Media API Requirements for Real-Time Interoperation of Web User Agents
draft-thomson-rtcweb-api-reqs-00

Abstract

   Direct, real-time communications between web user agents (browsers)
   requires two points of standardization. The first describes the on-
   the-wire protocol that is used. The second is the API that controls
   and configures what gets put on the wire and how. The capabilities
   of the protocol determines the nature of the API. This document
   outlines how the constraints imposed upon the API by the protocol.

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 January 10, 2013.

Table of Contents

1. Introduction

   The exchange of media - audio and video - between peers is a basic
   foundation of telecommunications. The fact that the web platform has
   successfully managed to avoid this fundamental feature for so long is
   a testament to the usefulness of its other features. The WebRTC/
   rtcweb effort attempts to remedy this shortcoming.

   Figure 1 shows the architecture for real-time communications on the
   web.

```
                          _____
                        ((         ))
                     .-(( The Cloud ))-.
                ?  /   ((_____))   \ ?
                  /                     \
      +-----------/----+         +--\-------------+
      | +---------/---+ |         | +-\-----------+ |
      | | Javascript  | |         | | Javascript  | |
      | |           | |         | |           | |
      | +--<W3C API>--+ |    IETF    | +--<W3C API>--+ |
      |                === protocols ===              |
      | Web User Agent  |         | Web User Agent  |
      +----------------+         +----------------+
```
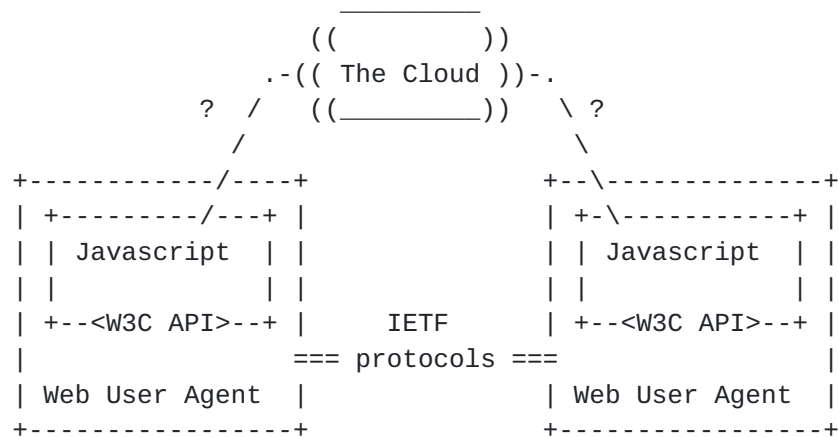
                  Figure 1: Architecture for the Real-Time Web

   Two points of interoperation are necessary in this architecture: API
   and peer-to-peer protocols. Critically, the points marked with a '?'
   are not subject to standardization. These points are effectively
   internal to the application that exists both in "The Cloud" and in
   the Javascript virtual machine provided by the web user agent.

   The IETF rtcweb working group has settled on Real-Time Transport
   Protocol (RTP) [RFC3550], Secure RTP (SRTP) [RFC3711] and Interactive
   Connectivity Establishment (ICE) [RFC5245] for the peer to peer
   protocol components. Use of RTP is documented in
   [I-D.ietf-rtcweb-rtp-usage]; use of SRTP and ICE is described in the
   rtcweb security architecture [I-D.ietf-rtcweb-security-arch].

   ICE and RTP are unable to operate without an out-of-band
   communications channel. For ICE this provides bootstrapping
   information; for RTP, a common understanding of the key protocol
   parameters. Though SRTP can operate without an out-of-band channel
   if certain assumptions are made, many uses depend on some form of
   out-of-band communication. The out-of-band channel is provided by
   the web application. The necessary information passes through the
   API to the browser.

This document describes what information must pass between a web
application and a web user agent in order to successfully establish
peer-to-peer media communications. No attempt is made to constrain
what the API looks like. After all:

>   [...] when [a] IETF documents start telling you how to build
>   Javascript APIs, you should run far away... quickly. :)
>   -- [I-D.kaplan-rtcweb-api-reqs]

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED",
"SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT
RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as
described in BCP 14, RFC 2119 [RFC2119] and indicate requirement
levels for compliant implementations.

2. Peer-to-Peer Transport

ICE [RFC5245] provides a mechanism for establishing peer-to-peer
communications using UDP.

The ICE process applies to a single UDP flow from a source address
and port to a destination address and port. Where multiple flows are
required, the process is applied once for each flow.

2.1. Overview of ICE Operation

In ICE, for each UDP flow, each peer performs the following steps:

o Candidate gathering. A candidate is an UDP port with associated
   session authentication parameters. Candidates are attributed to
   the peer, but they can be collected from the local host, by
   querying servers about the server perspective of the peer address,
   or a TURN [RFC5766] relay can allocate server-based ports that
   forward packets to the peer.

o Candidate exchange. Each peer learns of the candidates gathered
   by the other peer.

o Candidate pair testing. Candidates are paired, one local against
   one remote. A connectivity check is performed where a STUN
   [RFC5389] packet is sent from the local candidate to the remote.
   The pair is valid if a response arrives. Packets might be lost,
   so this process is repeated.

   o Candidate pair selection. As soon as one or more candidates are
      marked as valid, one of these can be selected and used.

[2.2](). Transport ICE Requirements

   The following requirements ensure that an application is able to
   establish a UDP flow between peers with proper consent:

   ICE-1 The application MUST be able to request the gathering of host
          candidates. Candidates are comprised of an IP address (v4 or
          v6), a UDP port number, an ICE username fragment and an ICE
          password.

   ICE-2 The application MUST be able to request the gathering of
          server reflexive address information using STUN. Input to
          this is the identity of a STUN server and any credentials
          required by the server.

   ICE-3 The application MUST be able to request the allocation of TURN
          relay candidates. Input to this is the identity of a TURN
          server and any credentials required by the server.

   ICE-4 The application MUST be able to request that a connectivity
          check be sent to a remote candidate and to be informed of
          success. Input to this is a choice of local candidate and
          details of the remote candidate.

          This establishes: whether the UDP packet is able to reach the
          remote peer and whether the remote peer consents to the
          receipt of packets from this peer.

   ICE-5 The application MUST be able to add STUN attributes to the
          STUN messages that are sent for connectivity checks.

   ICE-6 The application MUST be able to examine STUN attributes
          received on successful responses to connectivity checks.

   ICE-7 The application MUST be able to retrieve peer reflexive
          addresses that are discovered when connectivity checks are
          successful.

   ICE-8 The application MUST be able to request the creation of a UDP
          flow on a valid candidate pair.

   ICE-9 The application MUST be notified when a successful
          connectivity check is received from remote peer. The
          notification MUST include peer addressing information.

2.3. Established Flow Requirements

   The following requirements apply to established flows:

   UDP-1 The application MUST be able to send connectivity checks on an
          active UDP flow to test liveness of the flow.

   UDP-2 The application MUST be notified when consent for an active
          UDP flow expires.

   UDP-3 The application MUST be able to retrieve the bandwidth that
          the remote peer consents to receive for the UDP flow.

   UDP-4 The application MUST be able to learn the bandwidth limit that
          the browser has detected for the flow based on feedback from
          the network.

   UDP-5 The application MUST be notified when the browser detects
          changes in the available bandwidth for the UDP flow.

   UDP-6 The application MUST be notified when the browser detects
          network congestion that affects the UDP flow.

   UDP-7 The application MUST be notified when changes in network
          connectivity occur.

   UDP-8 The application MUST be able to pause connectivity checking on
          an active UDP flow.

          This allows an application to conserve resources for inactive
          flows, especially for battery-powered devices.

3. Securing Media

   SRTP [RFC3711] provides confidentiality, message authentication and
   replay protection for RTP media.

   SRTP requires external key negotiation. Two methods are possible:
   Datagram Transport Layer Security (DTLS) can be used for key
   negotiation (DTLS-SRTP) [RFC5764], or the SRTP master keys can be
   provided directly. The application chooses which of these modes the
   application uses. [[Ed: requirement for second option not
   established]]

   With DTLS key negotiation, the asymmetry of the DTLS handshake means
   that out-of-band methods must be used to determine whether peers act
   in the server or client roles. DTLS offers a means for peer

   authentication, allowing the application to learn of the identity of
   the peer through the certificate they offer during the handshake.

   Without DTLS key negotiation, the SRTP master keys are determined
   out-of-band and provided to the browser through the API (e.g.,
   [RFC4568]).   SRTP does not offer any method for peer authentication.

   The following requirements apply:

   SEC-1 The application MUST be able to select how SRTP key
          negotiation is performed, either DTLS-SRTP or through the API.

   SEC-2 Where DTLS-SRTP is chosen, the application MUST be able to
          specify whether the browser is to take the client or server
          role.

   SEC-3 Where DTLS-SRTP is chosen, the application MUST be able to
          view the certificate offered by the peer.

   SEC-4 Where DTLS-SRTP is chosen, the application MUST be able to
          discover if the peer certificate is a trusted certificate for
          an identified domain.

   SEC-5 Where DTLS-SRTP is chosen, the application MUST be able to
          reject communication with peers based on information presented
          in their certificate.

   SEC-6 Where SRTP is chosen without DTLS for key negotiation, the
          application MUST be able to set SRTP parameters, including:
          the encryption, key generation and authentication algorithms;
          key and parameter size; key and salt value; key usage
          lifetime; key derivation interval; and an optional master key
          identifier.

4. Sending Peer-to-Peer Media

   RTP [RFC3550] sends real-time data - media - from a source to a sink.

   An RTP stream is the manifestation of media as packets. Most media
   is a single stream, though layered codecs (such as H.264 SVC
   [RFC6190]) can be split into multiple streams. Browser need to
   handle RTP streams in two directions: outbound from a local source,
   and inbound toward a local sink.

   The following requirements apply:

MED-1   The application MUST be able to select the UDP flow that an
         RTP stream uses.

MED-2   The application MUST be able select the UDP flow that RTCP
         for a given RTP stream uses.

MED-3   The application MUST be able to move RTP streams (or
         associated RTCP) between UDP flows without losing packets.

MED-4   The application MUST be able to specify the SSRC for RTP
         streams, both outbound and inbound.

MED-5   The application MUST be able to discover the CNAME that will
         be used for streams that it generates.

MED-6   The application MUST be able to discover the SRCNAME
         [I-D.westerlund-avtext-rtcp-sdes-srcname] that will be used
         for streams that it generates. [[Ed: conditional on
         acceptance of SRCNAME]]

MED-7   The application MUST be able to specify the RTP packet type
         that is used to identify codecs in RTP streams, both inbound
         and outbound.

MED-8   The application MUST be able to select the codecs that are
         used for outbound RTP streams and to specify the codecs that
         are present in inbound RTP streams.

MED-9   The application MUST be able to limit the bandwidth allocated
         to a single outbound RTP stream.

MED-10   The application MUST be able to specify the number of audio
         channels used for an audio stream.

MED-11   The application MUST be able to specify codec parameters
         (such as appear in SDP [RFC4566] "a=fmtp" lines).

MED-12   The application MUST be able to mark the priority of an RTP
         stream.

         This lets the browser choose appropriate packet marking using
         DiffServ and the relative sensitivity of streams to
         congestion.

MED-13   The application MUST be able to update the configuration for
         an active RTP stream.

5. DTMF Tones

Dual-Tone Multifrequency (DTMF) is commonly used by legacy telephony
applications. DTMF tones are typically interleaved/mixed with audio
streams.

The following requirements apply:

DTMF-1 The application MUST be able to insert DTMF tones into an
        audio stream.

DTMF-2 The application MUST be able to be informed of the existence
        of DTMF tone events on an audio stream.

6. IANA Considerations

This document has no IANA actions.

[RFC Editor: please remove this section prior to publication.]

7. Security Considerations

The security of your precious bodily fluids can only be achieved
through purity of essence.

8. Acknowledgments

This document takes ideas from [I-D.kaplan-rtcweb-api-reqs],
including the short title you see on each page.

9. References

9.1. Normative References

[I-D.ietf-rtcweb-rtp-usage]
          Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time
          Communication (WebRTC): Media Transport and Use of RTP",
          draft-ietf-rtcweb-rtp-usage-03 (work in progress),
          June 2012.

[I-D.ietf-rtcweb-security-arch]
          Rescorla, E., "RTCWEB Security Architecture",
          draft-ietf-rtcweb-security-arch-02 (work in progress),
          June 2012.

[I-D.westerlund-avtext-rtcp-sdes-srcname]
          Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES
          Item SRCNAME to Label Individual Sources",
          draft-westerlund-avtext-rtcp-sdes-srcname-00 (work in
          progress), October 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
          Jacobson, "RTP: A Transport Protocol for Real-Time
          Applications", STD 64, RFC 3550, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
          Norrman, "The Secure Real-time Transport Protocol (SRTP)",
          RFC 3711, March 2004.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
          (ICE): A Protocol for Network Address Translator (NAT)
          Traversal for Offer/Answer Protocols", RFC 5245,
          April 2010.

[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.

9.2. Informative References

[I-D.kaplan-rtcweb-api-reqs]
          Kaplan, H., Stratford, N., and T. Panton, "API
          Requirements for WebRTC-enabled Browsers",
          draft-kaplan-rtcweb-api-reqs-01 (work in progress),
          October 2011.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
          Description Protocol", RFC 4566, July 2006.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session
          Description Protocol (SDP) Security Descriptions for Media
          Streams", RFC 4568, July 2006.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
          "Session Traversal Utilities for NAT (STUN)", RFC 5389,
          October 2008.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using
          Relays around NAT (TURN): Relay Extensions to Session
          Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

   [RFC6190]   Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis,
               "RTP Payload Format for Scalable Video Coding", RFC 6190,
               May 2011.

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

Authors' Addresses

   Martin Thomson
   Microsoft
   3210 Porter Drive
   Palo Alto, CA 94304
   US

   Phone: +1 650-353-1925
   Email: martin.thomson@skype.net

   Bernard Aboba
   Microsoft
   One Microsoft Way
   Redmond, WA 98052
   US

   Email: bernard_aboba@hotmail.com