Network Working Group   R. Jesup
Internet-Draft   Mozilla
Intended status: Standards Track   S. Loreto
Expires: July 8, 2015   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

Copyright Notice

Table of Contents

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.
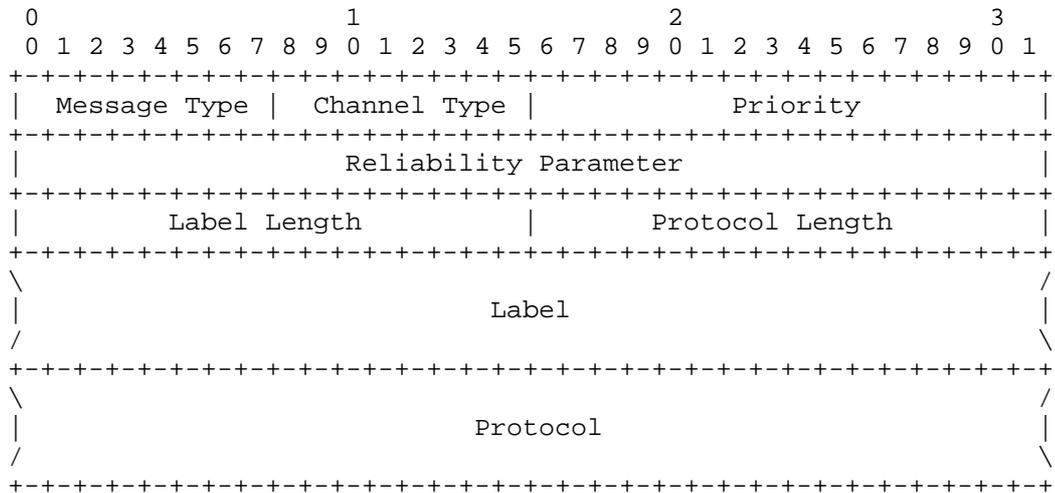
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Message Type | Channel Type |            Priority           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Reliability Parameter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Label Length          |       Protocol Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               /
|                             Label                             |
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               /
|                            Protocol                           |
/                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 1 byte (unsigned integer)
   This field holds the IANA defined message type for the
   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 milli-
seconds 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)
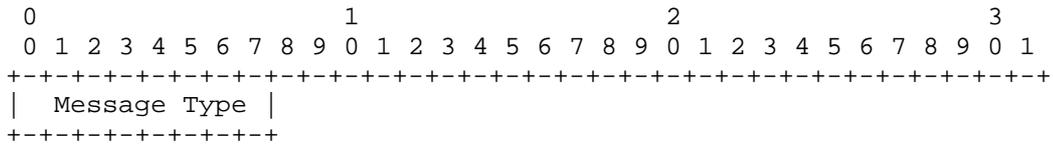The length of the protocol field in bytes.

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 end-
point 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