

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2015

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
January 4, 2015

WebRTC Data Channels
draft-ietf-rtcweb-data-channel-13.txt

Abstract

The WebRTC framework specifies protocol support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies the non-media data transport aspects of the WebRTC framework. It provides an architectural overview of how the Stream Control Transmission Protocol (SCTP) is used in the WebRTC context as a generic transport service allowing WEB-browsers to exchange generic data from peer to peer.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 8, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	3
3. Use Cases	3
3.1. Use Cases for Unreliable Data Channels	4
3.2. Use Cases for Reliable Data Channels	4
4. Requirements	4
5. SCTP over DTLS over UDP Considerations	6
6. The Usage of SCTP for Data Channels	8
6.1. SCTP Protocol Considerations	8
6.2. SCTP Association Management	9
6.3. SCTP Streams	9
6.4. Data Channel Definition	10
6.5. Opening a Data Channel	10
6.6. Transferring User Data on a Data Channel	11
6.7. Closing a Data Channel	12
7. Security Considerations	13
8. IANA Considerations	13
9. Acknowledgments	14
10. References	14
10.1. Normative References	14
10.2. Informative References	15
Authors' Addresses	16

1. Introduction

In the WebRTC framework, communication between the parties consists of media (for example audio and video) and non-media data. Media is sent using SRTP, and is not specified further here. Non-media data is handled by using SCTP [RFC4960] encapsulated in DTLS. DTLS 1.0 is defined in [RFC4347] and the present latest version, DTLS 1.2, is defined in [RFC6347].

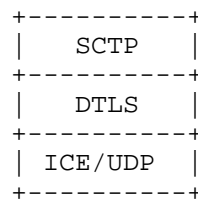


Figure 1: Basic stack diagram

The encapsulation of SCTP over DTLS (see [I-D.ietf-tsvwg-sctp-dtls-encaps]) over ICE/UDP (see [RFC5245]) provides a NAT traversal solution together with confidentiality, source authentication, and integrity protected transfers. This data transport service operates in parallel to the SRTP media transports, and all of them can eventually share a single UDP port number.

SCTP as specified in [RFC4960] with the partial reliability extension defined in [RFC3758] and the additional policies defined in [I-D.ietf-tsvwg-sctp-prpolicies] provides multiple streams natively with reliable, and the relevant partially-reliable delivery modes for user messages. Using the reconfiguration extension defined in [RFC6525] allows to increase the number of streams during the lifetime of an SCTP association and to reset individual SCTP streams. Using [I-D.ietf-tsvwg-sctp-ndata] allows to interleave large messages to avoid the monopolization and adds the support of prioritizing of SCTP streams.

The remainder of this document is organized as follows: Section 3 and Section 4 provide use cases and requirements for both unreliable and reliable peer to peer data channels; Section 5 discusses SCTP over DTLS over UDP; Section 6 provides the specification of how SCTP should be used by the WebRTC protocol framework for transporting non-media data between WEB-browsers.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Use Cases

This section defines use cases specific to data channels. Please note that this section is informational only.

3.1. Use Cases for Unreliable Data Channels

- U-C 1: A real-time game where position and object state information is sent via one or more unreliable data channels. Note that at any time there may be no SRTP media channels, or all SRTP media channels may be inactive, and that there may also be reliable data channels in use.
- U-C 2: Providing non-critical information to a user about the reason for a state update in a video chat or conference, such as mute state.

3.2. Use Cases for Reliable Data Channels

- U-C 3: A real-time game where critical state information needs to be transferred, such as control information. Such a game may have no SRTP media channels, or they may be inactive at any given time, or may only be added due to in-game actions.
- U-C 4: Non-realtime file transfers between people chatting. Note that this may involve a large number of files to transfer sequentially or in parallel, such as when sharing a folder of images or a directory of files.
- U-C 5: Realtime text chat during an audio and/or video call with an individual or with multiple people in a conference.
- U-C 6: Renegotiation of the configuration of the PeerConnection.
- U-C 7: Proxy browsing, where a browser uses data channels of a PeerConnection to send and receive HTTP/HTTPS requests and data, for example to avoid local Internet filtering or monitoring.

4. Requirements

This section lists the requirements for P2P data channels between two browsers. Please note that this section is informational only.

- Req. 1: Multiple simultaneous data channels must be supported. Note that there may be 0 or more SRTP media streams in parallel with the data channels in the same PeerConnection, and the number and state (active/inactive) of these SRTP media streams may change at any time.
- Req. 2: Both reliable and unreliable data channels must be supported.

- Req. 3: Data channels of a PeerConnection must be congestion controlled; either individually, as a class, or in conjunction with the SRTP media streams of the PeerConnection, to ensure that data channels don't cause congestion problems for these SRTP media streams, and that the WebRTC PeerConnection does not cause excessive problems when run in parallel with TCP connections.
- Req. 4: The application should be able to provide guidance as to the relative priority of each data channel relative to each other, and relative to the SRTP media streams. This will interact with the congestion control algorithms.
- Req. 5: Data channels must be secured; allowing for confidentiality, integrity and source authentication. See [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch] for detailed info.
- Req. 6: Data channels must provide message fragmentation support such that IP-layer fragmentation can be avoided no matter how large a message the JavaScript application passes to be sent. It also must ensure that large data channel transfers don't unduly delay traffic on other data channels.
- Req. 7: The data channel transport protocol must not encode local IP addresses inside its protocol fields; doing so reveals potentially private information, and leads to failure if the address is depended upon.
- Req. 8: The data channel transport protocol should support unbounded-length "messages" (i.e., a virtual socket stream) at the application layer, for such things as image-file-transfer; Implementations might enforce a reasonable message size limit.
- Req. 9: The data channel transport protocol should avoid IP fragmentation. It must support PMTU (Path MTU) discovery and must not rely on ICMP or ICMPv6 being generated or being passed back, especially for PMTU discovery.
- Req. 10: It must be possible to implement the protocol stack in the user application space.

5. SCTP over DTLS over UDP Considerations

The important features of SCTP in the WebRTC context are:

- o Usage of a TCP-friendly congestion control.
- o The congestion control is modifiable for integration with the SRTP media stream congestion control.
- o Support of multiple unidirectional streams, each providing its own notion of ordered message delivery.
- o Support of ordered and out-of-order message delivery.
- o Supporting arbitrary large user messages by providing fragmentation and reassembly.
- o Support of PMTU-discovery.
- o Support of reliable or partially reliable message transport.

The WebRTC Data Channel mechanism does not support SCTP multihoming. The SCTP layer will simply act as if it were running on a single-homed host, since that is the abstraction that the DTLS layer (a connection oriented, unreliable datagram service) exposes.

The encapsulation of SCTP over DTLS defined in [I-D.ietf-tsvwg-sctp-dtls-encaps] provides confidentiality, source authenticated, and integrity protected transfers. Using DTLS over UDP in combination with ICE enables middlebox traversal in IPv4 and IPv6 based networks. SCTP as specified in [RFC4960] MUST be used in combination with the extension defined in [RFC3758] and provides the following features for transporting non-media data between browsers:

- o Support of multiple unidirectional streams.
- o Ordered and unordered delivery of user messages.
- o Reliable and partial-reliable transport of user messages.

Each SCTP user message contains a Payload Protocol Identifier (PPID) that is passed to SCTP by its upper layer on the sending side and provided to its upper layer on the receiving side. The PPID can be used to multiplex/demultiplex multiple upper layers over a single SCTP association. In the WebRTC context, the PPID is used to distinguish between UTF-8 encoded user data, binary encoded userdata and the Data Channel Establishment Protocol defined in

[I-D.ietf-rtcweb-data-protocol]. Please note that the PPID is not accessible via the Javascript API.

The encapsulation of SCTP over DTLS, together with the SCTP features listed above satisfies all the requirements listed in Section 4.

The layering of protocols for WebRTC is shown in the following Figure 2.

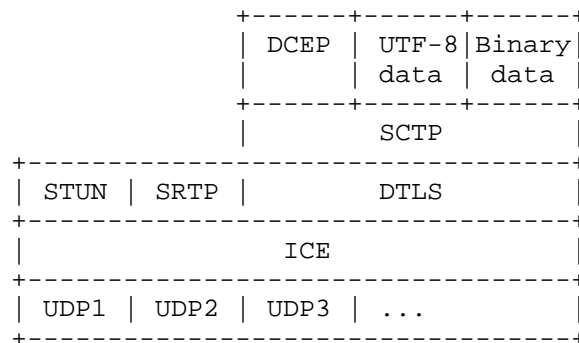


Figure 2: WebRTC protocol layers

This stack (especially in contrast to DTLS over SCTP [RFC6083] in combination with SCTP over UDP [RFC6951]) has been chosen because it

- o supports the transmission of arbitrary large user messages.
- o shares the DTLS connection with the SRTP media channels of the PeerConnection.
- o provides privacy for the SCTP control information.

Considering the protocol stack of Figure 2 the usage of DTLS 1.0 over UDP is specified in [RFC4347] and the usage of DTLS 1.2 over UDP is specified in [RFC6347], while the usage of SCTP on top of DTLS is specified in [I-D.ietf-tsvwg-sctp-dtls-encaps]. Please note that the demultiplexing STUN vs. SRTP vs. DTLS is done as described in Section 5.1.2 of [RFC5764] and SCTP is the only payload of DTLS.

Since DTLS is typically implemented in user application space, the SCTP stack also needs to be a user application space stack.

The ICE/UDP layer can handle IP address changes during a session without needing interaction with the DTLS and SCTP layers. However, SCTP SHOULD be notified when an address changes has happened. In this case SCTP SHOULD retest the Path MTU and reset the congestion

state to the initial state. In case of a window based congestion control like the one specified in [RFC4960], this means setting the congestion window and slow start threshold to its initial values.

Incoming ICMP or ICMPv6 messages can't be processed by the SCTP layer, since there is no way to identify the corresponding association. Therefore SCTP MUST support performing Path MTU discovery without relying on ICMP or ICMPv6 as specified in [RFC4821] using probing messages specified in [RFC4820]. The initial Path MTU at the IP layer SHOULD NOT exceed 1200 bytes for IPv4 and 1280 for IPv6.

In general, the lower layer interface of an SCTP implementation should be adapted to address the differences between IPv4 and IPv6 (being connection-less) or DTLS (being connection-oriented).

When the protocol stack of Figure 2 is used, DTLS protects the complete SCTP packet, so it provides confidentiality, integrity and source authentication of the complete SCTP packet.

SCTP provides congestion control on a per-association base. This means that all SCTP streams within a single SCTP association share the same congestion window. Traffic not being sent over SCTP is not covered by the SCTP congestion control. Using a congestion control different from than the standard one might improve the impact on the parallel SRTP media streams.

SCTP uses the same port number concept as TCP and UDP do. Therefore an SCTP association uses two port numbers, one at each SCTP end-point.

6. The Usage of SCTP for Data Channels

6.1. SCTP Protocol Considerations

The DTLS encapsulation of SCTP packets as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] MUST be used.

This SCTP stack and its upper layer MUST support the usage of multiple SCTP streams. A user message can be sent ordered or unordered and with partial or full reliability.

The following SCTP protocol extensions are required:

- o The stream reconfiguration extension defined in [RFC6525] MUST be supported. It is used for closing channels.

- o The dynamic address reconfiguration extension defined in [RFC5061] MUST be used to signal the support of the stream reset extension defined in [RFC6525]. Other features of [RFC5061] are OPTIONAL.
- o The partial reliability extension defined in [RFC3758] MUST be supported. In addition to the timed reliability PR-SCTP policy defined in [RFC3758], the limited retransmission policy defined in [I-D.ietf-tsvwg-sctp-prpolicies] MUST be supported. Limiting the number of retransmissions to zero combined with unordered delivery provides a UDP-like service where each user message is sent exactly once and delivered in the order received.

The support for message interleaving as defined in [I-D.ietf-tsvwg-sctp-ndata] SHOULD be used.

6.2. SCTP Association Management

In the WebRTC context, the SCTP association will be set up when the two endpoints of the WebRTC PeerConnection agree on opening it, as negotiated by JSEP (typically an exchange of SDP) [I-D.ietf-rtcweb-jsep]. It will use the DTLS connection selected via ICE; typically this will be shared via BUNDLE or equivalent with DTLS connections used to key the SRTP media streams.

The number of streams negotiated during SCTP association setup SHOULD be 65535, which is the maximum number of streams that can be negotiated during the association setup.

SCTP supports two ways of terminating an SCTP association. A graceful one, using a procedure which ensures that no messages are lost during the shutdown of the association. The second method is a non-graceful one, where one side can just abort the association.

Each SCTP end-point supervises continuously the reachability of its peer by monitoring the number of retransmissions of user messages and test messages. In case of excessive retransmissions, the association is terminated in a non-graceful way.

If an SCTP association is closed in a graceful way, all of its data channels are closed. In case of a non-graceful teardown, all data channels are also closed, but an error indication SHOULD be provided if possible.

6.3. SCTP Streams

SCTP defines a stream as a unidirectional logical channel existing within an SCTP association to another SCTP endpoint. The streams are used to provide the notion of in-sequence delivery and for

multiplexing. Each user message is sent on a particular stream, either ordered or unordered. Ordering is preserved only for ordered messages sent on the same stream.

6.4. Data Channel Definition

Data channels are defined such that their accompanying application-level API can closely mirror the API for WebSockets, which implies bidirectional streams of data and a textual field called 'label' used to identify the meaning of the data channel.

The realization of a data channel is a pair of one incoming stream and one outgoing SCTP stream having the same SCTP stream identifier. How these SCTP stream identifiers are selected is protocol and implementation dependent. This allows a bidirectional communication.

Additionally, each data channel has the following properties in each direction:

- o reliable or unreliable message transmission. In case of unreliable transmissions, the same level of unreliability is used. Please note that in SCTP this is a property of an SCTP user message and not of an SCTP stream.
- o in-order or out-of-order message delivery for message sent. Please note that in SCTP this is a property of an SCTP user message and not of an SCTP stream.
- o A priority, which is a 2 byte unsigned integer. These priorities MUST be interpreted as weighted-fair-queuing scheduling priorities per the definition of the corresponding stream scheduler supporting interleaving in [I-D.ietf-tsvwg-sctp-ndata]. For use in WebRTC, the values used SHOULD be one of 128 ("below normal"), 256 ("normal"), 512 ("high") or 1024 ("extra high").
- o an optional label.
- o an optional protocol.

Please note that for a data channel being negotiated with the protocol specified in [I-D.ietf-rtcweb-data-protocol] all of the above properties are the same in both directions.

6.5. Opening a Data Channel

Data channels can be opened by using negotiation within the SCTP association, called in-band negotiation, or out-of-band negotiation. Out-of-band negotiation is defined as any method which results in an

agreement as to the parameters of a channel and the creation thereof. The details are out of scope of this document. Applications using data channels need to use the negotiation methods consistently on both end-points.

A simple protocol for in-band negotiation is specified in [I-D.ietf-rtcweb-data-protocol].

When one side wants to open a channel using out-of-band negotiation, it picks a stream. Unless otherwise defined or negotiated, the streams are picked based on the DTLS role (the client picks even stream identifiers, the server odd stream identifiers). However, the application is responsible for avoiding collisions with existing streams. If it attempts to re-use a stream which is part of an existing data channel, the addition MUST fail. In addition to choosing a stream, the application SHOULD also determine the options to use for sending messages. The application MUST ensure in an application-specific manner that the application at the peer will also know the selected stream to be used, and the options for sending data from that side.

6.6. Transferring User Data on a Data Channel

All data sent on a data channel in both directions MUST be sent over the underlying stream using the reliability defined when the data channel was opened unless the options are changed, or per-message options are specified by a higher level.

The message-orientation of SCTP is used to preserve the message boundaries of user messages. Therefore, senders MUST NOT put more than one application message into an SCTP user message. Unless the deprecated PPID-based fragmentation and reassembly is used, the sender MUST include exactly one application message in each SCTP user message.

The SCTP Payload Protocol Identifiers (PPIDs) are used to signal the interpretation of the "Payload data". The following PPIDs MUST be used (see Section 8):

WebRTC String: to identify a non-empty JavaScript string encoded in UTF-8.

WebRTC String Empty: to identify an empty JavaScript string encoded in UTF-8.

WebRTC Binary: to identify a non-empty JavaScript binary data (ArrayBuffer, ArrayBufferView or Blob).

WebRTC Binary Empty: to identify an empty JavaScript binary data (ArrayBuffer, ArrayBufferView or Blob).

SCTP does not support the sending of empty user messages. Therefore, if an empty message has to be sent, the appropriate PPID (WebRTC String Empty or WebRTC Binary Empty) is used and the SCTP user message of one zero byte is sent. When receiving an SCTP user message with one of these PPIDs, the receiver MUST ignore the SCTP user message and process it as an empty message.

The usage of the PPIDs "WebRTC String Partial" and "WebRTC Binary Partial" is deprecated. They were used for a PPID-based fragmentation and reassembly of user messages belonging to reliable and ordered data channels.

If a message with an unsupported PPID is received or some error condition related to the received message is detected by the receiver (for example, illegal ordering), the receiver SHOULD close the corresponding data channel. This implies in particular that extensions using additional PPIDs can't be used without prior negotiation.

The SCTP base protocol specified in [RFC4960] does not support the interleaving of user messages. Therefore sending a large user message can monopolize the SCTP association. To overcome this limitation, [I-D.ietf-tsvwg-sctp-ndata] defines an extension to support message interleaving, which SHOULD be used. As long as message interleaving is not supported, the sender SHOULD limit the maximum message size to 16 KB to avoid monopolization.

It is recommended that the message size be kept within certain size bounds as applications will not be able to support arbitrarily-large single messages. This limit has to be negotiated, for example by using [I-D.ietf-mmusic-sctp-sdp].

The sender SHOULD disable the Nagle algorithm (see [RFC1122]) to minimize the latency.

6.7. Closing a Data Channel

Closing of a data channel MUST be signaled by resetting the corresponding outgoing streams [RFC6525]. This means that if one side decides to close the data channel, it resets the corresponding outgoing stream. When the peer sees that an incoming stream was reset, it also resets its corresponding outgoing stream. Once this is completed, the data channel is closed. Resetting a stream sets the Stream Sequence Numbers (SSNs) of the stream back to 'zero' with a corresponding notification to the application layer that the reset

has been performed. Streams are available for reuse after a reset has been performed.

[RFC6525] also guarantees that all the messages are delivered (or abandoned) before the stream is reset.

7. Security Considerations

This document does not add any additional considerations to the ones given in [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch].

It should be noted that a receiver must be prepared that the sender tries to send arbitrary large messages.

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

This document uses six already registered SCTP Payload Protocol Identifiers (PPIDs): "DOMString Last", "Binary Data Partial", "Binary Data Last", "DOMString Partial", "WebRTC String Empty", and "WebRTC Binary Empty". [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which these identifiers were assigned. IANA is requested to update the reference of these six assignments to point to this document and change the names of the first four PPIDs. The corresponding dates should be kept.

Therefore these six assignments should be updated to read:

Value	SCTP PPID	Reference	Date
WebRTC String	51	[RFCXXXX]	2013-09-20
WebRTC Binary Partial (Deprecated)	52	[RFCXXXX]	2013-09-20
WebRTC Binary	53	[RFCXXXX]	2013-09-20
WebRTC String Partial (Deprecated)	54	[RFCXXXX]	2013-09-20
WebRTC String Empty	56	[RFCXXXX]	2014-08-22
WebRTC Binary Empty	57	[RFCXXXX]	2014-08-22

9. Acknowledgments

Many thanks for comments, ideas, and text from Harald Alvestrand, Richard Barnes, Adam Bergkvist, Alissa Cooper, Benoit Claise, Spencer Dawkins, Gunnar Hellstrom, Christer Holmberg, Cullen Jennings, Paul Kyzivat, Eric Rescorla, Adam Roach, Irene Ruengeler, Randall Stewart, Martin Stiemerling, Justin Uberti, and Magnus Westerlund.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

- [I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann,
"Stream Schedulers and a New Data Chunk for the Stream
Control Transmission Protocol", draft-ietf-tsvwg-sctp-
ndata-01 (work in progress), July 2014.
- [I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel
Establishment Protocol", draft-ietf-rtcweb-data-
protocol-08 (work in progress), September 2014.
- [I-D.ietf-tsvwg-sctp-dtls-encaps]
Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS
Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-
dtls-encaps-07 (work in progress), December 2014.
- [I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-
ietf-rtcweb-security-07 (work in progress), July 2014.
- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-
rtcweb-security-arch-10 (work in progress), July 2014.
- [I-D.ietf-rtcweb-jsep]
Uberti, J., Jennings, C., and E. Rescorla, "Javascript
Session Establishment Protocol", draft-ietf-rtcweb-jsep-08
(work in progress), October 2014.
- [I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto,
"Additional Policies for the Partial Reliability Extension
of the Stream Control Transmission Protocol", draft-ietf-
tsvwg-sctp-prpolicies-06 (work in progress), December
2014.
- [I-D.ietf-mmusic-sctp-sdp]
Holmberg, C., Loreto, S., and G. Camarillo, "Stream
Control Transmission Protocol (SCTP)-Based Media Transport
in the Session Description Protocol (SDP)", draft-ietf-
mmusic-sctp-sdp-11 (work in progress), December 2014.

10.2. Informative References

- [RFC1122] Braden, R., "Requirements for Internet Hosts -
Communication Layers", STD 3, RFC 1122, October 1989.

- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
- [RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.
- [RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

Authors' Addresses

Randell Jesup
Mozilla
US

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: salvatore.loreto@ericsson.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

Email: tuexen@fh-muenster.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2015

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
January 4, 2015

WebRTC Data Channel Establishment Protocol
draft-ietf-rtcweb-data-protocol-09.txt

Abstract

The WebRTC framework specifies protocol support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies a simple protocol for establishing symmetric Data Channels between the peers. It uses a two way handshake and allows sending of user data without waiting for the handshake to complete.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 8, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions	2
3. Terminology	3
4. Protocol Overview	3
5. Message Formats	4
5.1. DATA_CHANNEL_OPEN Message	4
5.2. DATA_CHANNEL_ACK Message	7
6. Procedures	7
7. Security Considerations	8
8. IANA Considerations	9
8.1. SCTP Payload Protocol Identifier	9
8.2. New Standalone Registry for the DCEP	9
8.2.1. New Message Type Registry	9
8.2.2. New Channel Type Registry	10
9. Acknowledgments	11
10. References	11
10.1. Normative References	11
10.2. Informational References	12
Authors' Addresses	12

1. Introduction

The Data Channel Establishment Protocol (DCEP) is designed to provide, in the WebRTC Data Channel context [I-D.ietf-rtcweb-data-channel], a simple in-band method to open symmetric Data Channels. As discussed in [I-D.ietf-rtcweb-data-channel], the protocol uses the Stream Control Transmission Protocol (SCTP) [RFC4960] encapsulated in the Datagram Transport Layer Security (DTLS) as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] to benefit from their already standardized transport and security features. DTLS 1.0 is defined in [RFC4347] and the present latest version, DTLS 1.2, is defined in [RFC6347].

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Terminology

This document uses the following terms:

Association: An SCTP association.

Stream: A unidirectional stream of an SCTP association. It is uniquely identified by an SCTP stream identifier (0-65534). Note: the SCTP stream identifier 65535 is reserved due to SCTP INIT and INIT-ACK chunks only allowing a maximum of 65535 Streams to be negotiated (0-65534).

Stream Identifier: The SCTP stream identifier uniquely identifying a Stream.

Data Channel: Two Streams with the same Stream Identifier, one in each direction, which are managed together.

4. Protocol Overview

The Data Channel Establishment Protocol is a simple, low-overhead way to establish bidirectional Data Channels over an SCTP association with a consistent set of properties.

The set of consistent properties includes:

- o reliable or unreliable message transmission. In case of unreliable transmissions, the same level of unreliability is used.
- o in-order or out-of-order message delivery.
- o the priority of the Data Channel.
- o an optional label for the Data Channel.
- o an optional protocol for the Data Channel.
- o the Streams.

This protocol uses a two way handshake to open a Data Channel. The handshake pairs one incoming and one outgoing Stream, both having the same Stream Identifier, into a single bidirectional Data Channel. The peer that initiates opening a Data Channel selects a Stream Identifier for which the corresponding incoming and outgoing Streams are unused and sends a DATA_CHANNEL_OPEN message on the outgoing Stream. The peer responds with a DATA_CHANNEL_ACK message on its corresponding outgoing Stream. Then the Data Channel is open. Data Channel Establishment Protocol messages are sent on the same Stream

as the user messages belonging to the Data Channel. The demultiplexing is based on the SCTP payload protocol identifier (PPID), since the Data Channel Establishment Protocol uses a specific PPID.

Note: The opening side MAY send user messages before the DATA_CHANNEL_ACK is received.

To avoid collisions where both sides try to open a Data Channel with the same Stream Identifiers, each side MUST use Streams with either even or odd Stream Identifiers when sending a DATA_CHANNEL_OPEN message. When using SCTP over DTLS [I-D.ietf-tsvwg-sctp-dtls-encaps], the method used to determine which side uses odd or even is based on the underlying DTLS connection role: the side acting as the DTLS client MUST use Streams with even Stream Identifiers, the side acting as the DTLS server MUST use Streams with odd Stream Identifiers.

Note: There is no attempt to ensure uniqueness for the label; if both sides open a Data Channel labeled "x" at the same time, there will be two Data Channels labeled "x" - one on an even Stream pair, one on an odd pair.

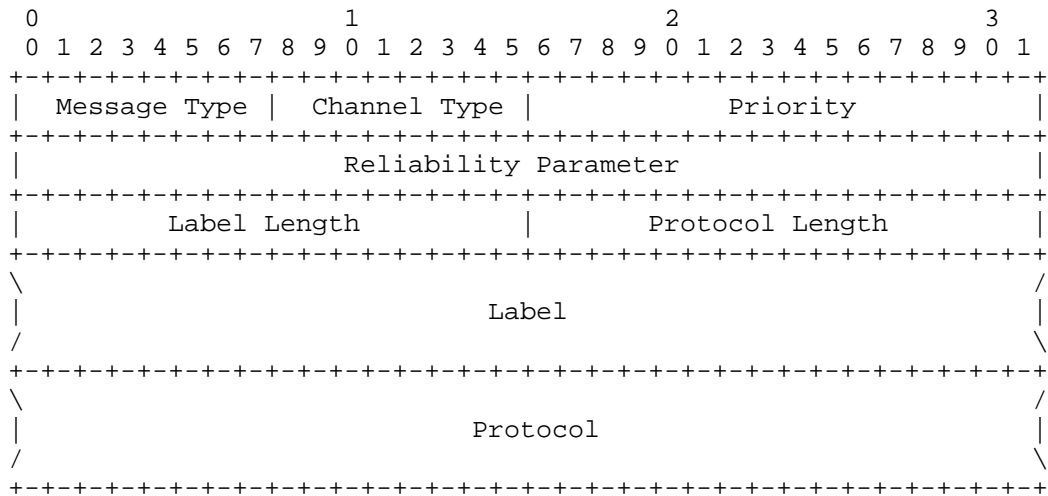
The protocol field is to ease cross-application interoperation ("federation") by identifying the user data being passed with an IANA-registered string ('WebSocket Subprotocol Name Registry' defined in [RFC6455]), and may be useful for homogeneous applications which may create more than one type of Data Channel. Please note that there is also no attempt to ensure uniqueness for the protocol field.

5. Message Formats

Every Data Channel Establishment Protocol message starts with a one byte field called "Message Type" which indicates the type of the message. The corresponding values are managed by IANA (see Section 8.2.1).

5.1. DATA_CHANNEL_OPEN Message

This message is sent initially on the Stream used for user messages using the Data Channel.



Message Type: 1 byte (unsigned integer)

This field holds the IANA defined messagetype for the DATA_CHANNEL_OPEN message. The value of this field is 0x03 as specified in Section 8.2.1.

Channel Type: 1 byte (unsigned integer)

This field specifies the type of the Data Channel to be opened and the values are managed by IANA (see Section 8.2.2):

DATA_CHANNEL_RELIABLE (0x00): The Data Channel provides a reliable in-order bi-directional communication.

DATA_CHANNEL_RELIABLE_UNORDERED (0x80): The Data Channel provides a reliable unordered bi-directional communication.

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT (0x01): The Data Channel provides a partially-reliable in-order bi-directional communication. User messages will not be retransmitted more times than specified in the Reliability Parameter.

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED (0x81): The Data Channel provides a partial reliable unordered bi-directional communication. User messages will not be retransmitted more times than specified in the Reliability Parameter.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED (0x02): The Data Channel provides a partial reliable in-order bi-directional communication. User messages might not be transmitted or retransmitted after a specified life-time given in milli-

seconds in the Reliability Parameter. This life-time starts when providing the user message to the protocol stack.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED (0x82): The Data Channel provides a partial reliable unordered bi-directional communication. User messages might not be transmitted or retransmitted after a specified life-time given in milliseconds in the Reliability Parameter. This life-time starts when providing the user message to the protocol stack.

Priority: 2 bytes (unsigned integer)

The priority of the Data Channel as described in [I-D.ietf-rtcweb-data-channel].

Reliability Parameter: 4 bytes (unsigned integer)

For reliable Data Channels this field MUST be set to 0 on the sending side and MUST be ignored on the receiving side. If a partial reliable Data Channel with limited number of retransmissions is used, this field specifies the number of retransmissions. If a partial reliable Data Channel with limited lifetime is used, this field specifies the maximum lifetime in milliseconds. The following table summarizes this:

Channel Type	Reliability Parameter
DATA_CHANNEL_RELIABLE	Ignored
DATA_CHANNEL_RELIABLE_UNORDERED	Ignored
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT	Number of RTX
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED	Number of RTX
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED	Lifetime in ms
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED	Lifetime in ms

Label Length: 2 bytes (unsigned integer)

The length of the label field in bytes.

Protocol Length: 2 bytes (unsigned integer)

The length of the protocol field in bytes.

Label: Variable Length (sequence of characters)

The name of the Data Channel as a UTF-8 encoded string as specified in [RFC3629]. This may be an empty string.

Protocol: Variable Length (sequence of characters)

If this is an empty string the protocol is unspecified. If it is a non-empty string, it specifies a protocol registered in the

'WebSocket Subprotocol Name Registry' created in [RFC6455]. This string is UTF-8 encoded as specified in [RFC3629].

5.2. DATA_CHANNEL_ACK Message

This message is sent in response to a DATA_CHANNEL_OPEN_RESPONSE message on the stream used for user messages using the Data Channel. Reception of this message tells the opener that the Data Channel setup handshake is complete.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|  Message Type  |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Message Type: 1 byte (unsigned integer)

This field holds the IANA defined message type for the DATA_CHANNEL_ACK message. The value of this field is 0x02 as specified in Section 8.2.1.

6. Procedures

All Data Channel Establishment Protocol messages MUST be sent using ordered delivery and reliable transmission. They MUST be sent on the same outgoing Stream as the user messages belonging to the corresponding Data Channel. Multiplexing and demultiplexing is done by using the SCTP payload protocol identifier (PPID). Therefore Data Channel Establishment Protocol message MUST be sent with the assigned PPID for the Data Channel Establishment Protocol (see Section 8.1). Other messages MUST NOT be sent using this PPID.

The peer that initiates opening a Data Channel selects a Stream Identifier for which the corresponding incoming and outgoing Streams are unused. If the side is the DTLS client, it MUST choose an even Stream Identifier, if the side is the DTLS server, it MUST choose an odd one. It fills in the parameters of the DATA_CHANNEL_OPEN message and sends it on the chosen Stream.

If a DATA_CHANNEL_OPEN message is received on an unused Stream, the Stream Identifier corresponds to the role of the peer and all parameters in the DATA_CHANNEL_OPEN message are valid, then a corresponding DATA_CHANNEL_ACK message is sent on the Stream with the same Stream Identifier as the one the DATA_CHANNEL_OPEN message was received on.

If the DATA_CHANNEL_OPEN message doesn't satisfy the conditions above, for instance if a DATA_CHANNEL_OPEN message is received on an

already used Stream or there are any problems with parameters within the DATA_CHANNEL_OPEN message, the odd/even rule is violated or the DATA_CHANNEL_OPEN message itself is not well-formed, the receiver MUST close the corresponding Data Channel using the procedure described in [I-D.ietf-rtcweb-data-channel] and MUST NOT send a DATA_CHANNEL_ACK message in response to the received message. Therefore, receiving an SCTP stream reset request for a Stream on which no DATA_CHANNEL_ACK message has been received indicates to the sender of the corresponding DATA_CHANNEL_OPEN message the failure of the Data Channel setup procedure. After also successfully resetting the corresponding outgoing Stream, which concludes the Data Channel closing initiated by the peer, a new DATA_CHANNEL_OPEN message can be sent on the Stream.

After the DATA_CHANNEL_OPEN message has been sent, the sender of the DATA_CHANNEL_OPEN MAY start sending messages containing user data without waiting for the reception of the corresponding DATA_CHANNEL_ACK message. However, before the DATA_CHANNEL_ACK message or any other message has been received on a Data Channel, all other messages containing user data and belonging to this Data Channel MUST be sent ordered, no matter whether the Data Channel is ordered or not. After the DATA_CHANNEL_ACK or any other message has been received on the Data Channel, messages containing user data MUST be sent ordered on ordered Data Channels and MUST be sent unordered on unordered Data Channels. Therefore receiving a message containing user data on an unused Stream indicates an error. The corresponding Data Channel MUST be closed as described in [I-D.ietf-rtcweb-data-channel].

7. Security Considerations

The DATA_CHANNEL_OPEN messages contains two variable length fields: the protocol and the label. A receiver must be prepared to receive DATA_CHANNEL_OPEN messages where these field have the maximum length of 65535 bytes. Error cases like the use of inconsistent lengths fields, unknown parameter values or violation the odd/even rule must also be handled by closing the corresponding Data Channel. An endpoint must also be prepared that the peer open the maximum number of Data Channels.

This protocol does not provide privacy, integrity or authentication. It needs to be used as part of a protocol suite that contains all these things. Such a protocol suite is specified in [I-D.ietf-tsvwg-sctp-dtls-encaps].

For general considerations see [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch].

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

IANA is asked to update the reference of an already existing SCTP PPID assignment (Section 8.1) and to create a new standalone registry with its own URL for the DCEP (Section 8.2) containing two new registration tables (Section 8.2.1 and Section 8.2.2).

8.1. SCTP Payload Protocol Identifier

This document uses one already registered SCTP Payload Protocol Identifier (PPID) named "WebRTC Control". [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which this identifier was assigned. IANA is requested to update the reference of this assignment to point to this document and to update the name. The corresponding date should be kept.

Therefore this assignment should be updated to read:

Value	SCTP PPID	Reference	Date
WebRTC DCEP	50	[RFCXXXX]	2013-09-20

8.2. New Standalone Registry for the DCEP

IANA is requested to create a new standalone registry (aka a webpage) with its own URL for the Data Channel Establishment Protocol (DCEP). The title should be "Data Channel Establishment Protocol (DCEP) Parameters". It will contain the two tables as described in Section 8.2.1 and Section 8.2.2.

8.2.1. New Message Type Registry

IANA is requested to create a new registration table "Message Type Registry" for the Data Channel Establishment Protocol (DCEP) to manage the one byte "Message Type" field in DCEP messages (see Section 5). This registration table should be part of the registry described in Section 8.2.

The assignment of new message types is done through an RFC required action, as defined in [RFC5226]. Documentation of the new message type MUST contain the following information:

1. A name for the new message type;
2. A detailed procedural description of the use of messages with the new type within the operation of the Data Channel Establishment Protocol.

Initially the following values need to be registered:

Name	Type	Reference
Reserved	0x00	[RFCXXXX]
Reserved	0x01	[RFCXXXX]
DATA_CHANNEL_ACK	0x02	[RFCXXXX]
DATA_CHANNEL_OPEN	0x03	[RFCXXXX]
Unassigned	0x04-0xfe	
Reserved	0xff	[RFCXXXX]

Please note that the values 0x00 and 0x01 are reserved to avoid interoperability problems, since they have been used in earlier versions of the document. The value 0xff has been reserved for future extensibility. The range of possible values is from 0x00 to 0xff.

8.2.2. New Channel Type Registry

IANA is requested to create a new registration table "Channel Type Registry" for the Data Channel Establishment Protocol to manage the one byte "Channel Type" field in DATA_CHANNEL_OPEN messages (see Section 5.1). This registration table should be part of the registry described in Section 8.2.

The assignment of new message types is done through an RFC required action, as defined in [RFC5226]. Documentation of the new Channel Type MUST contain the following information:

1. A name for the new Channel Type;
2. A detailed procedural description of the user message handling for Data Channels using this new Channel Type.

Please note that if new Channel Types support ordered and unordered message delivery, the high order bit MUST be used to indicate whether the message delivery is unordered or not.

Initially the following values need to be registered:

Name	Type	Reference
DATA_CHANNEL_RELIABLE	0x00	[RFCXXXX]
DATA_CHANNEL_RELIABLE_UNORDERED	0x80	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT	0x01	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED	0x81	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED	0x02	[RFCXXXX]
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED	0x82	[RFCXXXX]
Reserved	0x7f	[RFCXXXX]
Reserved	0xff	[RFCXXXX]
Unassigned	rest	

Please note that the values 0x7f and 0xff have been reserved for future extensibility. The range of possible values is from 0x00 to 0xff.

9. Acknowledgments

The authors wish to thank Harald Alvestrand, Richard Barnes, Adam Bergkvist, Spencer Dawkins, Barry Dingle, Stefan Haekansson, Cullen Jennings, Paul Kyzivat, Doug Leonard, Alexey Melnikov, Pete Resnick, Irene Ruengeler, Randall Stewart, Peter Thatcher, Martin Thompson, Justin Uberti, and many others for their invaluable comments.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[I-D.ietf-tsvwg-sctp-dtls-encaps]
Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-dtls-encaps-07 (work in progress), December 2014.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-12 (work in progress), September 2014.

10.2. Informational References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-07 (work in progress), July 2014.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-10 (work in progress), July 2014.

Authors' Addresses

Randell Jesup
Mozilla
US

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
FI

Email: salvatore.loreto@ericsson.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

Email: tuexen@fh-muenster.de

RTCWEB Working Group
Internet-Draft
Intended status: Informational
Expires: July 27, 2015

C. Holmberg
S. Hakansson
G. Eriksson
Ericsson
January 23, 2015

Web Real-Time Communication Use-cases and Requirements
draft-ietf-rtcweb-use-cases-and-requirements-16.txt

Abstract

This document describes web based real-time communication use-cases. Requirements on the browser functionality are derived from the use-cases.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is being published to record the early conclusions of the working group. It will not be used as a set of rigid guidelines that specifications and implementations will be held to in the future.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 27, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	3
3. Use-cases	3
3.1. Introduction	4
3.2. Common requirements	4
3.3. Browser-to-browser use-cases	4
3.3.1. Simple Video Communication Service	4
3.3.2. Simple Video Communication Service, NAT/Firewall that blocks UDP	7
3.3.3. Simple Video Communication Service, Firewall that only allows traffic via a HTTP Proxy	7
3.3.4. Simple Video Communication Service, global service provider	7
3.3.5. Simple Video Communication Service, enterprise aspects	8
3.3.6. Simple Video Communication Service, access change . .	9
3.3.7. Simple Video Communication Service, QoS	10
3.3.8. Simple Video Communication Service with screen sharing	10
3.3.9. Simple Video Communication Service with file exchange	11
3.3.10. Hockey Game Viewer	11
3.3.11. Multiparty video communication	12
3.3.12. Multiparty on-line game with voice communication . .	14
3.4. Browser - GW/Server use cases	15
3.4.1. Telephony terminal	15
3.4.2. Fedex Call	16
3.4.3. Video conferencing system with central server	17
4. Requirements summary	18
4.1. General	18
4.2. Browser requirements	18
5. IANA Considerations	22
6. Security Considerations	22
6.1. Introduction	22
6.2. Browser Considerations	22
6.3. Web Application Considerations	23
7. Acknowledgements	23
8. Change Log	23
9. Normative References	30
Appendix A. API requirements	30

Authors' Addresses	33
------------------------------	----

1. Introduction

This document presents a few use-cases of web applications that are executed in a browser and use real-time communication capabilities. In most of the use-cases all end-user clients are web applications, but there are some use-cases where at least one of the end-user clients is of another type (e.g. a mobile phone or a SIP User Agent (UA)).

Based on the use-cases, the document derives requirements related to browser functionality. These requirements are named "Fn", where n is an integer, and are listed in conjunction with the use-cases. A summary is provided in Section 4.2.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is proposed to be used in a later phase to evaluate the protocols and solutions developed by the WG.

This document also lists requirements related to the API to be used by web applications as an appendix. The reason is that the W3C WebRTC WG has decided to not develop its own use-case/requirement document, but instead use this document. These requirements are named "An", where n is an integer, and are described in Appendix A.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is being published to record the early conclusions of the working group. It will not be used as a set of rigid guidelines that specifications and implementations will be held to in the future.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

3. Use-cases

3.1. Introduction

This section describes web based real-time communication use-cases, from which requirements are derived.

The following considerations are applicable to all use cases:

- o Clients can be on IPv4-only
- o Clients can be on IPv6-only
- o Clients can be on dual-stack
- o Clients can be connected to networks with different throughput capabilities
- o Clients can be on variable-media-quality networks (wireless)
- o Clients can be on congested networks
- o Clients can be on firewalled networks with no UDP allowed
- o Clients can be on networks with a NAT or IPv4-IPv6 translation devices using any type of Mapping and Filtering behaviors (as described in RFC4787).

3.2. Common requirements

The requirements retrieved from the Simple Video Communication Service use-case (Section 3.3.1) by default apply to all other use-cases, and are considered common. For each individual use-case, only the additional requirements are listed.

3.3. Browser-to-browser use-cases

3.3.1. Simple Video Communication Service

3.3.1.1. Description

Two or more users have loaded a video communication web application into their browsers, provided by the same service provider, and logged into the service it provides. The web service publishes information about user login status by pushing updates to the web application in the browsers. When one online user selects a peer online user, a 1-1 audiovisual communication session between the browsers of the two peers is initiated. The invited user might accept or reject the session.

During session establishment a self-view is displayed, and once the session has been established the video sent from the remote peer is displayed in addition to the self-view. During the session, each user can select to remove and re-insert the self-view as often as desired. Each user can also change the sizes of his/her two video displays during the session. Each user can also pause sending of media (audio, video, or both) and mute incoming media.

It is essential that media and data be encrypted, authenticated and integrity protected on a per IP packet basis and that media and data packets failing the integrity check not be delivered to the application.

The application gives the users the opportunity to stop it from exposing the host IP address to the application of the other user.

Any session participant can end the session at any time.

The two users may be using communication devices with different operating systems and browsers from different vendors.

The web service monitors the quality of the service (focus on quality of audio and video) the end-users experience.

3.3.1.2. Common Requirements

REQ-ID	DESCRIPTION
F1	The browser must be able to use microphones and cameras as input devices to generate streams.
F2	The browser must be able to send streams and data to a peer in the presence of NATs.
F3	Transmitted streams and data must be rate controlled (meaning that the browser must, regardless of application behavior, reduce send rate when there is congestion).
F4	The browser must be able to receive, process and render streams and data ("render" does not apply for data) from peers.
F5	The browser should be able to render good quality audio and video even in the presence of reasonable levels of jitter and packet losses.

- F6 The browser must detect when a stream from a peer is not received anymore.
-
- F7 When there are both incoming and outgoing audio streams, echo cancellation must be made available to avoid disturbing echo during conversation.
-
- F8 The browser must support synchronization of audio and video.
-
- F9 The browser should use encoding of streams suitable for the current rendering (e.g. video display size) and should change parameters if the rendering changes during the session.
-
- F10 The browser must support a baseline audio and video codec.
-
- F11 It must be possible to protect streams and data from wiretapping [RFC2804][RFC7258].
-
- F12 The browser must enable verification, given the right circumstances and by use of other trusted communication, that streams and data received have not been manipulated by any party.
-
- F13 The browser must encrypt, authenticate and integrity protect media and data on a per IP packet basis, and must drop incoming media and data packets that fail the per IP packet integrity check. In addition, the browser must support a mechanism for cryptographically binding media and data security keys to the user identity (see R-ID-BINDING in [RFC5479]).
-
- F14 The browser must make it possible to set up a call between two parties without one party learning the other party's host IP address.
-
- F15 The browser must be able to collect statistics, related to the transport of audio and video between peers, needed to estimate quality of experience.
-

A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A25, A26

3.3.2. Simple Video Communication Service, NAT/Firewall that blocks UDP

3.3.2.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that one of the users is behind a NAT/Firewall that blocks UDP traffic.

3.3.2.2. Additional Requirements

REQ-ID	DESCRIPTION
F18	The browser must be able to send streams and data to a peer in the presence of NATs and Firewalls that block UDP traffic.

3.3.3. Simple Video Communication Service, Firewall that only allows traffic via a HTTP Proxy

3.3.3.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that one of the users is behind a Firewall that only allows traffic via a HTTP Proxy.

3.3.3.2. Additional Requirements

REQ-ID	DESCRIPTION
F21	The browser must be able to send streams and data to a peer in the presence of Firewalls that only allows traffic via a HTTP Proxy, when Firewall policy allows WebRTC traffic.

3.3.4. Simple Video Communication Service, global service provider

3.3.4.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1).

What is added is that the service provider is operating over large geographical areas (or even globally).

Assuming that the Interactive Connectivity Establishment (ICE) mechanism [RFC5245] will be used, this means that the service provider would like to be able to provide several STUN and TURN servers (via the app) to the browser; selection of which one(s) to use is part of the ICE processing. Other reasons for wanting to provide several STUN and TURN servers include support for IPv4 and IPv6, load balancing and redundancy.

Note that ICE support being mandatory does not preclude a WebRTC endpoint from supporting more traversal mechanisms than ICE using STUN and TURN.

3.3.4.2. Additional Requirements

REQ-ID	DESCRIPTION
F19	The browser must be able to use several STUN and TURN servers

A22

3.3.5. Simple Video Communication Service, enterprise aspects

3.3.5.1. Description

This use-case is similar to the Simple Video Communication Service use-case (Section 3.3.1).

What is added is aspects when using the service in enterprises. ICE is assumed in the further description of this use-case.

An enterprise that uses a RTCWEB based web application for communication desires to audit all RTCWEB based application sessions used from inside the company towards any external peer. To be able to do this they deploy a TURN server that straddles the boundary between the internal and the external network.

The firewall will block all attempts to use STUN with an external destination unless they go to the enterprise auditing TURN server. In cases where employees are using RTCWEB applications provided by an external service provider they still want the traffic to stay inside their internal network and in addition not load the straddling TURN server, thus they deploy a STUN server allowing the RTCWEB client to

determine its server reflexive address on the internal side. Thus enabling cases where peers are both on the internal side to connect without the traffic leaving the internal network. It must be possible to configure the browsers used in the enterprise with network specific STUN and TURN servers. This should be possible to achieve by auto-configuration methods. The RTCWEB functionality will need to utilize both network specific STUN and TURN resources and STUN and TURN servers provisioned by the web application.

3.3.5.2. Additional Requirements

REQ-ID	DESCRIPTION
F20	The browser must support the use of STUN and TURN servers that are supplied by entities other than the web application (i.e. the network provider).

3.3.6. Simple Video Communication Service, access change

3.3.6.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that the user changes network access during the session.

The communication device used by one of the users has several network adapters (Ethernet, WiFi, Cellular). The communication device is accessing the Internet using Ethernet, but the user has to start a trip during the session. The communication device automatically changes to use WiFi when the Ethernet cable is removed and then moves to cellular access to the Internet when moving out of WiFi coverage. The session continues even though the access method changes.

3.3.6.2. Additional Requirements

REQ-ID	DESCRIPTION
F17	The communication session must survive across a change of the network interface used by the session

3.3.7. Simple Video Communication Service, QoS

3.3.7.1. Description

This use-case is almost identical to the Simple Video Communication Service, access change use-case (Section 3.3.6). The use of Quality of Service (QoS) capabilities is added:

The user in the previous use case that starts a trip is behind a common residential router that supports differentiation of traffic. In addition, the user's provider of cellular access has QoS support enabled. The user is able to take advantage of the QoS support both when accessing via the residential router and when using cellular.

3.3.7.2. Additional Requirements

REQ-ID	DESCRIPTION
F17	The communication session must survive across a change of the network interface used by the session
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.

3.3.8. Simple Video Communication Service with screen sharing

3.3.8.1. Description

This use-case has the audio and video communication of the Simple Video Communication Service use-case (Section 3.3.1).

But in addition to this, one of the users can share what is being displayed on her/his screen with a peer. The user can choose to share the entire screen, part of the screen (part selected by the user) or what a selected application displays with the peer.

3.3.8.2. Additional Requirements

REQ-ID	DESCRIPTION
F36	The browser must be able to generate streams using the entire user display, a specific area of the user's display or the information being displayed by a specific application.

A21

3.3.9. Simple Video Communication Service with file exchange

3.3.9.1. Description

This use-case has the audio and video communication of the Simple Video Communication Service use-case (Section 3.3.1).

But in addition to this, the users can send and receive files stored in the file system of the device used.

3.3.9.2. Additional Requirements

REQ-ID	DESCRIPTION
F35	The browser must be able to send reliable data traffic to a peer browser.

A21, A24

3.3.10. Hockey Game Viewer

3.3.10.1. Description

An ice-hockey club uses an application that enables talent scouts to, in real-time, show and discuss games and players with the club manager. The talent scouts use a mobile phone with two cameras, one front facing and one rear facing.

The club manager uses a desktop, equipped with one camera, for viewing the game and discussing with the talent scout.

Before the game starts, and during game breaks, the talent scout and the manager have a 1-1 audiovisual communication session. On the mobile phone, only the camera facing the talent scout is used. On the user display of the mobile phone, the video of the club manager

is shown with a picture-in-picture thumbnail of the rear facing camera (self-view). On the display of the desktop, the video of the talent scout is shown with a picture-in-picture thumbnail of the desktop camera (self-view).

When the game is on-going, the talent scout activates the use of the front facing camera, and that stream is sent to the desktop (the stream from the rear facing camera continues to be sent all the time). The video stream captured by the front facing camera (that is capturing the game) of the mobile phone is shown in a big window on the desktop screen, with picture-in-picture thumbnails of the rear facing camera and the desktop camera (self-view). On the display of the mobile phone the game is shown (front facing camera) with picture-in-picture thumbnails of the rear facing camera (self-view) and the desktop camera. As the most important stream in this phase is the video showing the game, the application used in the talent scout's mobile sets higher priority for that stream.

3.3.10.2. Additional Requirements

REQ-ID	DESCRIPTION
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.
F25	The browser must be able to render several concurrent audio and video streams.

A17, A23

3.3.11. Multiparty video communication

3.3.11.1. Description

In this use-case, the Simple Video Communication Service use-case (Section 3.3.1) is extended by allowing multiparty sessions. No central server is involved - the browser of each participant sends and receives streams to and from all other session participants. The web application in the browser of each user is responsible for setting up streams to all receivers.

In order to enhance the user experience, the web application renders the audio coming from different participants so that it is

experienced to come from different spatial locations. This is done automatically, but users can change how the different participants are placed in the (virtual) room. In addition the levels in the audio signals are adjusted before mixing.

Another feature intended to enhance the use experience is that the video window that displays the video of the currently speaking peer is highlighted.

Each video stream received is by default displayed in a thumbnail frame within the browser, but users can change the display size.

Note: What this use-case adds in terms of requirements is capabilities to send streams to and receive streams from several peers concurrently, as well as the capabilities to render the video from all received streams and be able to spatialize, level adjust and mix the audio from all received streams locally in the browser. It also adds the capability to measure the audio level/activity.

3.3.11.2. Additional Requirements

REQ-ID	DESCRIPTION
F23	The browser must be able to transmit streams and data to several peers concurrently.
F24	The browser must be able to receive streams and data from multiple peers concurrently.
F25	The browser must be able to render several concurrent audio and video streams.
F26	The browser must be able to mix several audio streams.
F27	The browser must be able to apply spatialization effects to audio streams.
F28	The browser must be able to measure the voice activity level in audio streams.
F29	The browser must be able to change the voice activity level in audio streams.
A13, A14, A15, A16	

3.3.12. Multiparty on-line game with voice communication

3.3.12.1. Description

This use case is based on the previous one. In this use-case, the voice part of the multiparty video communication use case is used in the context of an on-line game. The received voice audio media is rendered together with game sound objects. For example, the sound of a tank moving from left to right over the screen must be rendered and played to the user together with the voice media.

Quick updates of the game state is required, and have higher priority than the voice.

Note: the difference regarding local audio processing compared to the "Multiparty video communication" use-case is that other sound objects than the streams must be possible to be included in the spatialization and mixing. "Other sound objects" could for example be a file with the sound of the tank; that file could be stored locally or remotely.

3.3.12.2. Additional Requirements

REQ-ID	DESCRIPTION
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.
F23	The browser must be able to transmit streams and data to several peers concurrently.
F24	The browser must be able to receive streams and data from multiple peers concurrently.
F25	The browser must be able to render several concurrent audio and video streams.
F26	The browser must be able to mix several audio streams.
F27	The browser must be able to apply spatialization effects when playing audio streams.
F28	The browser must be able to measure the voice activity level in audio streams.
F29	The browser must be able to change the voice activity level in audio streams.
F30	The browser must be able to process and mix sound objects (media that is retrieved from another source than the established media stream(s) with the peer(s) with audio streams.
F34	The browser must be able to send short latency unreliable datagram traffic to a peer browser [RFC5405].

A13, A14, A15, A16, A17, A18, A23

3.4. Browser - GW/Server use cases

3.4.1. Telephony terminal

3.4.1.1. Description

A mobile telephony operator allows its customers to use a web browser to access their services. After a simple log in the user can place and receive calls in the same way as when using a normal mobile phone. When a call is received or placed, the identity is shown in the same manner as when a mobile phone is used.

Note: With "place and receive calls in the same way as when using a normal mobile phone" it is meant that you can dial a number, and that your mobile telephony operator has made available your phone contacts on line, so they are available and can be clicked to call, and be used to present the identity of an incoming call. If the callee is not in your phone contacts the number is displayed. Furthermore, your call logs are available, and updated with the calls made/received from the browser. And for people receiving calls made from the web browser the usual identity (i.e. the phone number of the mobile phone) will be presented.

3.4.1.2. Additional Requirements

REQ-ID	DESCRIPTION
F31	The browser must support an audio media format (codec) that is commonly supported by existing telephony services.
F33	The browser must be able to initiate and accept a media session where the data needed for establishment can be carried in SIP.

3.4.2. Fedex Call

3.4.2.1. Description

Alice uses her web browser with a service that allows her to call PSTN numbers. Alice calls 1-800-gofedex. Alice should be able to hear the initial prompts from the fedex Interactive Voice Responder (IVR) and when the IVR says press 1, there should be a way for Alice to navigate the IVR.

3.4.2.2. Additional Requirements

REQ-ID	DESCRIPTION
F31	The browser must support an audio media format (codec) that is commonly supported by existing telephony services.
F32	There should be a way to navigate a Dual-tone multi-frequency signaling (DTMF) based Interactive voice response (IVR) System

3.4.3. Video conferencing system with central server

3.4.3.1. Description

An organization uses a video communication system that supports the establishment of multiparty video sessions using a central conference server.

The browser of each participant sends an audio stream (type in terms of mono, stereo, 5.1, ... depending on the equipment of the participant) to the central server. The central server mixes the audio streams (and can in the mixing process naturally add effects such as spatialization) and sends towards each participant a mixed audio stream which is played to the user.

The browser of each participant sends video towards the server. For each participant one high resolution video is displayed in a large window, while a number of low resolution videos are displayed in smaller windows. The server selects what video streams to be forwarded as main- and thumbnail videos respectively, based on speech activity. As the video streams to display can change quite frequently (as the conversation flows) it is important that the delay from when a video stream is selected for display until the video can be displayed is short.

All participants are authenticated by the central server, and authorized to connect to the central server. The participants are identified to each other by the central server, and the participants do not have access to each others' credentials such as e-mail addresses or login IDs.

Note: This use-case adds requirements on support for fast stream switches F16. There exist several solutions that enable the server to forward one high resolution and several low resolution video streams: a) each browser could send a high resolution, but scalable

stream, and the server could send just the base layer for the low resolution streams, b) each browser could in a simulcast fashion send one high resolution and one low resolution stream, and the server just selects or c) each browser sends just a high resolution stream, the server transcodes into low resolution streams as required.

3.4.3.2. Additional Requirements

REQ-ID	DESCRIPTION
F16	The browser must support insertion of reference frames in outgoing media streams when requested by a peer.
F25	The browser must be able to render several concurrent audio and video streams.

4. Requirements summary

4.1. General

This section contains the requirements on the browser derived from the use-cases in Section 3.

NOTE: It is assumed that the user applications are executed on a browser. Whether the capabilities to implement specific browser requirements are implemented by the browser application, or are provided to the browser application by the underlying operating system, is outside the scope of this document.

4.2. Browser requirements

Common, basic requirements	
REQ-ID	DESCRIPTION
F1	The browser must be able to use microphones and cameras as input devices to generate streams.
F2	The browser must be able to send streams and data to a peer in the presence of NATs.
F3	Transmitted streams and data must be rate controlled (meaning that the browser must, regardless of application behavior, reduce send rate when there is congestion).

-
- F4 The browser must be able to receive, process and render streams and data ("render" does not apply for data) from peers.
-
- F5 The browser should be able to render good quality audio and video even in the presence of reasonable levels of jitter and packet losses.
-
- F6 The browser must detect when a stream from a peer is not received anymore
-
- F7 When there are both incoming and outgoing audio streams, echo cancellation must be made available to avoid disturbing echo during conversation.
-
- F8 The browser must support synchronization of audio and video.
-
- F9 The browser should use encoding of streams suitable for the current rendering (e.g. video display size) and should change parameters if the rendering changes during the session
-
- F10 The browser must support a baseline audio and video codec
-
- F11 It must be possible to protect streams and data from wiretapping [RFC2804][RFC7258].
-
- F12 The browser must enable verification, given the right circumstances and by use of other trusted communication, that streams and data received have not been manipulated by any party.
-
- F13 The browser must encrypt, authenticate and integrity protect media and data on a per-packet basis, and must drop incoming media and data packets that fail the per-packet integrity check. In addition, the browser must support a mechanism for cryptographically binding media and data security keys to the user identity (see R-ID-BINDING in [RFC5479]).
-
- F14 The browser must make it possible to set up a call between two parties without one party

learning the other party's host IP address.

F15 The browser must be able to collect statistics,
 related to the transport of audio and video
 between peers, needed to estimate quality of
 experience.

Requirements related to network and topology

REQ-ID DESCRIPTION

F16 The browser must support insertion of reference frames
 in outgoing media streams when requested by a peer.

F17 The communication session must survive across a
 change of the network interface used by the
 session

F18 The browser must be able to send streams and
 data to a peer in the presence of NATs and
 Firewalls that block UDP traffic.

F19 The browser must be able to use several STUN
 and TURN servers

F20 The browser must support the use of STUN and TURN
 servers that are supplied by entities other than
 the web application (i.e. the network provider).

F21 The browser must be able to send streams and
 data to a peer in the presence of Firewalls that only
 allows traffic via a HTTP Proxy, when Firewall policy
 allows WebRTC traffic.

F22 The browser should be able to take advantage
 of available capabilities (supplied by network
 nodes) to differentiate voice, video and data
 appropriately.

Requirements related to multiple peers and streams

REQ-ID DESCRIPTION

F23 The browser must be able to transmit streams and
 data to several peers concurrently.

F24 The browser must be able to receive streams and
 data from multiple peers concurrently.

F25 The browser must be able to render several
 concurrent audio and video streams.

F26 The browser must be able to mix several
 audio streams.

Requirements related to audio processing

REQ-ID DESCRIPTION

F27 The browser must be able to apply spatialization
 effects when playing audio streams.

F28 The browser must be able to measure the
 voice activity level in audio streams.

F29 The browser must be able to change the
 voice activity level in audio streams.

F30 The browser must be able to process and mix
 sound objects (media that is retrieved from
 another source than the established media
 stream(s) with the peer(s) with audio streams.

Requirements related to legacy interop

REQ-ID DESCRIPTION

F31 The browser must support an audio media format
 (codec) that is commonly supported by existing
 telephony services.

F32 There should be a way to navigate
 a Dual-tone multi-frequency signaling (DTMF)
 based Interactive voice response (IVR) System

F33 The browser must be able to initiate and
 accept a media session where the data needed
 for establishment can be carried in SIP.

Other requirements

REQ-ID DESCRIPTION

F34 The browser must be able to send short
 latency unreliable datagram traffic to a
 peer browser [RFC5405].

-
- F35 The browser must be able to send reliable data traffic to a peer browser.
-
- F36 The browser must be able to generate streams using the entire user display, a specific area of the user's display or the information being displayed by a specific application.
-

5. IANA Considerations

There are no IANA actions in this document.

6. Security Considerations

6.1. Introduction

A malicious web application might use the browser to perform Denial Of Service (DOS) attacks on NAT infrastructure, or on peer devices. Also, a malicious web application might silently establish outgoing, and accept incoming, streams on an already established connection.

Based on the identified security risks, this section will describe security considerations for the browser and web application.

6.2. Browser Considerations

The browser is expected to provide mechanisms for getting user consent to use device resources such as camera and microphone.

The browser is expected to provide mechanisms for informing the user that device resources such as camera and microphone are in use ("hot").

The browser must provide mechanisms for users to revise and even completely revoke consent to use device resources such as camera and microphone.

The browser is expected to provide mechanisms for getting user consent to use the screen (or a certain part of it) or what a certain application displays on the screen as source for streams.

The browser is expected to provide mechanisms for informing the user that the screen, part thereof or an application is serving as a stream source ("hot").

The browser must provide mechanisms for users to revise and even completely revoke consent to use the screen, part thereof or an application is serving as a stream source.

The browser is expected to provide mechanisms in order to assure that streams are the ones the recipient intended to receive.

The browser is expected to provide mechanisms that allows the users to verify that the streams received have not be manipulated (F12).

The browser needs to ensure that media is not sent, and that received media is not rendered, until the associated stream establishment and handshake procedures with the remote peer have been successfully finished.

The browser needs to ensure that the stream negotiation procedures are not seen as Denial Of Service (DOS) by other entities.

6.3. Web Application Considerations

The web application is expected to ensure user consent in sending and receiving media streams.

7. Acknowledgements

The authors wish to thank Bernard Aboba, Gunnar Hellstrom, Martin Thomson, Lars Eggert, Matthew Kaufman, Emil Ivov, Eric Rescorla, Eric Burger, John Leslie, Dan Wing, Richard Barnes, Barry Dingle, Dale Worley, Ted hardie, Mary Barnes, Dan Burnett, Stephan Wenger, Harald Alvestrand, Cullen Jennings, Andrew Hutton and everyone else in the RTCWEB community that have provided comments, feedback, text and improvement proposals on the document. A big thank you to everyone that provided comments as part of the IESG evaluation, and to everyone else that provided comments and input in order to improve the document.

8. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-ietf-rtcweb-use-cases-and-requirements-15

- o Changes based on comment from Stephen Farrell:

- o - A1 modified, to also cover access to the local file system.
- o Changes based on comments from Benoit Claise:
- o - RFC 5245 added to references.
- o - Note added to Annex A, indicating that the API requirements are not normative.
- o Changes based on comments from Brian Carpenter:
- o - RFC 7258 added to references.
- o - Terminology fixes:
- o -- 'prioritize' -> 'differentiate'.
- o -- 'prioritization' -> 'differentiation'.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-14

- o Changes based on comments from the ops-dir:
- o - Editorial fixes.
- o - F13: 'per-packet basis' -> 'per IP packet basis'.
- o - F22: Text corrected in one occurrence.
- o - F25: 'audio' added.
- o Changes based on comments from IESG
- o - Editorial fixes.
- o - Disclaimer text suggested by Alissa Cooper added.
- o - F11: Reference to RFC 7258 added.
- o - F27: 'when playing' removed.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-10

- o Described that the API requirements are really from a W3C perspective and are supplied as an appendix in the introduction. Moved API requirements to an Appendix.

- o Removed the "Conventions" section with the key-words and reference to RFC2119. Also changed uppercase MUST's/SHOULD's to lowercase.
- o Added a note on the proposed use of the document to the introduction.
- o Removed the note talking about WS from the "Firewall that only allows http" use-case.
- o Removed the word "Skype" that was used as example in one of the use-cases.
- o Clarified F3 (the req saying the everything the browser sends must be rate controlled).
- o Removed the TBD saying we need to define reasonable levels from the requirement saying that quality must be good even in presence of packet losses (F5), and changed "must" to "should" (Based on a list discussion involving Bernard).
- o Removed F6 ("The browser must be able to handle high loss and jitter levels in a graceful way."), also after a list discussion.
- o Clarified F7 (used to say that the browser must support fast stream switches, now says that reference frames must be inserted when requested).
- o Removed the questions from F9 (echo cancellation), F10 (synchronization), F21 (telephony codec).
- o Exchanged "restrictive firewalls" for "limited middleboxes" in F19 (as proposed by Martin).
- o Expanded DTMF and IVR in F22 (proposed by Martin)
- o Added ref to RFC5405 in F23 (proposed by Lars Eggert).
- o Exchanged "service provided" for "web application" in F32.
- o Changed the text in 3.2.1 that motivates F36 (new text "It is essential that media and data be encrypted, authenticated ... bound to the user identity."); and rewrote F36, included a ref to RFC5479.
- o Changed "quality of service" to "quality of experience" in F38.
- o Added F39.

- o Used new formulation of A17 (proposed by Martin).
- o Updated A20.
- o Updated A25.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-09

- o Changed "video communication session" to "audiovisual communication session."

Changes from draft-ietf-rtcweb-use-cases-and-requirements-08

- o Changed "eavesdropping" to "wiretapping" and referenced RFC2804.
- o Removed informal ref webrtc_req; that document has been abandoned by the W3C webrtc WG.
- o Added use-case where one user is behind a Firewall that only allows http; derived req. F37.
- o Changed F24 slightly; MUST-> SHOULD, inserted "available".
- o Added a clause to "Simple video communication service" saying that the service provider monitors the quality of service, and derived reqs F38 and A26.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-07

- o Added "and data exchange" to 1. Introduction.
- o Removed cone and symmetric NAT from 4.1 Introduction, refers to RFC4787 instead.
- o Added text on enabling verification of that the media has not been manipulated by anyone to use-case "Simple Video Communication Service", derived req. F35
- o Added text on that the browser should reject media (data) that has been created/injected/modified by non-trusted party, derived req. F36
- o Added text on enabling the app to refrain from revealing IP address to use-case "Simple Video Communication Service", derived req. A25
- o Added use-case "Simple Video Communication Service with file exchange", derived reqs F33 and A24

- o Added priority of video streams to "Hockey game viewer" use case, added priority of data to "on-line game use-case", derived reqs F34 and A23
- o In F22, "the IVR" -> "a DTMF based IVR".
- o Updated req F23 to clarify that requirements such as NAT traversal, protection from eavesdropping, rate control applies also to datagram.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-06

- o Renaming of requirements (FaI1 -> F31), (FaI2 -> F32) and (AaI1 -> A22)

Changes from draft-ietf-rtcweb-use-cases-and-requirements-05

- o Added use-case "global service provider", derived reqs associated with several STUN/TURN servers
- o Added use-case "enterprise aspects", derived req associated with enabling the network provider to supply STUN and TURN servers
- o The requirements from the above are ICE specific and labeled accordingly
- o Separated the requirements phrased like "processing such as pan, mix and render" for audio to be specific reqs on spatialization, level measurement, level adjustment and mixing (discussed on the lists in <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01648.html> and <http://lists.w3.org/Archives/Public/public-webrtc/2011Sep/0102.html>)
- o Added use-case on sharing as decided in <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01700.html>, derived reqs F30 and A21
- o Added the list of common considerations proposed in mail <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01562.html> to the Introduction of the use-case section

Changes from draft-ietf-rtcweb-use-cases-and-requirements-04

- o Most changes based on the input from Dan Burnett <http://www.ietf.org/mail-archive/web/rtcweb/current/msg00948.html>
- o Many editorial changes
- o 4.2.1.1 Clarified

- o Some clarification added to 4.3.1.1 as a note
- o F-requirements updated (see reply to Dan's mail).
- o Almost all A-requirements updated to start "The Web API MUST provide ..."
- o A8 removed, A9 rephrased to cover A8 and old A9
- o A15 rephrased
- o For more details, and discussion, look at the response to Dan's mail <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01177.html>

Changes from draft-ietf-rtcweb-use-cases-and-requirements-03

- o Editorials
- o Changed when the self-view is displayed in 4.2.1.1, and added words about allowing users to remove and re-insert it.
- o Clarified 4.2.6.1
- o Removed the "mono" stuff from 4.2.7.1
- o Added that communication should not be possible to eavesdrop to most use cases - and req. F17
- o Re-phrased 4.3.3.1 to not describe the technical solution so much, and removed "stereo" stuff. Solution possibilities are now in a note.
- o Re-inserted API requirements after discussion in the W3C webrtc WG. (Re-phrased A15 and added A18 compared to version -02).

Changes from draft-ietf-rtcweb-use-cases-and-requirements-02

- o Removed description/list of API requirements, instead
- o Reference to W3C webrtc_reqs document for API requirements

Changes from draft-ietf-rtcweb-ucreqs-01

- o Changed Intended status to Information
- o Changed "Ipr" to "trust200902"

- o Added use case "Simple video communication service, NAT/Firewall that blocks UDP", and derived new req F26
- o Added use case "Distributed Music Band" and derived new req A17
- o Added F24 as requirement derived from use case "Simple video communication service with inter-operator calling"
- o Added section "Additional use cases"
- o Added text about ID handling to multiparty with central server use case
- o Re-phrased A1 slightly

Changes from draft-ietf-rtcweb-ucreqs-00

- o - Reshuffled: Just two main groups of use cases (b2b and b2GW/Server); removed some specific use cases and added them instead as flavors to the base use case (Simple video communication)
- o - Changed the formulation of F19
- o - Removed the requirement on an API for DTMF
- o - Removed "FX3: There SHOULD be a mapping of the minimum needed data for setting up connections into SIP, so that the restriction to SIP-carriable data can be verified. Not a rew on the browser but rather on a document"
- o - (see <http://www.ietf.org/mail-archive/web/rtcweb/current/msg00227.html> for more details)
- o -Added text on informing user of that mic/cam is being used and that it must be possible to revoke permission to use them in section 7.

Changes from draft-holmberg-rtcweb-ucreqs-01

- o - Draft name changed to draft-ietf-rtcweb-ucreqs
- o - Use-case grouping introduced
- o - Additional use-cases added
- o - Additional reqs added (derived from use cases): F19-F25, A16-A17

Changes from draft-holmberg-rtcweb-ucreqs-00

- o - Mapping between use-cases and requirements added (Harald Alvestrand, 090311)
- o - Additional security considerations text (Harald Alvestrand, 090311)
- o - Clarification that user applications are assumed to be executed by a browser (Ted Hardie, 080311)
- o - Editorial corrections and clarifications

9. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Appendix A. API requirements

This section contains the requirements on the API derived from the use-cases in Section 3.

NOTE: As W3C is responsible for the API, the API requirements in this specification are not normative.

REQ-ID	DESCRIPTION

A1	The Web API must provide means for the application to ask the browser for permission

to use cameras and microphones as input devices,
and to have access to the local file system.

-
- A2 The Web API must provide means for the web application to control how streams generated by input devices are used.
-
- A3 The Web API must provide means for the web application to control the local rendering of streams (locally generated streams and streams received from a peer).
-
- A4 The Web API must provide means for the web application to initiate sending of stream/stream components to a peer.
-
- A5 The Web API must provide means for the web application to control the media format (codec) to be used for the streams sent to a peer.
- NOTE: The level of control depends on whether the codec negotiation is handled by the browser or the web application.
-
- A6 The Web API must provide means for the web application to modify the media format for streams sent to a peer after a media stream has been established.
-
- A7 The Web API must provide means for informing the web application of whether the establishment of a stream with a peer was successful or not.
-
- A8 The Web API must provide means for the web application to mute/unmute a stream or stream component(s). When a stream is sent to a peer mute status must be preserved in the stream received by the peer.
-
- A9 The Web API must provide means for the web application to cease the sending of a stream to a peer.
-
- A10 The Web API must provide means for the web application to cease processing and rendering of a stream received from a peer.
-

- A11 The Web API must provide means for informing the web application when a stream from a peer is no longer received.
-
- A12 The Web API must provide means for informing the web application when high loss rates occur.
-
- A13 The Web API must provide means for the web application to apply spatialization effects to audio streams.
-
- A14 The Web API must provide means for the web application to detect the level in audio streams.
-
- A15 The Web API must provide means for the web application to adjust the level in audio streams.
-
- A16 The Web API must provide means for the web application to mix audio streams.
-
- A17 The Web API must provide a way to identify streams such that an application is able to match streams on a sending peer with the same stream on all receiving peers.
-
- A18 The Web API must provide a mechanism for sending and receiving isolated discrete chunks of data.
-
- A19 The Web API must provide means for the web application to indicate the type of audio signal (speech, audio) for audio stream(s)/stream component(s).
-
- A20 It must be possible for an initiator or a responder web application to indicate the types of media it is willing to accept incoming streams for when setting up a connection (audio, video, other). The types of media to be accepted can be a subset of the types of media the browser is able to accept.
-
- A21 The Web API must provide means for the application to ask the browser for permission to the screen, a certain area on the screen or what a certain application displays on the

screen as input to streams.

-
- A22 The Web API must provide means for the application to specify several STUN and/or TURN servers to use.
-
- A23 The Web API must provide means for the application to specify the priority to apply for outgoing streams and data.
-
- A24 The Web API must provide a mechanism for sending and receiving files.
-
- A25 It must be possible for the application to instruct the browser to refrain from exposing the host IP address to the application
-
- A26 The Web API must provide means for the application to obtain the statistics (related to transport, and collected by the browser) needed to estimate quality of service.
-

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Stefan Hakansson
Ericsson
Laboratoriegatan 11
Lulea 97128
Sweden

Email: stefan.lk.hakansson@ericsson.com

Goran AP Eriksson
Ericsson
Farogatan 6
Stockholm 16480
Sweden

Email: goran.ap.eriksson@ericsson.com