Stream:	Internet Engineering Task Force (IETF)
RFC:	8832
Category:	Standards Track
Published:	September 2020
ISSN:	2070-1721

Authors:	R. Jesup	S. Loreto	M. Tüxen
	Mozilla	*Ericsson*	*Münster Univ. of Appl. Sciences*

RFC 8832
WebRTC Data Channel Establishment Protocol

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 [RFC8831], a simple in-band method for opening symmetric data channels. As discussed in [RFC8831], the protocol uses the Stream Control Transmission Protocol (SCTP) [RFC4960] encapsulated in Datagram Transport Layer Security (DTLS) (described in [RFC8261]). This allows DCEP to benefit from the already standardized transport and security features of SCTP and DTLS. 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**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP?14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. 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:

- reliable or unreliable message transmission. In case of unreliable transmissions, the same level of unreliability is used.
- in-order or out-of-order message delivery.
- the priority of the data channel.
- an optional label for the data channel.
- an optional protocol for the data channel.
- 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. DCEP 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 DCEP 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 [RFC8261], 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 purpose of the protocol field is to ease cross-application interoperation ("federation") by identifying the user data being passed by means of an IANA-registered string from the "WebSocket Subprotocol Name Registry" defined in [RFC6455]. The field may be useful for homogeneous applications that may create more than one type of data channel. Note that there is no attempt to ensure uniqueness for the protocol field.

5. Message Formats

Every DCEP message starts with a one-byte field called "Message Type" that 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 initially sent using the data channel on the stream used for user messages.

```
 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 data channel to be opened. The values are managed by IANA (see Section 8.2.2):

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

DATA_CHANNEL_RELIABLE_UNORDERED (0x80): The data channel provides a reliable unordered bidirectional communication.

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT (0x01): The data channel provides a partially reliable in-order bidirectional 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 partially reliable unordered bidirectional 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 partially reliable in-order bidirectional communication. User messages might not be transmitted or retransmitted after a specified lifetime given in milliseconds in the Reliability Parameter. This lifetime starts when providing the user message to the protocol stack.

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED (0x82): The data channel provides a partially reliable unordered bidirectional communication. User messages might not be transmitted or retransmitted after a specified lifetime given in milliseconds in the Reliability Parameter. This lifetime starts when providing the user message to the protocol stack.

Priority: 2 bytes (unsigned integer)
The priority of the data channel, as described in [RFC8831].

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 partially reliable data channel with a limited number of retransmissions is used, this field specifies the number of retransmissions. If a partially reliable data channel with a 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

Table 1

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. It is sent 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 DCEP 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 PPID. Therefore, a DCEP 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 acting as the DTLS client, it **MUST** choose an even stream identifier; if the side is acting as the DTLS server, it **MUST** choose an odd one. The initiating peer 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, the receiver **MUST** close the corresponding data channel using the procedure described in [RFC8831] and **MUST NOT** send a DATA_CHANNEL_ACK message in response to the received message. This might occur if, for example, a DATA_CHANNEL_OPEN message is received on an already used stream, there are 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. 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 that message **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. In that case, the corresponding data channel **MUST** be closed, as described in [RFC8831].

7. Security Considerations

The DATA_CHANNEL_OPEN message contains two variable-length fields: the protocol and the label. A receiver must be prepared to receive DATA_CHANNEL_OPEN messages where these fields have the maximum length of 65535 bytes. Error cases such as using inconsistent lengths of fields, using unknown parameter values, or violating the odd/even rule must also be handled by closing the corresponding data channel. An end point must also be prepared for the peer to 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 [RFC8261].

For general considerations, see [RFC8826] and [RFC8827].

8. IANA Considerations

IANA has updated the reference of an already existing SCTP PPID assignment (Section 8.1) and created a new standalone registry with its own URL for DCEP (Section 8.2) containing two new registration tables (Sections 8.2.1 and 8.2.2).

8.1. SCTP Payload Protocol Identifier

This document uses an SCTP Payload Protocol Identifier (PPID) previously registered as "WebRTC Control". [RFC4960] creates the "SCTP Payload Protocol Identifiers" registry, in which this identifier was assigned. IANA has updated the PPID name from "WebRTC Control" to "WebRTC DCEP" and has updated the reference to point to this document. The corresponding date has been kept.

Therefore, this assignment now appears as follows:

Value	SCTP PPID	Reference	Date
WebRTC DCEP	50	RFC 8832	2013-09-20

Table 2

8.2. New Standalone Registry for DCEP

IANA has created the "Data Channel Establishment Protocol (DCEP) Parameters" registry. It contains the two tables provided in Sections 8.2.1 and 8.2.2.

8.2.1. New Message Type Registry

IANA has created the "Message Types" registry for DCEP to manage the one-byte "Message Type" field in DCEP messages (see Section 5). This registration table is a subregistry of the registry described in Section 8.2.

The assignment of new message types is done through an RFC Required action, as defined in [RFC8126]. Documentation of new message types **MUST** contain the following information:

1. A name for the new message type.
2. A detailed procedural description of how each message type is used with within DCEP.

The following are the inital registrations:

Name	Type	Reference
Reserved	0x00	RFC 8832
Reserved	0x01	RFC 8832
DATA_CHANNEL_ACK	0x02	RFC 8832
DATA_CHANNEL_OPEN	0x03	RFC 8832
Unassigned	0x04-0xfe	
Reserved	0xff	RFC 8832

Table 3

Note that values 0x00 and 0x01 are reserved to avoid interoperability problems, since they have been used in draft 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 has created the "Channel Types" registry for DCEP to manage the one-byte "Channel Type" field in DATA_CHANNEL_OPEN messages (see Section 5.1). This registration table is a subregistry within the registry described in Section 8.2.

The assignment of new message types is done through an RFC Required action, as defined in [RFC8126]. Documentation of new Channel Types **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.

If new Channel Types support ordered and unordered message delivery, the high-order bit **MUST** be used to indicate whether or not the message delivery is unordered.

The following are the initial registrations:

Name	Type	Reference
DATA_CHANNEL_RELIABLE	0x00	RFC 8832
DATA_CHANNEL_RELIABLE_UNORDERED	0x80	RFC 8832
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT	0x01	RFC 8832
DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED	0x81	RFC 8832
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED	0x02	RFC 8832
DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED	0x82	RFC 8832
Reserved	0x7f	RFC 8832
Reserved	0xff	RFC 8832
Unassigned	rest	

Table 4

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

9. References

9.1. Normative References

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

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC8261] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets", RFC 8261, DOI 10.17487/RFC8261, November 2017, <https://www.rfc-editor.org/info/rfc8261>.

[RFC8831] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channels", RFC 8831, DOI 10.17487/RFC8831, September 2020, <https://www.rfc-editor.org/info/rfc8831>.

9.2. Informative References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, September 2020, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, September 2020, <https://www.rfc-editor.org/info/rfc8827>.

Acknowledgements

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 Rüngeler, Randall Stewart, Peter Thatcher, Martin Thompson, Justin Uberti, and many others for their invaluable comments.

Authors' Addresses

Randell Jesup
Mozilla
United States of America
Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
FI-02420 Jorvas
Finland
Email: salvatore.loreto@ericsson.com

Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany
Email: tuexen@fh-muenster.de