocol options and features will be invoked. Both are subject to constraints of the implementation, of course.

The following terms are used across the documents specifying the WebRTC suite, in the specific meanings given here. Not all terms are used in this document. Other terms are used in their commonly used meaning.

Agent: Undefined term. See "SDP Agent" and "ICE Agent".

Application Programming Interface (API): A specification of a set of calls and events, usually tied to a programming language or an abstract formal specification such as WebIDL, with its defined semantics.

Browser: Used synonymously with "Interactive User Agent" as defined in the HTML specification [W3C.WD-html5-20110525]. See also "WebRTC User Agent".

Data Channel: An abstraction that allows data to be sent between WebRTC endpoints in the form of messages. Two endpoints can have multiple data channels between them.

ICE Agent: An implementation of the Interactive Connectivity Establishment (ICE) protocol [RFC8445]. An ICE Agent may also be an SDP Agent, but there exist ICE Agents that do not use SDP (for instance, those that use Jingle [XEP-0166]).

Interactive: Communication between multiple parties, where the expectation is that an action from one party can cause a reaction by another party, and the reaction can be observed by

the first party, where the total time required for the action/reaction/observation is on the order of no more than hundreds of milliseconds.

Media: Audio and video content. Not to be confused with "transmission media" such as wires.

Media Path: The path that media data follows from one WebRTC endpoint to another.

Protocol: A specification of a set of data units, their representation, and rules for their transmission, with their defined semantics. A protocol is usually thought of as going between systems.

Real-Time Media: Media where generation of content and display of content are intended to occur closely together in time (on the order of no more than hundreds of milliseconds). Real-time media can be used to support interactive communication.

SDP Agent: The protocol implementation involved in the Session Description Protocol (SDP) offer/answer exchange, as defined in [RFC3264], Section 3.

Signaling: Communication that happens in order to establish, manage, and control media paths and data paths.

Signaling Path: The communication channels used between entities participating in signaling to transfer signaling. There may be more entities in the signaling path than in the media path.

WebRTC Browser (also called a WebRTC User Agent or WebRTC UA): Something that conforms to both the protocol specification and the JavaScript API cited above.

WebRTC Non-Browser: Something that conforms to the protocol specification but does not claim to implement the JavaScript API. This can also be called a "WebRTC device" or "WebRTC native application".

WebRTC Endpoint: Either a WebRTC browser or a WebRTC non-browser. It conforms to the protocol specification.

WebRTC-Compatible Endpoint: An endpoint that is able to successfully communicate with a WebRTC endpoint, but may fail to meet some requirements of a WebRTC endpoint. This may limit where in the network such an endpoint can be attached, or may limit the security guarantees that it offers to others. It is not constrained by this specification; when it is mentioned at all, it is to note the implications on WebRTC-compatible endpoints of the requirements placed on WebRTC endpoints.

WebRTC Gateway: A WebRTC-compatible endpoint that mediates media traffic to non-WebRTC entities.

All WebRTC browsers are WebRTC endpoints, so any requirement on a WebRTC endpoint also applies to a WebRTC browser.

A WebRTC non-browser may be capable of hosting applications in a similar way to the way in which a browser can host JavaScript applications, typically by offering APIs in other languages. For instance it may be implemented as a library that offers a C++ API intended to be loaded into

applications. In this case, similar security considerations as for JavaScript may be needed; however, since such APIs are not defined or referenced here, this document cannot give any specific rules for those interfaces.

WebRTC gateways are described in a separate document, [WebRTC-Gateways].

2.3. On Interoperability and Innovation

The "Mission statement of the IETF" [RFC3935] states that "The benefit of a standard to the Internet is in interoperability - that multiple products implementing a standard are able to work together in order to deliver valuable functions to the Internet's users."

Communication on the Internet frequently occurs in two phases:

- Two parties communicate, through some mechanism, what functionality they both are able to support
- They use that shared communicative functionality to communicate, or, failing to find anything in common, give up on communication.

There are often many choices that can be made for communicative functionality; the history of the Internet is rife with the proposal, standardization, implementation, and success or failure of many types of options, in all sorts of protocols.

The goal of having a mandatory to implement function set is to prevent negotiation failure, not to preempt or prevent negotiation.

The presence of a mandatory to implement function set serves as a strong changer of the marketplace of deployment -- in that it gives a guarantee that, as long as you conform to a specification, and the other party is willing to accept communication at the base level of that specification, you can communicate successfully.

The alternative, that is having no mandatory to implement, does not mean that you cannot communicate, it merely means that in order to be part of the communications partnership, you have to implement the standard "and then some". The "and then some" is usually called a profile of some sort; in the version most antithetical to the Internet ethos, that "and then some" consists of having to use a specific vendor's product only.

2.4. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Architecture and Functionality Groups

For browser-based applications, the model for real-time support does not assume that the browser will contain all the functions needed for an application such as a telephone or a video conference. The vision is that the browser will have the functions needed for a Web application, working in conjunction with its backend servers, to implement these functions.

This means that two vital interfaces need specification: the protocols that browsers use to talk to each other, without any intervening servers, and the APIs that are offered for a JavaScript application to take advantage of the browser's functionality.

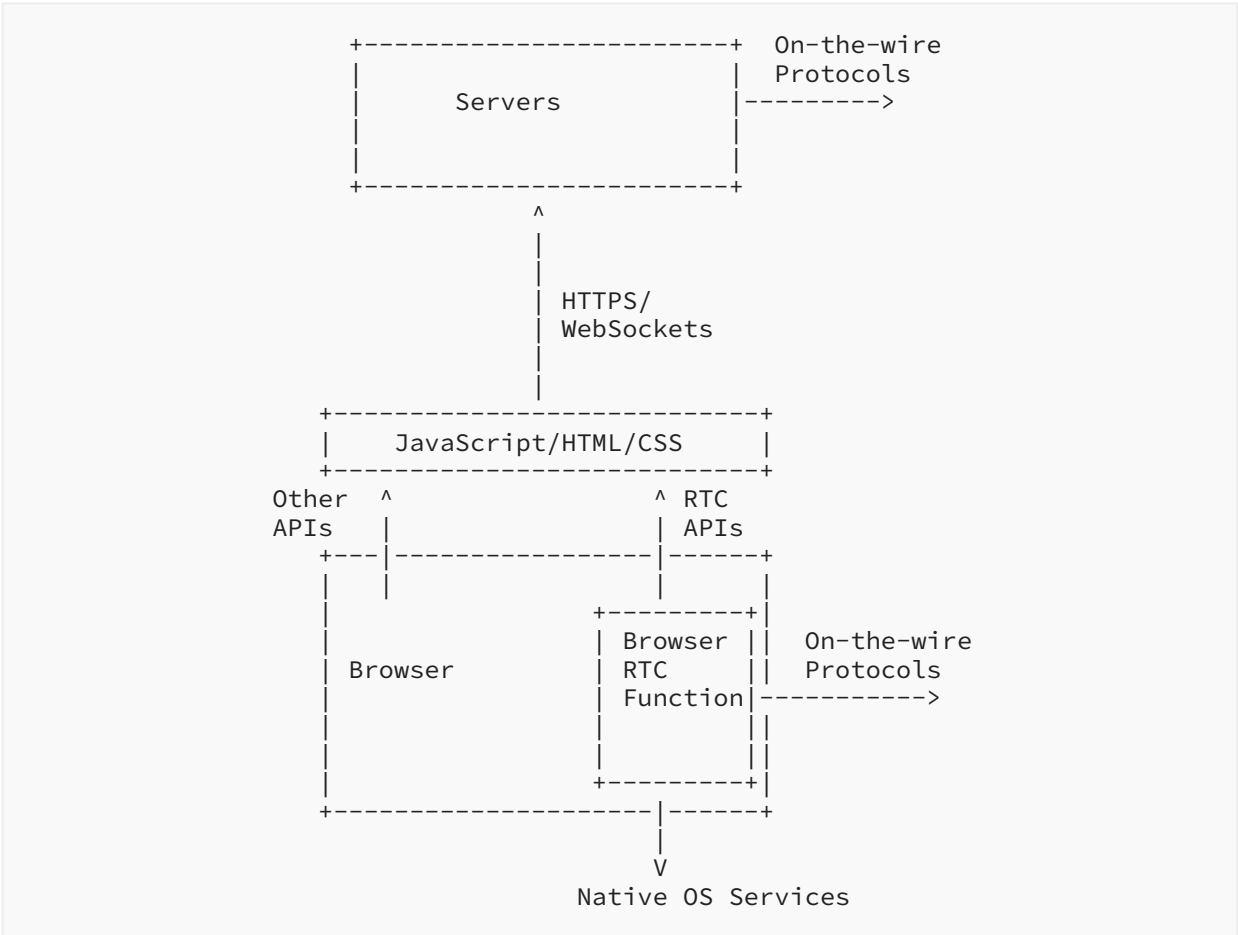

Figure 1: Browser Model

Note that HTTPS and WebSockets are also offered to the JavaScript application through browser APIs.

As for all protocol and API specifications, there is no restriction that the protocols can only be used to talk to another browser; since they are fully specified, any endpoint that implements the protocols faithfully should be able to interoperate with the application running in the browser.

A commonly imagined model of deployment is the one depicted below. In the figure below, "JS" stands for JavaScript.

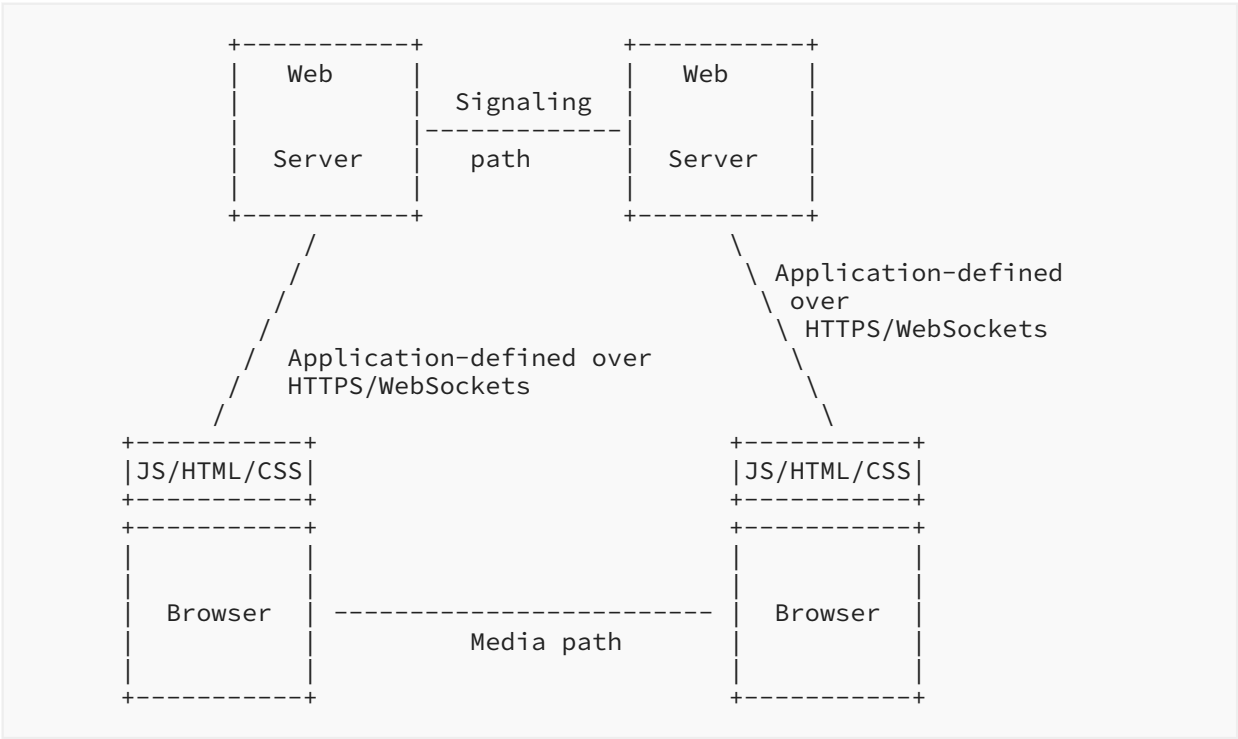

Figure 2: Browser RTC Trapezoid

On this drawing, the critical part to note is that the media path ("low path") goes directly between the browsers, so it has to be conformant to the specifications of the WebRTC protocol suite; the signaling path ("high path") goes via servers that can modify, translate or manipulate the signals as needed.

If the two web servers are operated by different entities, the inter-server signaling mechanism needs to be agreed upon, by either standardization or other means of agreement. Existing protocols (e.g., SIP [RFC3261] or the Extensible Messaging and Presence Protocol (XMPP) [RFC6120]) could be used between servers, while either a standards-based or proprietary protocol could be used between the browser and the web server.

For example, if both operators' servers implement SIP, SIP could be used for communication between servers, along with either a standardized signaling mechanism (e.g., SIP over WebSockets) or a proprietary signaling mechanism used between the application running in the browser and the web server. Similarly, if both operators' servers implement Extensible Messaging and Presence Protocol (XMPP), XMPP could be used for communication between XMPP servers, with either a standardized signaling mechanism (e.g., XMPP over WebSockets or Bidirectional-streams Over Synchronous HTTP (BOSH) [XEP-0124] or a proprietary signaling mechanism used between the application running in the browser and the web server.

The choice of protocols for client-server and inter-server signaling, and definition of the translation between them, is outside the scope of the WebRTC protocol suite described in the document.

The functionality groups that are needed in the browser can be specified, more or less from the bottom up, as:

- Data transport: such as TCP, UDP and the means to securely set up connections between entities, as well as the functions for deciding when to send data: congestion management, bandwidth estimation, and so on.
- Data framing: RTP, the Stream Control Transmission Protocol (SCTP), DTLS, and other data formats that serve as containers, and their functions for data confidentiality and integrity.
- Data formats: Codec specifications, format specifications and functionality specifications for the data passed between systems. Audio and video codecs, as well as formats for data and document sharing, belong in this category. In order to make use of data formats, a way to describe them, a session description, is needed.
- Connection management: Setting up connections, agreeing on data formats, changing data formats during the duration of a call; SDP, SIP, and Jingle/XMPP belong in this category.
- Presentation and control: What needs to happen in order to ensure that interactions behave in a non-surprising manner. This can include floor control, screen layout, voice activated image switching and other such functions -- where part of the system require the cooperation between parties. Centralized Conferencing (XCON) and Cisco/Tandberg's Telepresence Interoperability Protocol (TIP) were some attempts at specifying this kind of functionality; many applications have been built without standardized interfaces to these functions.
- Local system support functions: These are things that need not be specified uniformly, because each participant may choose to do these in a way of the participant's choosing, without affecting the bits on the wire in a way that others have to be cognizant of. Examples in this category include echo cancellation (some forms of it), local authentication and authorization mechanisms, OS access control and the ability to do local recording of conversations.

Within each functionality group, it is important to preserve both freedom to innovate and the ability for global communication. Freedom to innovate is helped by doing the specification in terms of interfaces, not implementation; any implementation able to communicate according to the interfaces is a valid implementation. Ability to communicate globally is helped both by having core specifications be unencumbered by IPR issues and by having the formats and protocols be fully enough specified to allow for independent implementation.

One can think of the three first groups as forming a "media transport infrastructure", and of the three last groups as forming a "media service". In many contexts, it makes sense to use a common specification for the media transport infrastructure, which can be embedded in browsers and accessed using standard interfaces, and "let a thousand flowers bloom" in the "media service" layer; to achieve interoperable services, however, at least the first five of the six groups need to be specified.

4. Data Transport

Data transport refers to the sending and receiving of data over the network interfaces, the choice of network-layer addresses at each end of the communication, and the interaction with any intermediate entities that handle the data but do not modify it (such as TURN relays).

It includes necessary functions for congestion control, retransmission, and in-order delivery.

WebRTC endpoints **MUST** implement the transport protocols described in [RFCEEEE].

5. Data Framing and Securing

The format for media transport is RTP [RFC3550]. Implementation of the Secure Real-time Transport Protocol (SRTP) [RFC3711] is **REQUIRED** for all implementations.

The detailed considerations for usage of functions from RTP and SRTP are given in [RFCDDDD]. The security considerations for the WebRTC use case are in [RFCYYYY], and the resulting security functions are described in [RFCZZZZ].

Considerations for the transfer of data that is not in RTP format is described in [RFCBBBB], and a supporting protocol for establishing individual data channels is described in [RFCCCCC]. WebRTC endpoints **MUST** implement these two specifications.

WebRTC endpoints **MUST** implement [RFCDDDD], [RFCYYYY], [RFCZZZZ], and the requirements they include.

6. Data Formats

The intent of this specification is to allow each communications event to use the data formats that are best suited for that particular instance, where a format is supported by both sides of the connection. However, a minimum standard is greatly helpful in order to ensure that communication can be achieved. This document specifies a minimum baseline that will be supported by all implementations of this specification, and leaves further codecs to be included at the will of the implementer.

WebRTC endpoints that support audio and/or video **MUST** implement the codecs and profiles required in [RFC7874] and [RFC7742].

7. Connection Management

The methods, mechanisms and requirements for setting up, negotiating and tearing down connections is a large subject, and one where it is desirable to have both interoperability and freedom to innovate.

The following principles apply:

1. The WebRTC media negotiations will be capable of representing the same SDP offer/answer semantics [RFC3264] that are used in SIP, in such a way that it is possible to build a signaling gateway between SIP and the WebRTC media negotiation.
2. It will be possible to gateway between legacy SIP devices that support ICE and appropriate RTP / SDP mechanisms, codecs and security mechanisms without using a media gateway. A signaling gateway to convert between the signaling on the web side to the SIP signaling may be needed.
3. When an SDP for a new codec is specified, no other standardization should be required for it to be possible to use that in the web browsers. Adding new codecs that might have new SDP parameters should not change the APIs between the browser and JavaScript application. As soon as the browsers support the new codecs, old applications written before the codecs were specified should automatically be able to use the new codecs where appropriate with no changes to the JavaScript applications.

The particular choices made for WebRTC, and their implications for the API offered by a browser implementing WebRTC, are described in [RFCAAAA].

WebRTC browsers **MUST** implement [RFCAAAA].

WebRTC endpoints **MUST** implement the functions described in that document that relate to the network layer (e.g., Bundle [RFCHHHH], RTCP-mux [RFC5761] ("RTCP" stands for "RTP Control Protocol") and Trickle ICE [RFCGGGG]) but do not need to support the API functionality described there.

8. Presentation and Control

The most important part of control is the user's control over the browser's interaction with input/output devices and communications channels. It is important that the user have some way of figuring out where his audio, video, or texting is being sent, for what purported reason, and what guarantees are made by the parties that form part of this control channel. This is largely a local function between the browser, the underlying operating system and the user interface; this is specified in the peer connection API [W3C.WD-webrtc-20120209], and the media capture API [W3C.WD-mediacapture-streams-20120628].

WebRTC browsers **MUST** implement these two specifications.

9. Local System Support Functions

These are characterized by the fact that the quality of these functions strongly influence the user experience, but the exact algorithm does not need coordination. In some cases (for instance echo cancellation, as described below), the overall system definition may need to specify that the overall system needs to have some characteristics for which these facilities are useful, without requiring them to be implemented a certain way.

Local functions include echo cancellation; volume control; camera management, including focus, zoom, pan/tilt controls (if available); and more.

One would want to see certain parts of the system conform to certain properties, for instance:

- Echo cancellation should be good enough to achieve the suppression of acoustical feedback loops below a perceptually noticeable level.
- Privacy concerns **MUST** be satisfied; for instance, if remote control of camera is offered, the APIs should be available to let the local participant figure out who's controlling the camera, and possibly decide to revoke the permission for camera usage.
- Automatic Gain Control (AGC), if present, should normalize a speaking voice into a reasonable dB range.

The requirements on WebRTC systems with regard to audio processing are found in [RFC7874] and includes more guidance about echo cancellation and AGC; the proposed API for control of local devices are found in [W3C.WD-mediacapture-streams-20120628].

WebRTC endpoints **MUST** implement the processing functions in [RFC7874]. (Together with the requirement in Section 6, this means that WebRTC endpoints **MUST** implement the whole document.)

10. IANA Considerations

This document has no IANA actions.

11. Security Considerations

Security of the web-enabled real-time communications comes in several pieces:

- Security of the components: The browsers, and other servers involved. The most target-rich environment here is probably the browser; the aim here should be that the introduction of these components introduces no additional vulnerability.
- Security of the communication channels: It should be easy for a participant to reassure himself of the security of his communication -- by verifying the crypto parameters of the links he himself participates in, and to get reassurances from the other parties to the communication that they promise that appropriate measures are taken.
- Security of the partners' identity: verifying that the participants are who they say they are (when positive identification is appropriate), or that their identity cannot be uncovered (when anonymity is a goal of the application).

The security analysis, and the requirements derived from that analysis, is contained in [RFCYYYY].

It is also important to read the security sections of [W3C.WD-mediacapture-streams-20120628] and [W3C.WD-webrtc-20120209].

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC7742] Roach, A.B., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <https://www.rfc-editor.org/info/rfc7742>.

[RFC7874] Valin, JM. and C. Bran, "WebRTC Audio Codec and Processing Requirements", RFC 7874, DOI 10.17487/RFC7874, May 2016, <https://www.rfc-editor.org/info/rfc7874>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFCAAAA] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol", RFC AAAA, DOI 10.17487/RFCAAAA, December 2019, <https://www.rfc-editor.org/info/rfcAAAA>.

[RFCBBBB] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", RFC BBBB, DOI 10.17487/RFCBBBB, December 2019, <https://www.rfc-editor.org/info/rfcBBBB>.

[RFCCCCC] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", RFC CCCC, DOI 10.17487/RFCCCCC, December 2019, <https://www.rfc-editor.org/info/rfcCCCC>.

[RFCDDDD] Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", RFC DDDD, DOI 10.17487/RFCDDDD, December 2019, <https://www.rfc-editor.org/info/rfcDDDD>.

[RFCEEEE] Alvestrand, H., "Transports for WebRTC", RFC EEEE, DOI 10.17487/RFCEEEE, December 2019, <https://www.rfc-editor.org/info/rfcEEEE>.

[RFCYYYY] Rescorla, E., "Security Considerations for WebRTC", RFC YYYY, DOI 10.17487/RFCYYYY, December 2019, <https://www.rfc-editor.org/info/rfcYYYY>.

[RFCZZZZ] Rescorla, E., "WebRTC Security Architecture", RFC ZZZZ, DOI 10.17487/RFCZZZZ, December 2019, <https://www.rfc-editor.org/info/rfcZZZZ>.

[W3C.WD-mediacapture-streams-20120628] Burnett, D. and A. Narayanan, "Media Capture and Streams", World Wide Web Consortium WD WD-mediacapture-streams-20120628, June 2012, <https://www.w3.org/TR/2012/WD-mediacapture-streams-20120628>.

[W3C.WD-webrtc-20120209] Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120209, February 2012, <https://www.w3.org/TR/2012/WD-webrtc-20120209>.

12.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers", RFC 3361, DOI 10.17487/RFC3361, August 2002, <https://www.rfc-editor.org/info/rfc3361>.

[RFC3935] Alvestrand, H., "A Mission Statement for the IETF", BCP 95, RFC 3935, DOI 10.17487/RFC3935, October 2004, <https://www.rfc-editor.org/info/rfc3935>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <https://www.rfc-editor.org/info/rfc5245>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <https://www.rfc-editor.org/info/rfc5761>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <https://www.rfc-editor.org/info/rfc7478>.

[RFC8155] Patil, P., Reddy, T., and D. Wing, "Traversal Using Relays around NAT (TURN) Server Auto Discovery", RFC 8155, DOI 10.17487/RFC8155, April 2017, <https://www.rfc-editor.org/info/rfc8155>.

[RFCFFFF] Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "DSCP Packet Markings for WebRTC QoS", RFC FFFF, DOI 10.17487/RFCFFFF, December 2019, <https://www.rfc-editor.org/info/rfcFFFF>.

[RFCGGGG] Ivov, E., Rescorla, E., Uberti, J., and P. Saint-Andre, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", RFC GGGG, DOI 10.17487/RFCGGGG, December 2019, <https://www.rfc-editor.org/info/rfcGGGG>.

[RFCHHHH] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC HHHH, DOI 10.17487/RFCHHHH, December 2019, <https://www.rfc-editor.org/info/rfcHHHH>.

[W3C.WD-html5-20110525] Hickson, I., "HTML5", World Wide Web Consortium Last Call WD-html5-20110525, Work in Progress, May 2011, <https://www.w3.org/TR/2011/WD-html5-20110525>.

[WebRTC-Gateways] Alvestrand, H. and U. Rauschenbach, "WebRTC Gateways", Work in Progress, Internet-Draft, draft-ietf-rtcweb-gateways-02, 21 January 2016, <https://tools.ietf.org/html/draft-ietf-rtcweb-gateways-02>.

[XEP-0124] Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J., Stout, L., and W. Tilanus, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, November 2016, <https://xmpp.org/extensions/xep-0124.html>.

[XEP-0166] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, June 2007, <https://xmpp.org/extensions/xep-0166.html>.

Acknowledgements

The number of people who have taken part in the discussions surrounding this document are too numerous to list, or even to identify. The ones below have made special, identifiable contributions; this does not mean that others' contributions are less important.

Thanks to Cary Bran, Cullen Jennings, Colin Perkins, Magnus Westerlund, and Joerg Ott, who offered technical contributions on various versions of the document.

Thanks to Jonathan Rosenberg, Matthew Kaufman, and others at Skype for the ASCII drawings in Section 3.

Thanks to Alissa Cooper, Bjoern Hoehrmann, Colin Perkins, Colton Shields, Eric Rescorla, Heath Matlock, Henry Sinnreich, Justin Uberti, Keith Drage, Magnus Westerlund, Olle E. Johansson, Sean Turner, and Simon Leinen for document review.

Author's Address

Harald T. Alvestrand
Google
Kungsbron 2
SE-11122 Stockholm
Sweden
Email: harald@alvestrand.no