       Stream Control Transmission Protocol (SCTP) Stream Reconfiguration

Abstract

   Many applications that use the Stream Control Transmission Protocol
   (SCTP) want the ability to "reset" a stream. The intention of
   resetting a stream is to set the numbering sequence of the stream
   back to 'zero' with a corresponding notification to the application
   layer that the reset has been performed. Applications requiring this
   feature want it so that they can "reuse" streams for different
   purposes but still utilize the stream sequence number so that the
   application can track the message flows. Thus, without this feature,
   a new use of an old stream would result in message numbers greater
   than expected, unless there is a protocol mechanism to "reset the
   streams back to zero". This document also includes methods for
   resetting the transmission sequence numbers, adding additional
   streams, and resetting all stream sequence numbers.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6525.

Copyright Notice

Table of Contents

1. Introduction

   Many applications that use SCTP as defined in [RFC4960] want the
   ability to "reset" a stream. The intention of resetting a stream is
   to set 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. Applications requiring this
   feature want to "reuse" streams for different purposes but still
   utilize the SSN so that the application can track the message flows.
   Thus, without this feature, a new use of an old stream would result

in message numbers greater than expected, unless there is a protocol
mechanism to "reset the streams back to zero". This document also
includes methods for resetting the Transmission Sequence Numbers
(TSNs), adding additional streams, and resetting all SSNs.

The sockets API for SCTP defined in [RFC6458] exposes the sequence
numbers used by SCTP for user message transfer. Therefore, resetting
them can be used by application writers. Please note that the
corresponding sequence number for TCP is not exposed via the sockets
API for TCP.

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. New Chunk Type

   This section defines the new chunk type that will be used to
   reconfigure streams. Table 1 illustrates the new chunk type.

```
            +------------+----------------------------------+
            | Chunk Type | Chunk Name                       |
            +------------+----------------------------------+
            | 130        | Re-configuration Chunk (RE-CONFIG) |
            +------------+----------------------------------+
```

                                Table 1

   It should be noted that the format of the RE-CONFIG chunk requires
   that the receiver ignore the chunk if it is not understood and
   continue processing all chunks that follow. This is accomplished by
   the use of the upper bits of the chunk type as described in
   Section 3.2 of [RFC4960].

   All transported integer numbers are in "network byte order", a.k.a.
   Big Endian.

3.1. RE-CONFIG Chunk

   This document adds one new chunk type to SCTP. The chunk has the
   following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type = 130    | Chunk Flags   |         Chunk Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                  Re-configuration Parameter                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/            Re-configuration Parameter (optional)              /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
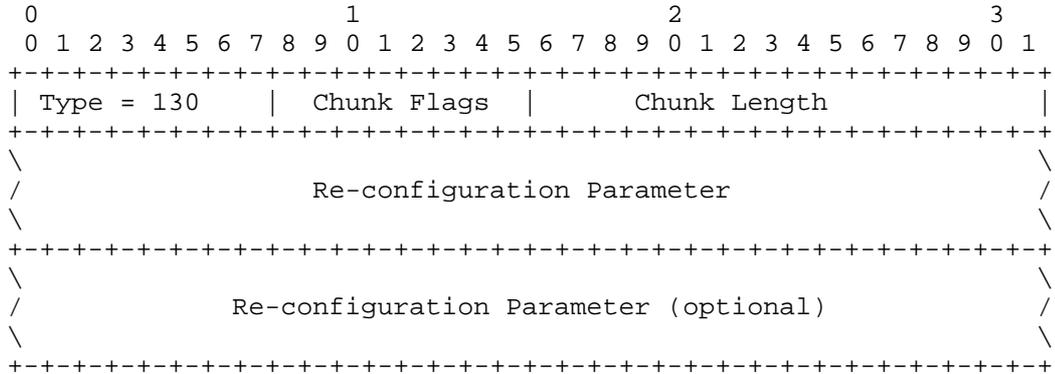
   Chunk Type: 1 byte (unsigned integer)
      This field holds the IANA-defined chunk type for the RE-CONFIG
      chunk. The value of this field is 130.

   Chunk Flags: 1 byte (unsigned integer)
      This field is set to 0 by the sender and ignored by the receiver.

   Chunk Length: 2 bytes (unsigned integer)
      This field holds the length of the chunk in bytes, including the
      Chunk Type, Chunk Flags, and Chunk Length.

   Re-configuration Parameter
      This field holds a Re-configuration Request Parameter or a
      Re-configuration Response Parameter.

   Note that each RE-CONFIG chunk holds at least one parameter and at
   most two parameters. Only the following combinations are allowed:

   1.   Outgoing SSN Reset Request Parameter.

   2.   Incoming SSN Reset Request Parameter.

   3.   Outgoing SSN Reset Request Parameter, Incoming SSN Reset Request
        Parameter.

   4.   SSN/TSN Reset Request Parameter.

   5.   Add Outgoing Streams Request Parameter.

6.   Add Incoming Streams Request Parameter.

7.   Add Outgoing Streams Request Parameter, Add Incoming Streams
     Request Parameter.

8.   Re-configuration Response Parameter.

9.   Re-configuration Response Parameter, Outgoing SSN Reset Request
     Parameter.

10. Re-configuration Response Parameter, Re-configuration Response
     Parameter.

If a sender transmits an unsupported combination, the receiver SHOULD
send an ERROR chunk with a Protocol Violation cause, as defined in
Section 3.3.10.13 of [RFC4960]).

4. New Parameter Types

This section defines the new parameter types that will be used in the
RE-CONFIG chunk. Table 2 illustrates the new parameter types.

```
+----------------+---------------------------------------+
| Parameter Type | Parameter Name                        |
+----------------+---------------------------------------+
| 13             | Outgoing SSN Reset Request Parameter  |
| 14             | Incoming SSN Reset Request Parameter  |
| 15             | SSN/TSN Reset Request Parameter       |
| 16             | Re-configuration Response Parameter   |
| 17             | Add Outgoing Streams Request Parameter|
| 18             | Add Incoming Streams Request Parameter |
+----------------+---------------------------------------+
```

Table 2
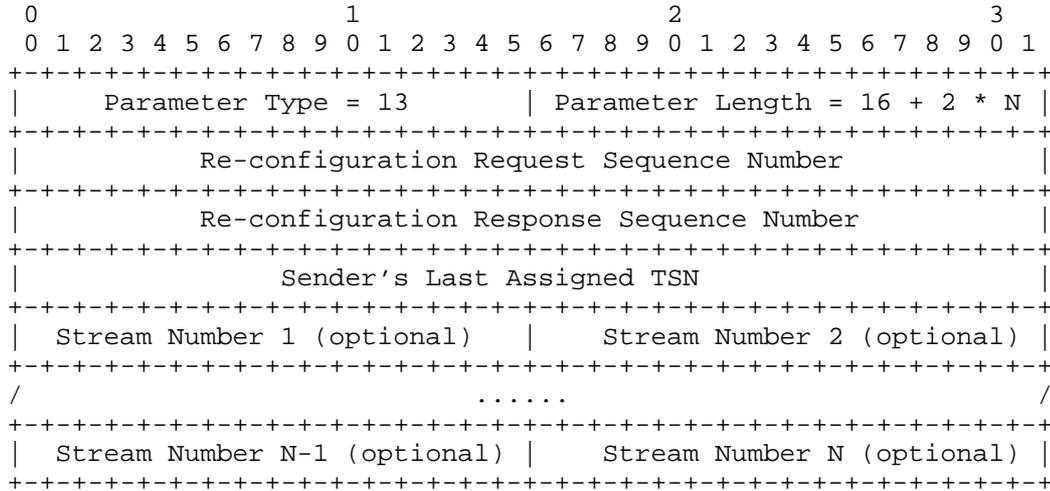
It should be noted that the parameter format requires that the
receiver stop processing the parameter and not process any further
parameters within the chunk if the parameter type is not recognized.
This is accomplished by the use of the upper bits of the parameter
type as described in Section 3.2.1 of [RFC4960].

All transported integer numbers are in "network byte order", a.k.a.
Big Endian.

4.1. Outgoing SSN Reset Request Parameter

   This parameter is used by the sender to request the reset of some or
   all outgoing streams.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 13       |  Parameter Length = 16 + 2 * N |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Request Sequence Number            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Response Sequence Number           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Sender's Last Assigned TSN                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Stream Number 1 (optional)   |    Stream Number 2 (optional) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          ......                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Stream Number N-1 (optional) |    Stream Number N (optional) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the Outgoing
      SSN Reset Request Parameter. The value of this field is 13.

   Parameter Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 16 + 2 * N, where N is the number of stream numbers
      listed.

   Re-configuration Request Sequence Number: 4 bytes (unsigned integer)
      This field is used to identify the request. It is a monotonically
      increasing number that is initialized to the same value as the
      initial TSN. It is increased by 1 whenever sending a new Re-
      configuration Request Parameter.

   Re-configuration Response Sequence Number: 4 bytes (unsigned
      integer)
      When this Outgoing SSN Reset Request Parameter is sent in response
      to an Incoming SSN Reset Request Parameter, this parameter is also
      an implicit response to the incoming request. This field then
      holds the Re-configuration Request Sequence Number of the incoming
      request. In other cases, it holds the next expected
      Re-configuration Request Sequence Number minus 1.

   Sender's Last Assigned TSN: 4 bytes (unsigned integer)
      This value holds the next TSN minus 1 -- in other words, the last
      TSN that this sender assigned.

   Stream Number 1..N: 2 bytes (unsigned integer)
      This optional field, if included, is used to indicate specific
      streams that are to be reset. If no streams are listed, then all
      streams are to be reset.

   This parameter can appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

4.2. Incoming SSN Reset Request Parameter

   This parameter is used by the sender to request that the peer reset
   some or all of its outgoing streams.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 14       |  Parameter Length = 8 + 2 * N |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Re-configuration Request Sequence Number              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Stream Number 1 (optional)   |    Stream Number 2 (optional) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          ......                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Stream Number N-1 (optional) |     Stream Number N (optional) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the Incoming
      SSN Reset Request Parameter. The value of this field is 14.

   Parameter Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 8 + 2 * N.

   Re-configuration Request Sequence Number: 4 bytes (unsigned integer)
      This field is used to identify the request. It is a monotonically
      increasing number that is initialized to the same value as the
      initial TSN. It is increased by 1 whenever sending a new Re-
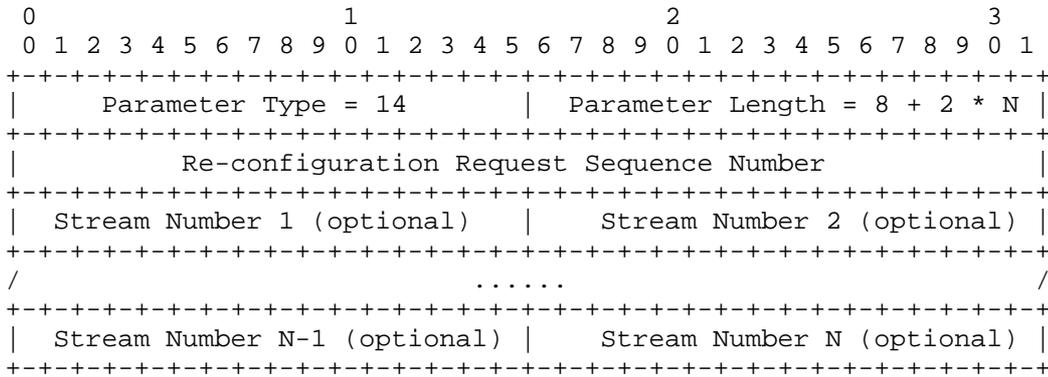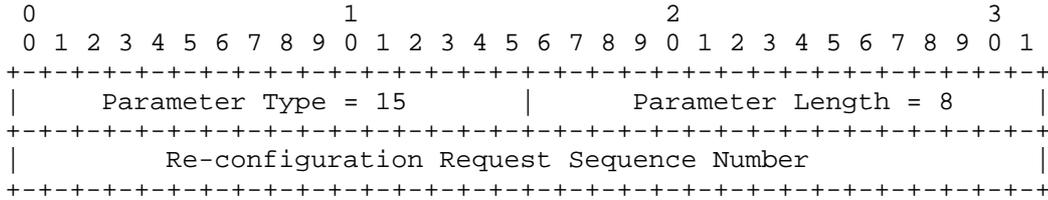   configuration Request Parameter.

   Stream Number 1..N: 2 bytes (unsigned integer)
      This optional field, if included, is used to indicate specific
      streams that are to be reset. If no streams are listed, then all
      streams are to be reset.

   This parameter can appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

4.3. SSN/TSN Reset Request Parameter

   This parameter is used by the sender to request a reset of the TSN
   and SSN numbering of all incoming and outgoing streams.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 15       |      Parameter Length = 8     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Request Sequence Number            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the SSN/TSN
      Reset Request Parameter. The value of this field is 15.

   Parameter Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 8.

   Re-configuration Request Sequence Number: 4 bytes (unsigned integer)
      This field is used to identify the request. It is a monotonically
      increasing number that is initialized to the same value as the
      initial TSN. It is increased by 1 whenever sending a new Re-
      configuration Request Parameter.

   This parameter can appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

4.4. Re-configuration Response Parameter

   This parameter is used by the receiver of a Re-configuration Request
   Parameter to respond to the request.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Parameter Type = 16      |         Parameter Length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Response Sequence Number           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Result                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Sender's Next TSN (optional)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Receiver's Next TSN (optional)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the
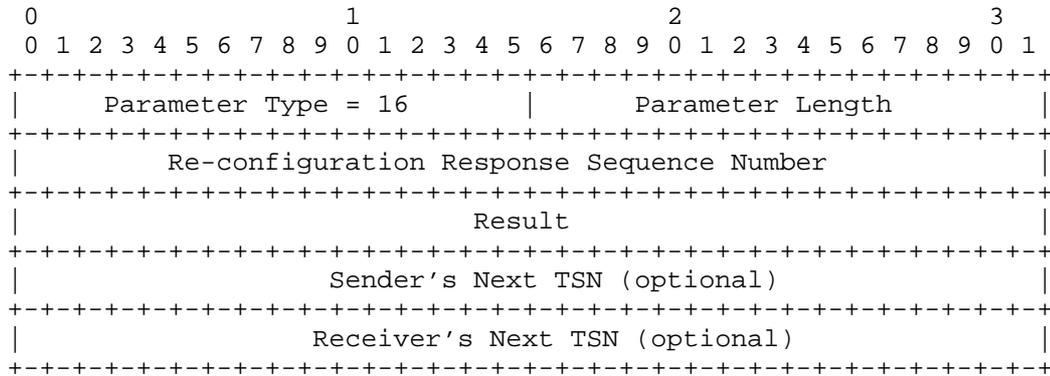   Re-configuration Response Parameter. The value of this field
      is 16.

   Parameter Type Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 12 if the optional fields are not present and 20
      otherwise.

   Re-configuration Response Sequence Number: 4 bytes (unsigned
      integer)
      This value is copied from the request parameter and is used by the
      receiver of the Re-configuration Response Parameter to tie the
      response to the request.

   Result: 4 bytes (unsigned integer)
      This value describes the result of the processing of the request.
      It is encoded as indicated in Table 3:

                +--------+----------------------------------+
                | Result | Description |
                +--------+----------------------------------+
                | 0 | Success - Nothing to do |
                | 1 | Success - Performed |
                | 2 | Denied |
                | 3 | Error - Wrong SSN |
                | 4 | Error - Request already in progress |
                | 5 | Error - Bad Sequence Number |
                | 6 | In progress |
                +--------+----------------------------------+

                                Table 3

   Sender's Next TSN: 4 bytes (unsigned integer)
      This field holds the TSN that the sender of the response will use
      to send the next DATA chunk. The field is only applicable in
      responses to SSN/TSN reset requests.

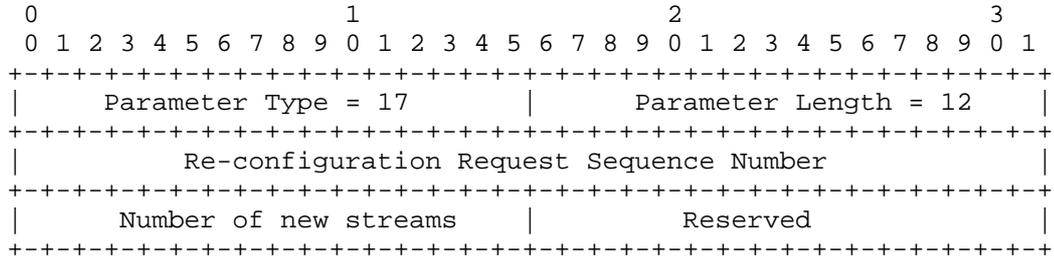   Receiver's Next TSN: 4 bytes (unsigned integer)
      This field holds the TSN that the receiver of the response must
      use to send the next DATA chunk. The field is only applicable in
      responses to SSN/TSN reset requests.

   Either both optional fields (Sender's Next TSN and Receiver's Next
   TSN) MUST be present, or no field.

   This parameter can appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

4.5. Add Outgoing Streams Request Parameter

   This parameter is used by the sender to request that an additional
   number of outgoing streams (i.e., the receiver's incoming streams) be
   added to the association.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 17       |      Parameter Length = 12    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Request Sequence Number            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Number of new streams    |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the Add
      Outgoing Streams Request Parameter. The value of this field
      is 17.

   Parameter Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 12.

   Re-configuration Request Sequence Number: 4 bytes (unsigned integer)
      This field is used to identify the request. It is a monotonically
      increasing number that is initialized to the same value as the
      initial TSN. It is increased by 1 whenever sending a new Re-
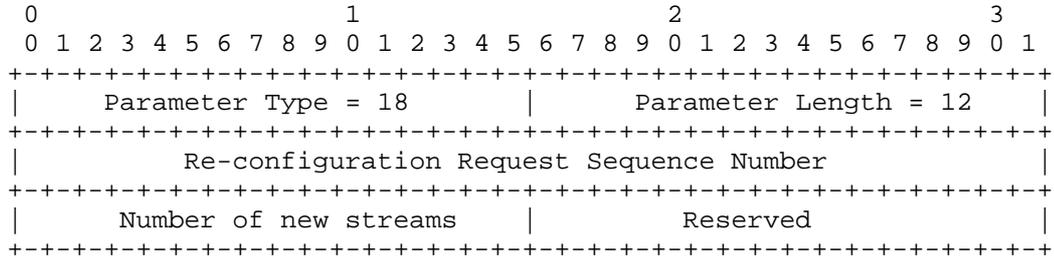   configuration Request Parameter.

   Number of new streams: 2 bytes (unsigned integer)
      This value holds the number of additional outgoing streams that
      the sender requests be added to the association. Streams are
      added in order and are consecutive; e.g., if an association has 4
      outgoing streams (0-3) and a request is made to add 3 streams,
      then the new streams will be 4, 5, and 6.

   Reserved: 2 bytes (unsigned integer)
      This field is reserved. It SHOULD be set to 0 by the sender and
      ignored by the receiver.

   This parameter MAY appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

4.6. Add Incoming Streams Request Parameter

   This parameter is used by the sender to request that the peer add an
   additional number of outgoing streams (i.e., the sender's incoming
   streams) to the association.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Parameter Type = 18       |      Parameter Length = 12    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Re-configuration Request Sequence Number            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Number of new streams    |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Parameter Type: 2 bytes (unsigned integer)
      This field holds the IANA-defined parameter type for the Add
      Incoming Streams Request Parameter. The value of this field
      is 18.

   Parameter Length: 2 bytes (unsigned integer)
      This field holds the length in bytes of the parameter; the value
      MUST be 12.

   Re-configuration Request Sequence Number: 4 bytes (unsigned integer)
      This field is used to identify the request. It is a monotonically
      increasing number that is initialized to the same value as the
      initial TSN. It is increased by 1 whenever sending a new Re-
      configuration Request Parameter.

   Number of new streams: 2 bytes (unsigned integer)
      This value holds the number of additional incoming streams that
      the sender requests be added to the association. Streams are
      added in order and are consecutive; e.g., if an association has 4
      outgoing streams (0-3) and a request is made to add 3 streams,
      then the new streams will be 4, 5, and 6.

   Reserved: 2 bytes (unsigned integer)
      This field is reserved. It SHOULD be set to 0 by the sender and
      ignored by the receiver.

   This parameter MAY appear in a RE-CONFIG chunk. This parameter MUST
   NOT appear in any other chunk type.

5. Procedures

   This section defines the procedures used by both the sender and
   receiver of a RE-CONFIG chunk. Various examples of re-configuration
   scenarios are given in Appendix A.

   One important thing to remember about SCTP streams is that they are
   uni-directional and there is no correspondence between outgoing and
   incoming streams. The procedures outlined in this section are
   designed so that the incoming side will always reset its SSN first
   (before the outgoing side), which means the re-configuration request
   must always originate from the outgoing side. These two issues have
   important ramifications upon how an SCTP endpoint might request that
   its incoming streams be reset. In effect, it must ask the peer to
   start an outgoing reset procedure and once that request is
   acknowledged let the peer actually control the reset operation.

5.1. Sender-Side Procedures

   This section describes the procedures related to the sending of
   RE-CONFIG chunks. A RE-CONFIG chunk is composed of one or two Type-
   Length-Value (TLV) parameters.

5.1.1. Sender-Side Procedures for the RE-CONFIG Chunk

   The SCTP protocol extension described in this document uses the
   Supported Extensions Parameter defined in [RFC5061] for negotiating
   the support.

   An SCTP endpoint supporting this extension MUST include the chunk
   type of the RE-CONFIG chunk in the Supported Extensions Parameter in
   either the INIT or INIT-ACK. Before sending a RE-CONFIG chunk, the
   sender MUST ensure that the peer advertised support for the
   re-configuration extension. If the chunk type of the RE-CONFIG chunk
   does not appear in the supported extension's list of chunks, then the
   sender MUST NOT send any re-configuration request to the peer, and
   any request by the application for such service SHOULD be responded
   to with an appropriate error indicating that the peer SCTP stack does
   not support the re-configuration extension.

   At any given time, there MUST NOT be more than one request in flight.
   So, if the Re-configuration Timer is running and the RE-CONFIG chunk
   contains at least one request parameter, the chunk MUST be buffered.

   After packaging the RE-CONFIG chunk and sending it to the peer, the
   sender MUST start the Re-configuration Timer if the RE-CONFIG chunk
   contains at least one request parameter. If it contains no request
   parameters, the Re-configuration Timer MUST NOT be started. This

timer MUST use the same value as SCTP's data transmission timer
(i.e., the retransmission timeout (RTO) timer) and MUST use
exponential backoff, doubling the value at every expiration. If the
timer expires, besides doubling the value, the sender MUST retransmit
the RE-CONFIG chunk, increment the appropriate error counts (for both
the association and the destination), and perform threshold
management, possibly destroying the association if SCTP
retransmission thresholds are exceeded.

5.1.2. Sender-Side Procedures for the Outgoing SSN Reset Request
        Parameter

   When an SCTP sender wants to reset the SSNs of some or all outgoing
   streams, it can send an Outgoing SSN Reset Request Parameter,
   provided that the Re-configuration Timer is not running. The
   following steps must be followed:

   A1: The sender MUST stop assigning new SSNs to new user data
        provided by the upper layer for the affected streams and queue
        it. This is because it is not known whether the receiver of the
        request will accept or deny it; moreover, a lost request might
        cause an out-of-sequence error in a stream that the receiver is
        not yet prepared to handle.

   A2: The sender MUST assign the next re-configuration request
        sequence number and MUST put it into the Re-configuration
        Request Sequence Number field of the Outgoing SSN Reset Request
        Parameter. The next re-configuration request sequence number
        MUST then be incremented by 1.

   A3: The Sender's Last Assigned TSN MUST be set to the next TSN the
        sender assigns minus 1.

   A4: If this Outgoing SSN Reset Request Parameter is sent in response
        to an Incoming SSN Reset Request Parameter, the stream numbers
        MUST be copied from the Incoming SSN Reset Request Parameter to
        the Outgoing SSN Reset Request Parameter. The Re-configuration
        Response Sequence Number of the Outgoing SSN Reset Request
        Parameter MUST be the Re-configuration Request Sequence Number
        of the Incoming SSN Reset Request Parameter. If this Outgoing
        SSN Reset Request Parameter is sent at the request of the upper
        layer and the sender requests that all outgoing streams be
        reset, stream numbers SHOULD NOT be put into the Outgoing SSN
        Reset Request Parameter. If the sender requests that only some
        outgoing streams be reset, these stream numbers MUST be placed
        in the Outgoing SSN Reset Request Parameter. The
        Re-configuration Response Sequence Number is the next expected
        Re-configuration Request Sequence Number of the peer minus 1.

   A5:   The Outgoing SSN Reset Request Parameter MUST be put into a
         RE-CONFIG Chunk. The Outgoing SSN Reset Request Parameter MAY
         be put together with either an Incoming SSN Reset Request
         Parameter or a Re-configuration Response Parameter, but not with
         both. It MUST NOT be put together with any other parameter, as
         described in Section 3.1.

   A6:   The RE-CONFIG chunk MUST be sent following the rules given in
         Section 5.1.1.

5.1.3. Sender-Side Procedures for the Incoming SSN Reset Request
        Parameter

   When an SCTP sender wants to reset the SSNs of some or all incoming
   streams, it can send an Incoming SSN Reset Request Parameter,
   provided that the Re-configuration Timer is not running. The
   following steps must be followed:

   B1:   The sender MUST assign the next re-configuration request
         sequence number and MUST put it into the Re-configuration
         Request Sequence Number field of the Incoming SSN Reset Request
         Parameter. After assigning it, the next re-configuration
         request sequence number MUST be incremented by 1.

   B2:   If the sender wants all incoming streams to be reset, stream
         numbers SHOULD NOT be put into the Incoming SSN Reset Request
         Parameter. If the sender wants only some incoming streams to be
         reset, these stream numbers MUST be filled in the Incoming SSN
         Reset Request Parameter.

   B3:   The Incoming SSN Reset Request Parameter MUST be put into a
         RE-CONFIG Chunk. It MAY be put together with an Outgoing SSN
         Reset Request Parameter but MUST NOT be put together with any
         other parameter.

   B4:   The RE-CONFIG chunk MUST be sent following the rules given in
         Section 5.1.1.

   When sending an Incoming SSN Reset Request, there is a potential that
   the peer has just reset or is in the process of resetting the same
   streams via an Outgoing SSN Reset Request. This collision scenario
   is discussed in Section 5.2.3.

5.1.4. Sender-Side Procedures for the SSN/TSN Reset Request Parameter

   When an SCTP sender wants to reset the SSNs and TSNs, it can send an
   SSN/TSN Reset Request Parameter, provided that the Re-configuration
   Timer is not running. The following steps must be followed:

   C1: The sender MUST assign the next re-configuration request
        sequence number and put it into the Re-configuration Request
        Sequence Number field of the SSN/TSN Reset Request Parameter.
        After assigning it, the next re-configuration request sequence
        number MUST be incremented by 1.

   C2: The sender has either no outstanding TSNs or considers all
        outstanding TSNs abandoned. The sender MUST queue any user
        data, suspending any new transmissions and TSN assignment until
        the reset procedure is finished by the peer either acknowledging
        or denying the request.

   C3: The SSN/TSN Reset Request Parameter MUST be put into a RE-CONFIG
        chunk. There MUST NOT be any other parameter in this chunk.

   C4: The RE-CONFIG chunk MUST be sent following the rules given in
        Section 5.1.1.

   Only one SSN/TSN Reset Request SHOULD be sent within 30 seconds,
   which is considered a maximum segment lifetime (the IP MSL).

5.1.5. Sender-Side Procedures for the Add Outgoing Streams Request
        Parameter

   When an SCTP sender wants to increase the number of outbound streams
   to which it is able to send, it may add an Add Outgoing Streams
   Request Parameter to the RE-CONFIG chunk. Upon sending the request,
   the sender MUST await a positive acknowledgment (Success) before
   using any additional stream added by this request. Note that new
   streams are added adjacent to the previous streams with no gaps.
   This means that if a request is made to add 2 streams to an
   association that already has 5 (0-4), then the new streams, upon
   successful completion, are streams 5 and 6. A new stream MUST use
   SSN 0 for its first ordered message.

5.1.6. Sender-Side Procedures for the Add Incoming Streams Request
        Parameter

   When an SCTP sender wants to increase the number of inbound streams
   to which the peer is able to send, it may add an Add Incoming Streams
   Request Parameter to the RE-CONFIG chunk. Note that new streams are
   added adjacent to the previous streams with no gaps. This means that

if a request is made to add 2 streams to an association that already
has 5 (0-4), then the new streams, upon successful completion, are
streams 5 and 6. A new stream MUST use SSN 0 for its first ordered
message.

5.1.7. Sender-Side Procedures for the Re-configuration Response
        Parameter

When an implementation receives a reset request parameter, it must
respond with a Re-configuration Response Parameter in the following
manner:

D1: The Re-configuration Request Sequence number of the incoming
     request MUST be copied to the Re-configuration Response Sequence
     Number field of the Re-configuration Response Parameter.

D2: The result of the processing of the incoming request according
     to Table 3 MUST be placed in the Result field of the
     Re-configuration Response Parameter.

D3: If the incoming request is an SSN/TSN reset request, the
     Sender's Next TSN field MUST be filled with the next TSN the
     sender of this Re-configuration Response Parameter will assign.
     For other requests, the Sender's Next TSN field, which is
     optional, MUST NOT be used.

D4: If the incoming request is an SSN/TSN reset request, the
     Receiver's Next TSN field MUST be filled with a TSN such that
     the sender of the Re-configuration Response Parameter can be
     sure it can discard received DATA chunks with smaller TSNs. The
     value SHOULD be the smallest TSN not acknowledged by the
     receiver of the request plus 2^{31}. For other requests, the
     Receiver's Next TSN field, which is optional, MUST NOT be used.

5.2. Receiver-Side Procedures

5.2.1. Receiver-Side Procedures for the RE-CONFIG Chunk

Upon reception of a RE-CONFIG chunk, each parameter within it SHOULD
be processed. If multiple parameters have to be returned, they MUST
be put into one RE_CONFIG chunk. If the received RE-CONFIG chunk
contains at least one request parameter, a selective acknowledgment
(SACK) chunk SHOULD be sent back and MAY be bundled with the
RE-CONFIG chunk. If the received RE-CONFIG chunk contains at least
one request and based on the analysis of the Re-configuration Request
Sequence Numbers this is the last received RE-CONFIG chunk (i.e., a
retransmission), the same RE-CONFIG chunk MUST to be sent back in
response, as it was earlier.

The decision to deny a re-configuration request is an administrative
decision and may be user configurable even after the association has
formed. If for whatever reason the endpoint does not wish to process
a received request parameter, it MUST send a corresponding response
parameter as described in Section 5.1.7, with an appropriate Result
field.

   Implementation Note: It is recommended that a SACK be bundled with
   any re-configuration response so that any retransmission
   processing that needs to occur can be expedited. A SACK chunk is
   not required for this feature to work, but it will in effect help
   minimize the delay in completing a re-configuration operation in
   the face of any data loss.

5.2.2. Receiver-Side Procedures for the Outgoing SSN Reset Request
        Parameter

   In the case that the endpoint is willing to perform a stream reset,
   the following steps must be followed:

   E1: If the Re-configuration Timer is running for the
        Re-configuration Request Sequence Number indicated in the
        Re-configuration Response Sequence Number field, the
        Re-configuration Request Sequence Number MUST be marked as
        acknowledged. If all Re-configuration Request Sequence Numbers
        for which the Re-configuration Timer is running are
        acknowledged, the Re-configuration Timer MUST be stopped.

   E2: If the Sender's Last Assigned TSN is greater than the cumulative
        acknowledgment point, then the endpoint MUST enter "deferred
        reset processing". In this mode, any data arriving with a TSN
        larger than the Sender's Last Assigned TSN for the affected
        stream(s) MUST be queued locally and held until the cumulative
        acknowledgment point reaches the Sender's Last Assigned TSN.
        When the cumulative acknowledgment point reaches the last
        assigned TSN, then proceed to the next step. If the endpoint
        enters "deferred reset processing", it MUST put a Re-
        configuration Response Parameter into a RE-CONFIG chunk
        indicating "In progress" and MUST send the RE-CONFIG chunk.

   E3: If no stream numbers are listed in the parameter, then all
        incoming streams MUST be reset to 0 as the next expected SSN.
        If specific stream numbers are listed, then only these specific
        streams MUST be reset to 0, and all other non-listed SSNs remain
        unchanged.

   E4: Any queued TSNs (queued at step E2) MUST now be released and
        processed normally.

   E5: A Re-configuration Response Parameter MUST be put into a
        RE-CONFIG chunk indicating successful processing.

   E6: The RE-CONFIG chunk MUST be sent after the incoming RE-CONFIG
        chunk is processed completely.

5.2.3. Receiver-Side Procedures for the Incoming SSN Reset Request
        Parameter

   In the case that the endpoint is willing to perform a stream reset,
   the following steps must be followed:

   F1: An Outgoing SSN Reset Request Parameter MUST be put into a
        RE-CONFIG chunk according to Section 5.1.2.

   F2: The RE-CONFIG chunk MUST be sent after the incoming RE-CONFIG
        chunk is processed completely.

   When a peer endpoint requests an Incoming SSN Reset Request, it is
   possible that the local endpoint has just sent an Outgoing SSN Reset
   Request on the same association and has not yet received a response.
   In such a case, the local endpoint MUST do the following:

   o If the Outgoing SSN Reset Request Parameter that was just sent
      completely overlaps the received Incoming SSN Reset Request
      Parameter, respond to the peer with an acknowledgment indicating
      that there was "Nothing to do".

   o Otherwise, process the Incoming SSN Reset Request Parameter
      normally, responding to the peer with an acknowledgment. Note
      that this case includes the situation where some of the streams
      requested overlap with the Outgoing SSN Reset Request that was
      just sent. Even in such a situation, the Incoming SSN Reset MUST
      be processed normally, even though this means that (if the
      endpoint elects to do the stream reset) streams that are already
      at SSN 0 will be reset a subsequent time.

   It is also possible that the Incoming request will arrive after the
   Outgoing SSN Reset Request just completed. In such a case, all of
   the streams being requested will be already set to 0. If so, the
   local endpoint SHOULD send back a Re-configuration Response with the
   success code "Nothing to do".

   Note that in either race condition, the local endpoint could
   optionally also perform the reset. This would result in streams that
   are already at sequence 0 being reset again to 0, which would cause
   no harm to the application but will add an extra message to the
   network.

5.2.4. Receiver-Side Procedures for the SSN/TSN Reset Request Parameter

   In the case that the endpoint is willing to perform an SSN/TSN reset,
   the following steps must be followed:

   G1: Compute an appropriate value for the Receiver's Next TSN -- the
        TSN that the peer should use to send the next DATA chunk. The
        value SHOULD be the smallest TSN not acknowledged by the
        receiver of the request plus 2^{31}.

   G2: Compute an appropriate value for the local endpoint's next TSN,
        i.e., the next TSN assigned by the receiver of the SSN/TSN reset
        chunk. The value SHOULD be the highest TSN sent by the receiver
        of the request plus 1.

   G3: The same processing as though a SACK chunk with no gap report
        and a cumulative TSN ACK of the Sender's Next TSN minus 1 were
        received MUST be performed.

   G4: The same processing as though a FWD-TSN chunk (as defined in
        [RFC3758]) with all streams affected and a new cumulative TSN
        ACK of the Receiver's Next TSN minus 1 were received MUST be
        performed.

   G5: The next expected and outgoing SSNs MUST be reset to 0 for all
        incoming and outgoing streams.

   G6: A Re-configuration Response Parameter MUST be put into a
        RE-CONFIG chunk indicating successful processing.

   G7: The RE-CONFIG chunk MUST be sent after the incoming RE-CONFIG
        chunk is processed completely.

5.2.5. Receiver-Side Procedures for the Add Outgoing Streams Request
        Parameter

   When an SCTP endpoint receives a re-configuration request adding
   additional streams, it MUST send a response parameter either
   acknowledging or denying the request. If the response is successful,
   the receiver MUST add the requested number of inbound streams to the
   association, initializing the next expected SSN to 0. The SCTP
   endpoint SHOULD deny the request if the number of streams exceeds a
   limit that should be configurable by the application.

5.2.6. Receiver-Side Procedures for the Add Incoming Streams Request
        Parameter

   When an SCTP endpoint receives a re-configuration request adding
   additional incoming streams, it MUST either send a response parameter
   denying the request or send a corresponding Add Outgoing Streams
   Request Parameter, following the rules given in Section 5.1.5. The
   SCTP endpoint SHOULD deny the request if the number of streams
   exceeds a limit that should be configurable by the application.

5.2.7. Receiver-Side Procedures for the Re-configuration Response
        Parameter

   On receipt of a Re-configuration Response Parameter, the following
   must be performed:

   H1: If the Re-configuration Timer is running for the Re-
        configuration Request Sequence Number indicated in the Re-
        configuration Response Sequence Number field, the
        Re-configuration Request Sequence Number MUST be marked as
        acknowledged. If all Re-configuration Request Sequence Numbers
        for which the Re-configuration Timer is running are
        acknowledged, the Re-configuration Timer MUST be stopped. If
        the timer was not running for the Re-configuration Request
        Sequence Number, the processing of the Re-configuration Response
        Parameter is complete.

   H2: If the Result field indicates "In progress", the timer for the
        Re-configuration Request Sequence Number is started again. If
        the timer runs out, the RE-CONFIG chunk MUST be retransmitted
        but the corresponding error counters MUST NOT be incremented.

   H3: If the Result field does not indicate successful processing, the
        processing of this response is complete.

   H4: If the request was an Outgoing SSN Reset Request, the affected
        streams MUST now be reset and all queued data should now be
        processed. The assigning of SSNs is allowed again.

   H5: If the request was an SSN/TSN Reset Request, new data MUST be
        sent from the Receiver's Next TSN, beginning with SSN 0 for all
        outgoing streams. All incoming streams MUST be reset to 0 as
        the next expected SSN. The peer will send DATA chunks starting
        with the Sender's Next TSN.

   H6: If the request was to add outgoing streams, the endpoint MUST
        add the additional streams to the association. Note that an
        implementation may allocate the memory at the time of the
        request, but it MUST NOT use the streams until the peer has
        responded with a positive acknowledgment.

6. Sockets API Considerations

   This section describes how the sockets API defined in [RFC6458] needs
   to be extended to make the features of SCTP re-configuration
   available to the application.

   Please note that this section is informational only.

6.1. Events

   When the SCTP_ASSOC_CHANGE notification is delivered and both peers
   support the extension described in this document,
   SCTP_ASSOC_SUPPORTS_RE_CONFIG should be listed in the sac_info field.

   The union sctp_notification {} is extended to contain three new
   fields: sn_strreset_event, sn_assocreset_event, and
   sn_strchange_event:

```
union sctp_notification {
  struct sctp_tlv {
    uint16_t sn_type; /* Notification type. */
    uint16_t sn_flags;
    uint32_t sn_length;
  } sn_header;
  ...
  struct sctp_stream_reset_event sn_strreset_event;
  struct sctp_assoc_reset_event sn_assocreset_event;
  struct sctp_stream_change_event sn_strchange_event;
  ...
}
```

   The corresponding sn_type values are given in Table 4.

   +-------------------------+---------------------------------------+
   | sn_type | valid field in union sctp_notification |
   +-------------------------+---------------------------------------+
SCTP_STREAM_RESET_EVENT	sn_strreset_event
SCTP_ASSOC_RESET_EVENT	sn_assocreset_event
SCTP_STREAM_CHANGE_EVENT	sn_strchange_event
   +-------------------------+---------------------------------------+

                                Table 4

These events are delivered when an incoming request was processed
successfully or the processing of an outgoing request has been
finished.

6.1.1. Stream Reset Event

The event delivered has the following structure:

```
struct sctp_stream_reset_event {
  uint16_t strreset_type;
  uint16_t strreset_flags;
  uint32_t strreset_length;
  sctp_assoc_t strreset_assoc_id;
  uint16_t strreset_stream_list[];
};
```

strreset_type: This field should be SCTP_STREAM_RESET_EVENT.

strreset_flags: This field is formed from the bitwise OR of one or
    more of the following currently defined flags:

    SCTP_STREAM_RESET_INCOMING_SSN: The stream identifiers given in
        strreset_stream_list[] refer to incoming streams of the
        endpoint.

    SCTP_STREAM_RESET_OUTGOING_SSN: The stream identifiers given in
        strreset_stream_list[] refer to outgoing streams of the
        endpoint.

    SCTP_STREAM_RESET_DENIED: The corresponding request was denied by
        the peer.

    SCTP_STREAM_RESET_FAILED: The corresponding request failed.

    At least one of SCTP_STREAM_RESET_INCOMING_SSN and
    SCTP_STREAM_RESET_OUTGOING_SSN is set. SCTP_STREAM_RESET_DENIED
    and SCTP_STREAM_RESET_FAILED are mutually exclusive. If the
    request was successful, none of these are set.

strreset_length: This field is the total length in bytes of the
    delivered event, including the header.

strreset_assoc_id: This association id field holds the identifier
    for the association. All notifications for a given association
    have the same association identifier. For one-to-one style
    sockets, this field is ignored.

   strreset_stream_list: This is the list of stream identifiers to
      which this event refers. An empty list identifies all streams as
      being reset. Depending on strreset_flags, the identifiers refer
      to incoming or outgoing streams, or both.

6.1.2. Association Reset Event

   The event delivered has the following structure:

   struct sctp_assoc_reset_event {
     uint16_t assocreset_type;
     uint16_t assocreset_flags;
     uint32_t assocreset_length;
     sctp_assoc_t assocreset_assoc_id;
     uint32_t assocreset_local_tsn;
     uint32_t assocreset_remote_tsn;
   };

   assocreset_type: This field should be SCTP_ASSOC_RESET_EVENT.

   assocreset_flags: This field is formed from the bitwise OR of one or
      more of the following currently defined flags:

      SCTP_ASSOC_RESET_DENIED: The corresponding outgoing request was
         denied by the peer.

      SCTP_ASSOC_RESET_FAILED: The corresponding outgoing request
         failed.

      SCTP_ASSOC_RESET_DENIED and SCTP_ASSOC_RESET_FAILED are mutually
      exclusive. If the request was successful, none of these are set.

   assocreset_length: This field is the total length in bytes of the
      delivered event, including the header.

   assocreset_assoc_id: This association id field holds the identifier
      for the association. All notifications for a given association
      have the same association identifier. For one-to-one style
      sockets, this field is ignored.

   assocreset_local_tsn: This field is the next TSN used by the
      endpoint.

   assocreset_remote_tsn: This field is the next TSN used by the peer.

6.1.3. Stream Change Event

   The event delivered has the following structure:

   struct sctp_stream_change_event {
     uint16_t strchange_type;
     uint16_t strchange_flags;
     uint32_t strchange_length;
     sctp_assoc_t strchange_assoc_id;
     uint16_t strchange_instrms;
     uint16_t strchange_outstrms;
   };

   strchange_type: This field should be SCTP_STREAM_CHANGE_EVENT.

   strchange_flags: This field is formed from the bitwise OR of one or
      more of the following currently defined flags:

      SCTP_STREAM_CHANGE_DENIED: The corresponding request was denied
         by the peer.

      SCTP_STREAM_CHANGE_FAILED: The corresponding request failed.

      SCTP_STREAM_CHANGE_DENIED and SCTP_STREAM_CHANGE_FAILED are
      mutually exclusive. If the request was successful, none of these
      are set.

   strchange_length: This field is the total length in bytes of the
      delivered event, including the header.

   strchange_assoc_id: This association id field holds the identifier
      for the association. All notifications for a given association
      have the same association identifier. For one-to-one style
      sockets, this field is ignored.

   strchange_instrms: The number of streams that the peer is allowed to
      use outbound.

   strchange_outstrms: The number of streams that the endpoint is
      allowed to use outbound.

6.2. Event Subscription

   Subscribing to events as described in [RFC6458] uses a setsockopt()
   call with the SCTP_EVENT socket option. This option takes the
   following structure, which specifies the association, the event type
   (using the same value found in the event type field), and an on/off
   boolean.

   struct sctp_event {
     sctp_assoc_t se_assoc_id;
     uint16_t   se_type;
     uint8_t   se_on;
   };

   The user fills in the se_type field with the same value found in the
   strreset_type field, i.e., SCTP_STREAM_RESET_EVENT. The user will
   also fill in the se_assoc_id field with either the association to set
   this event on (this field is ignored for one-to-one style sockets) or
   one of the reserved constant values defined in [RFC6458]. Finally,
   the se_on field is set with a 1 to enable the event or a 0 to disable
   the event.

6.3. Socket Options

   Table 5 describes the new socket options that make the
   re-configuration features accessible to the user. They all use
   IPPROTO_SCTP as their level.

   If a call to setsockopt() is used to issue a re-configuration request
   while the Re-configuration timer is running, setsockopt() will return
   -1, and error is set to EALREADY.

   +------------------------+-------------------------+-----+-----+
   | option name | data type | get | set |
   +------------------------+-------------------------+-----+-----+
SCTP_ENABLE_STREAM_RESET	struct sctp_assoc_value	X	X
SCTP_RESET_STREAMS	struct sctp_reset_streams		X
SCTP_RESET_ASSOC	sctp_assoc_t		X
SCTP_ADD_STREAMS	struct sctp_add_streams		X
   +------------------------+-------------------------+-----+-----+

                              Table 5

6.3.1. Enable/Disable Stream Reset (SCTP_ENABLE_STREAM_RESET)

   This option allows a user to control whether the SCTP implementation
   processes or denies incoming requests in STREAM_RESET chunks.

   The default is to deny all incoming requests.

   To set or get this option, the user fills in the following structure:

   struct sctp_assoc_value {
     sctp_assoc_t assoc_id;
     uint32_t assoc_value;
   };

   assoc_id: This parameter is ignored for one-to-one style sockets.
      For one-to-many style sockets, this parameter indicates which
      association the user is performing an action upon.

   assoc_value: This field is formed from the bitwise OR of one or more
      of the following currently defined flags:

      SCTP_ENABLE_RESET_STREAM_REQ: Process received Incoming/Outgoing
         SSN Reset Requests if this flag is set; deny them if not.

      SCTP_ENABLE_RESET_ASSOC_REQ: Process received SSN/TSN Reset
         Requests if this flag is set; deny them if not.

      SCTP_ENABLE_CHANGE_ASSOC_REQ: Process received Add Outgoing
         Streams Requests if this flag is set; deny them if not.

      The default value is !(SCTP_ENABLE_RESET_STREAM_REQ|
      SCTP_ENABLE_RESET_ASSOC_REQ|SCTP_ENABLE_CHANGE_ASSOC_REQ).

   Please note that using the option does not have any impact on
   subscribing to any related events.

6.3.2. Reset Incoming and/or Outgoing Streams (SCTP_RESET_STREAMS)

   This option allows the user to request the reset of incoming and/or
   outgoing streams.

   To set or get this option, the user fills in the following structure:

   struct sctp_reset_streams {
     sctp_assoc_t srs_assoc_id;
     uint16_t srs_flags;
     uint16_t srs_number_streams;
     uint16_t srs_stream_list[];
   };

   srs_assoc_id: This parameter is ignored for one-to-one style
      sockets. For one-to-many style sockets, this parameter indicates
      which association the user is performing an action upon.

   srs_flags: This parameter describes which class of streams is reset.
      It is formed from the bitwise OR of one or more of the following
      currently defined flags:

      * SCTP_STREAM_RESET_INCOMING

      * SCTP_STREAM_RESET_OUTGOING

   srs_number_streams: This parameter is the number of elements in the
      srs_stream_list. If it is zero, the operation is performed on all
      streams.

   srs_stream_list: This parameter contains a list of stream
      identifiers the operation is performed upon. It contains
      srs_number_streams elements. If it is empty, the operation is
      performed on all streams. Depending on srs_flags, the identifiers
      refer to incoming or outgoing streams, or both.

6.3.3. Reset SSN/TSN (SCTP_RESET_ASSOC)

   This option allows a user to request the reset of the SSN/TSN.

   To set this option, the user provides an option_value of type
   sctp_assoc_t.

   On one-to-one style sockets, the option_value is ignored. For one-
   to-many style sockets, the option_value is the association identifier
   of the association the action is to be performed upon.

6.3.4. Add Incoming and/or Outgoing Streams (SCTP_ADD_STREAMS)

   This option allows a user to request the addition of a number of
   incoming and/or outgoing streams.

   To set this option, the user fills in the following structure:

   struct sctp_add_streams {
     sctp_assoc_t sas_assoc_id;
     uint16_t sas_instrms;
     uint16_t sas_outstrms;
   };

   sas_assoc_id: This parameter is ignored for one-to-one style
      sockets. For one-to-many style sockets, this parameter indicates
      which association the user is performing an action upon.

   sas_instrms: This parameter is the number of incoming streams
      to add.

   sas_outstrms: This parameter is the number of outgoing streams
      to add.

   An endpoint can limit the number of incoming and outgoing streams by
   using the sinit_max_instreams field in the struct sctp_initmsg{} when
   issuing an SCTP_INIT socket option, as defined in [RFC6458]. An
   incoming request asking for more streams than allowed will be denied.

7. Security Considerations

   The SCTP sockets API as described in [RFC6458] exposes the sequence
   numbers of received DATA chunks to the application. An application
   might expect them to be monotonically increasing. When using the
   re-configuration extension, this might no longer be true. Therefore,
   the applications must enable this extension explicitly before it is
   used. In addition, applications must subscribe explicitly to
   notifications related to the re-configuration extension before
   receiving them.

   SCTP associations are protected against blind attackers by using
   verification tags. This is still valid when using the
   re-configuration extension. Therefore, this extension does not add
   any additional security risk to SCTP in relation to blind attackers.

   When both the SSN and TSN are reset, the maximum segment lifetime is
   used to avoid TSN wrap-around.

8. IANA Considerations

   This document (RFC 6525) is the reference for all registrations
   described in this section. The changes are described below.

8.1. A New Chunk Type

   A chunk type has been assigned by IANA. The values given in Table 1
   have been used. IANA has assigned this value from the pool of chunks
   with the upper two bits set to '10'.

   This has added a line in the "Chunk Types" registry for SCTP:

   Chunk Types

   ID Value   Chunk Type   Reference
   -----   ----------   ---------
   130   Re-configuration Chunk (RE-CONFIG)   [RFC6525]

   The registration table as defined in [RFC6096] for the chunk flags of
   this chunk type is empty.

8.2. Six New Chunk Parameter Types

   Six chunk parameter types have been assigned by IANA. It the values
   given in Table 2 have been used. IANA has assigned these values from
   the pool of parameters with the upper two bits set to '00'.

   Six additional lines in the "Chunk Parameter Types" registry for SCTP
   have been added:

   Chunk Parameter Types

   ID Value   Chunk Parameter Type   Reference
   --------   --- ---------
   13   Outgoing SSN Reset Request Parameter   [RFC6525]
   14   Incoming SSN Reset Request Parameter   [RFC6525]
   15   SSN/TSN Reset Request Parameter   [RFC6525]
   16   Re-configuration Response Parameter   [RFC6525]
   17   Add Outgoing Streams Request Parameter   [RFC6525]
   18   Add Incoming Streams Request Parameter   [RFC6525]

9. Acknowledgments

   The authors wish to thank Paul Aitken, Gorry Fairhurst, Tom Petch,
   Kacheong Poon, Irene Ruengeler, Robin Seggelmann, Gavin Shearer, and
   Vlad Yasevich 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.

   [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., Ed., "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.

   [RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission
              Protocol (SCTP) Chunk Flags Registration", RFC 6096,
              January 2011.

10.2. Informative References

   [RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V.
              Yasevich, "Sockets API Extensions for the Stream Control
              Transmission Protocol (SCTP)", RFC 6458, December 2011.

Appendix A. Examples of the Reconfiguration Procedures

   Please note that this appendix is informational only.

   The following message flows between Endpoints E-A and E-Z illustrate
   the described procedures. The time progresses in downward direction.

   The following example illustrates E-A resetting streams 1 and 2 for
   just its outgoing streams.

```
      E-A                                             E-Z
      ----------[RE-CONFIG(OUT-REQ:X/1,2)]---------->
      <------------[RE-CONFIG(RESP:X)]-------------
```

   The following example illustrates E-A resetting streams 1 and 2 for
   just its incoming streams.

```
      E-A                                             E-Z
      -----------[RE-CONFIG(IN-REQ:X/1,2)]---------->
      <--------[RE-CONFIG(OUT-REQ:Y,X/1,2)]----------
      -------------[RE-CONFIG(RESP:Y)]-------------->
```

   The following example illustrates E-A resetting all streams in both
   directions.

```
      E-A                                             E-Z
      -----[RE-CONFIG(OUT-REQ:X,Y-1|IN-REQ:X+1)]---->
      <------[RE-CONFIG(RESP:X|OUT-REQ:Y,X+1)]-------
      -------------[RE-CONFIG(RESP:Y)]-------------->
```

   The following example illustrates E-A requesting that the streams and
   TSNs be reset. At completion, E-A has the new sending TSN (selected
   by the peer) of B, and E-Z has the new sending TSN of A (also
   selected by the peer).

```
      E-A                                             E-Z
      ------------[RE-CONFIG(TSN-REQ:X)]------------>
      <-----[RE-CONFIG(RESP:X/S-TSN=A, R-TSN=B)]-----
```

   The following example illustrates E-A requesting the addition of 3
   outgoing streams.

```
      E-A                                             E-Z
      --------[RE-CONFIG(ADD_OUT_STRMS:X/3)]-------->
      <------------[RE-CONFIG(RESP:X)]-------------
```

   The following example illustrates E-A requesting the addition of 3
   incoming streams.

      E-A   E-Z
      ---------[RE-CONFIG(ADD_IN_STRMS:X/3)]-------->
      <----[RE-CONFIG(ADD_OUT_STRMS-REQ:Y,X/3)]------
      -------------[RE-CONFIG(RESP:Y)]-------------->

Authors' Addresses

   Randall R. Stewart
   Adara Networks
   Chapin, SC 29036
   USA

   EMail: randall@lakerest.net

   Michael Tuexen
   Muenster University of Applied Sciences
   Stegerwaldstr. 39
   48565 Steinfurt
   DE

   EMail: tuexen@fh-muenster.de

   Peter Lei
   Cisco Systems, Inc.
   9501 Technology Blvd.
   West Office Center
   Rosemont, IL 60018
   USA

   EMail: peterlei@cisco.com