RTCWeb Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences

July 15, 2013

WebRTC Data Channel Protocol
draft-ietf-rtcweb-data-protocol-00.txt

Abstract

The Web Real-Time Communication (WebRTC) working group is charged to provide protocols to support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies an actual (minor) protocol for how the JS-layer DataChannel objects provide the data channels between the peers.

Status of This Memo

This Internet-Draft will expire on January 16, 2014.

Copyright Notice

Table of Contents

1. Introduction

The DataChannel Protocol is designed to provide, in the WebRTC context [I-D.ietf-rtcweb-overview], a generic transport service allowing a Web Browser to exchange generic data in a bidirectional peer to peer fashion. As discussed in [I-D.ietf-rtcweb-data-channel] the protocol uses Stream Control Transmission Protocol (SCTP) [RFC4960] encapsulated on Datagram Transport Layer Security (DTLS) [RFC6347] as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] to benefit from their already standardized transport and security features.

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 a stream identifier (0-65535).

Channel: Two Streams with the same identifier, one in each direction, that are managed together.

4. Protocol Overview

This protocol is a simple, low-overhead way to establish bidirectional Channels over an SCTP association with a consistent set of properties.

Channels are created by sending an optional DATA_CHANNEL_OPEN message on an unused Stream. There is no handshake, and the channel is available to send on as soon as the DATA_CHANNEL_OPEN has been sent. Alternatively, both sides make externally agree to a set of parameters for a Channel, in which case no DATA_CHANNEL_OPEN message is required.

To avoid glare in opening Channels, each side must use either even or odd Streams when sending a DATA_CHANNEL_OPEN message. The method used to determine which side uses odd or even is based on the underlying DTLS connection role when used in rtcweb, with the side acting as the DTLS client using even stream identifiers.

Note: There is no attempt to resolve label glare; if both sides open a Channel labelled "x" at the same time, there will be two Channels labelled "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 data being passed with an IANA-registered string, and may also be useful for homogenous applications which may create more than one type of Channel.

Data that arrives which on an unused Stream MUST be held for a period TBD until a DATA_CHANNEL_OPEN arrived for that Channel, or until the protocol stack has been told to expect data on that Stream (via external negotiation), or until [TBD - report error]. This allows for external negotiation of streams (or assumption of negotiation by cooperating applications). If a later DATA_CHANNEL_OPEN arrives that conflicts with the pre-set properties of the Channel, an error should be signaled to higher levels.

Channels are closed by an SCTP reset of the Stream.

5. Opening Handshake

The opening handshake is based on the multimedia session description exchange that happens between the browsers, typically through a Web Server acting as the signaling service.

[I-D.ietf-mmusic-sctp-sdp] defines the protocol identifier, 'DTLS/SCTP', and defines how to establish an SCTP association over DTLS using the Session Description Protocol (SDP).

The SCTP association is created with the number of streams specified by the application, and if not specified, then it SHOULD default to 16 streams.

It is recommended that additional streams be available dynamically based on [RFC6525].

6. Control Messages

Control Messages are sent to manage opening bidirectional channels.

A DATA_CHANNEL_OPEN message is sent on the Stream that is intended to be used for a new Channel, and this creates a bidirectional Channel that may be used by both sides to send data.

6.1. DATA_CHANNEL_OPEN Message

This message is sent initially on the stream used for user messages using the channel. All DATA_CHANNEL_OPEN messages MUST be sent using SCTP options for reliable in-order delivery.

```
       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 |  Channel Type |            Priority           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                    Reliability Parameter                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         Label Length          |       Protocol Length         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      \                                                               /
      |                             Label                             |
      /                                                               \
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      \                                                               /
      |                            Protocol                           |
      /                                                               \
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 1 byte (unsigned integer)

This field holds the IANA defined message type for the the DATA_CHANNEL_OPEN message. The suggested value of this field for IANA is 0x03. NOTE: values 0x00-0x02 were used in an older draft with incompatible structures. Any future incompatible message changes should define new message types.

Channel Type: 1 byte (unsigned integer)

This field specifies the type of the channel to be opened:

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

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

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

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

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED (0x02): The channel provides a partial reliable in-order bi-directional Communication

channel. 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 Javascript engine.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED (0x82): The channel provides a partial reliable unordered bi-directional Communication channel. 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 Javascript engine.

Priority: 2 bytes (integer)
The priority of the channel.

Reliability Parameter: 4 bytes (unsigned integer)
This field is ignored if a reliable channel is used. If a partial reliable channel with limited number of retransmissions is used, this field specifies the number of retransmissions. If a partial reliable channel with limited lifetime is used, this field specifies the maximum lifetime in milliseconds.

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 channel. This may be an empty string.

Protocol: Variable Length (sequence of characters)
The protocol for the channel. This may be an empty string. If used, it SHOULD be an IANA-registered protocol.

7. Procedures

7.1. Adding a Channel (in-band open)

When one side wants to add a channel using in-band declaration, it picks an unused outgoing Stream (even stream identifier for DTLS clients, or odd for DTLS servers); if no unused streams are available a negotiation to increase the number is done via [RFC6525], or failure is returned. It should also check that the other side has the same channel available, and if not then initiate an increase in the number of streams. It then sends a DATA_CHANNEL_OPEN control message on the outgoing stream.

When an DATA_CHANNEL_OPEN is received on an incoming Stream, the Stream is associated with a newly-created Channel (unless the Stream was already part of an externally-negotiated Channel). If any data had arrived on the Stream before the Open arrives and had been buffered, it is now delivered on the new Channel.

The channel_type and reliability_parameters fields of the DATA_CHANNEL_OPEN message MUST be used to set up the reverse side of the Channel so that both directions use the same options by default.

7.2. Adding a Channel (external negotiation)

When one side wants to add a channel using external negotiation, it picks a Stream. This can be done by asking the protocol to select an unused Stream (of the approriate even or odd type), or by simply telling the protocol what Stream to use. In the latter case, the application is responsible for avoiding collisions with existing Streams. If it attempts to re-use a Stream which is part of an existing Channel, the addition should fail.

In addition to choosing a Stream, the application should also inform the protocol of the options to use for sending messages. Note that there is no requirement for both sides to have the same options for an externally-negotiated stream, though typically this will be the case.

The application must now ensure in an application-specific manner that the other side will also inform the protocol that the selected Stream is to be used, and the parameters for sending data from that side.

7.3. Closing a Channel

Channels MUST be closed by and SCTP reset of the outgoing Stream. If an incoming Stream is reset by the peer, an corresponding outgoing stream reset SHOULD be issued. If the streams in both directions of a Channel are reset, the Channel is considered fully closed and the Streams are available for reuse for new channel opens.

7.4. Sending and Receiving Data

Data shall be sent using PPID's other than the Data Channel Control PPID. These PPID's should be registered with IANA via (TBD). The meaning of these data PPIDs and the format of the data shall be specific to the usage of this protocol, and typically shall be provided to the higher layers to allow proper decoding of the data.

It is RECOMMENDED that higher layers wishing to transfer large messages fragment them using PPIDs or other mechanisms to avoid monopolization of the SCTP association by the transfer of a single large message, unless a future SCTP draft relaxes this concern. If fragmented solely with PPID values, then transmission MUST occur on a reliable in-order channel. If in-band application framing is used, then other options may be possible.

For WebRTC, data PPID's for DOMStrings and binary data (and fragmentation thereof) shall be created.

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

Data may be sent immediately after sending or receiving a DATA_CHANNEL_OPEN message, or after creating an externally-negotiated Channel.

It is recommended that message size be kept within certain size bounds (TBD) as applications will not be able to support arbitrarily-large single messages.

8. Security Considerations

To be done.

9. IANA Considerations

[NOTE to RFC-Editor:

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

]

[NOTE to RFC-Editor:

The suggested values for the Payload Protocol Identifiers are tentative and to be confirmed by IANA.

]

This document defines five new SCTP Payload Protocol Identifiers (PPIDs). [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which these identifiers need to be assigned. The following values are suggested:

Value	SCTP PPID	Reference
WebRTC Control	50	[RFCXXXX]
DOMString Last	51	[RFCXXXX]
Binary Data Partial	52	[RFCXXXX]
Binary Data Last	53	[RFCXXXX]
DOMString Partial	54	[RFCXXXX]

10. Acknowledgments

The authors wish to thank Martin Thompson, Cullen Jennings, Harald Alvestrand, Peter Thatcher, Adam Bergkvist, Justin Uberti, Randall Stewart, Stefan Haekansson and many others for their invaluable comments.

11. References

11.1. 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.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[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-mmusic-sctp-sdp]
Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-04 (work in progress), June 2013.

[I-D.ietf-tsvwg-sctp-dtls-encaps]
Jesup, R., Loreto, S., Stewart, R., and M. Tuexen, "DTLS Encapsulation of SCTP Packets for RTCWEB", draft-ietf-tsvwg-sctp-dtls-encaps-00 (work in progress), February 2013.

11.2. Informational References

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications", draft-ietf-rtcweb-overview-06 (work in progress), February 2013.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Data Channels", draft-ietf-rtcweb-data-channel-04 (work in progress), February 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