SCTP Extensions for Dynamic Reconfiguration of IP Addresses and Enforcement of Flow and Message Limits

<draft-ietf-tsvwg-addip-sctp-02.txt>

Status of This Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026 [RFC2026]. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document describes extensions to the Stream Control Transmission Protocol (SCTP) [RFC2960] that provide methods to (1) reconfigure IP address information on an existing association and (2) request that a peer set flow limits in units of bytes or messages, either on a per-stream or per-association basis.

TABLE OF CONTENTS

1. Introduction................................................. 2
2. Conventions.................................................. 3
3. Additional Chunks and Parameters............................ 3
   3.1 New Chunk Types.......................................... 4
       3.1.1 Address/Stream Configuration Change Chunk (ASCONF)..... 4
       3.1.2 Address/Stream Configuration Acknowledgment Chunk (ASCONF-ACK)........................................ 5
   3.2 New Parameter Types...................................... 6
       3.2.1 Add IP Address...................................... 7
1. Introduction

To extend the utility and application scenarios of SCTP, this document introduces optional extensions that provide SCTP with the ability to reconfigure IP address information on an existing association or to request that the peer set flow limits in units of bytes or messages, either on a per-stream or per-association basis.

These extensions enable SCTP to be utilized in the following applications:

- Dynamic IP address reconfiguration extension: For computational or networking platforms that allow addition/removal of physical interface cards this feature can provide:

  A) a graceful method to add to the interfaces of an existing association. For IPv6 this feature allows renumbering of existing associations.

  B) a method for an endpoint to request that its peer set
its primary destination address. This can be useful when an address is about to be deleted, or when an endpoint has some predetermined knowledge about which is the preferred address to receive SCTP packets upon.

- The SCTP flow limit extension: This extension enables a receiver to request that a sender impose a byte limit on the outstanding data present on a per-stream basis.

The SCTP flow limit extension provides:

A) The ability to minimize the occurrence of a single stream monopolizing all transport level resources (e.g. a_rwnd "deadlock").

B) The ability to dynamically change the stream buffering limits as the application deems appropriate at any particular instant.

- The SCTP message limit extension: This extension enables a receiver to request that a sender impose a limit on the number of outstanding messages present on:

A) each stream, and/or

B) the whole association.

The SCTP message limit extension provides a method for minimizing the occurrence of a lack of resources needed for messages even when resources for payload data are still available. This can become important when handling a large number of short messages.

2. Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [RFC2119].

3. Additional Chunks and Parameters

This section describes the addition of two new chunks and, eight new parameters to allow:

- Dynamic addition of IP Addresses to an association.
- Dynamic deletion of IP Addresses to an association.
- A request to set the primary address the peer will use when sending to an endpoint.
- The setting of stream byte limits.
- The setting of stream message limits.
- The setting of association message limits.

Additionally, this section describes three new error causes that
support these new chunks and parameters.

3.1 New Chunk Types

This section defines two new chunk types that will be used to transfer the control information reliably. Table 1 illustrates the two new chunk types.

<table>
<thead>
<tr>
<th>Chunk Type</th>
<th>Chunk Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC1</td>
<td>Address/Stream Configuration Change Chunk (ASCONF)</td>
</tr>
<tr>
<td>0x80</td>
<td>Address Configuration Acknowledgment (ASCONF-ACK)</td>
</tr>
</tbody>
</table>

Table 1: Address/Stream Configuration Chunks

It should be noted that the ASCONF Chunk format requires the receiver to report to the sender if it does not understand the ASCONF Chunk. This is accomplished by setting the upper bits in the chunk type as described in [RFC2960] section 3.2. Note that the upper two bits in the ASCONF Chunk are set to one. As defined in [RFC2960] section 3.2, setting these upper bits in this manner will cause the receiver that does not understand this chunk to skip the chunk and continue processing, but report in an Operation Error Chunk using the ‘Unrecognized Chunk Type’ cause of error.

3.1.1 Address/Stream Configuration Change Chunk (ASCONF)

This chunk is used to communicate to the remote endpoint one of the configuration change requests that MUST be acknowledged. The information carried in the ASCONF Chunk uses the form of a Tag-Length-Value (TLV), as described in "3.2.1 Optional/Variable-length Parameter Format" in [RFC2960], for all variable parameters.
Serial Number: 32 bits (unsigned integer)

This value represents a Serial Number for the ASCONF Chunk. The valid range of Serial Number is from 0 to 4294967295 (2^32 - 1). Serial Numbers wrap back to 0 after reaching 4294967295.

Reserved: 24 bits

Reserved, set to 0 by the sender and ignored by the receiver.

Address Type: 8 bits (unsigned char)

This value determines the type of address found in the Address Bytes field. If the value is 5 then the first 4 bytes of the Address Bytes field contain an IPv4 address, in network byte order. If the value is 6 then the first 16 bytes of the Address Bytes field contain an IPv6 address, in network byte order.

Address Bytes: 16 bytes (unsigned chars)

This field contains an address which is part of the association. This field may be used by the receiver of the ASCONF to help in finding the association.

ASCONF-Request Correlation ID: 32 bits (unsigned integer)

This is an opaque integer assigned by the sender to identify each request parameter. It is in host byte order and is only meaningful to the sender. The receiver of the ASCONF Chunk will copy this 32 bit value into the ASCONF Correlation ID field of the ASCONF-ACK. The sender of the ASCONF can use this same value in the ASCONF-ACK to find which request the response is for.

ASCONF Parameter: TLV format

Each Address configuration change is represented by a TLV parameter as defined in Section 3.2. One or more requests may be present in an ASCONF Chunk.

3.1.2 Address/Stream Configuration Acknowledgment Chunk (ASCONF-ACK)

This chunk is used by the receiver of an ASCONF Chunk to acknowledge...
the reception. It carries zero or more results for any ASCONF Parameters that were processed by the receiver.

<table>
<thead>
<tr>
<th>Type = 0x80</th>
<th>Chunk Flags</th>
<th>Chunk Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCONF-Request Correlation ID #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCONF Parameter Response#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCONF-Request Correlation ID #N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCONF Parameter Response#N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Serial Number: 32 bits (unsigned integer)

This value represents the Serial Number for the received ASCONF Chunk that is acknowledged by this chunk. This value is copied from the received ASCONF Chunk.

ASCONF-Request Correlation ID: 32 bits (unsigned integer)

This value is copied from the ASCONF Correlation ID received in the ASCONF Chunk. It is used by the receiver of the ASCONF-ACK to identify which ASCONF parameter this response is associated with.

ASCONF Parameter Response: TLV format

The ASCONF Parameter Response is used in the ASCONF-ACK to report status of ASCONF processing. By default, if a responding endpoint does not include any Error Cause, a success is indicated. Thus a sender of an ASCONF-ACK MAY indicate complete success of all TLVs in an ASCONF by returning only the Chunk Type, Chunk Flags, Chunk Length (set to 8) and the Serial Number.

3.2 New Parameter Types

The eight new parameters added follow the format defined in section 3.2.1 of [RFC2960]. Table 2 describes the parameters.

<table>
<thead>
<tr>
<th>Address Configuration Parameters</th>
<th>Parameter Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add IP Address</td>
<td>0xC001</td>
</tr>
<tr>
<td>Delete IP Address</td>
<td>0xC002</td>
</tr>
<tr>
<td>Stream Byte Limit Request</td>
<td>0xC003</td>
</tr>
</tbody>
</table>
Error Cause Indication 0xC004
Set Primary Address 0xC005
Success report 0xC006
Stream Message Limit Request 0xC007
Association Message Limit Request 0xC008

Table 2: Address Configuration Parameters

3.2.1 Add IP Address

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Valid Chunk Appearance

The Add IP Address parameter may only appear in the ASCONF Chunk type.

3.2.2 Delete IP Address

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Valid Chunk Appearance

The Delete IP Address parameter may only appear in the ASCONF Chunk type.

An example TLV requesting that the IPv4 address 10.1.1.1 be added to the association would look as follows:

| +--------------------------------+ |
| | Type=0xC001 | Length = 12 |
| +--------------------------------+ |
| | Type=5       | Length = 8  |
| +----------------+---------------+ |
| | Value=0x0a010101 |
| +----------------+---------------+ |

This field contains an IPv4 or IPv6 address parameter as described in 3.3.2.1 of RFC2960. The complete TLV is wrapped within this parameter. It informs the receiver that the address specified is to be added to the existing association.

An example TLV requesting that the IPv4 address 10.1.1.1 be added to the association would look as follows:

| +--------------------------------+ |
| | Type=0xC001 | Length = 12 |
| +--------------------------------+ |
| | Type=5       | Length = 8  |
| +----------------+---------------+ |
| | Value=0x0a010101 |
| +----------------+---------------+ |

Valid Chunk Appearance

The Add IP Address parameter may only appear in the ASCONF Chunk type.
3.3.2.1 of [RFC2960]. The complete TLV is wrapped within this parameter. It informs the receiver that the address specified is to be removed from the existing association.

An example TLV deleting the IPv4 address 10.1.1.1 from an existing association would look as follows:

```
+--------------------------------+
|  Type=0xC002   | Length = 12   |
+--------------------------------+
|  Type=5        | Length = 8    |
+----------------+---------------+
|       Value=0x0a010101         |
|----------------+---------------+
```

Valid Chunk Appearance

The Delete IP Address parameter may only appear in the ASCONF Chunk type.

3.2.3 Stream Flow Limit Change

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type =0xC003           |    Length = Variable          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Number 1        |    Byte Limit 1               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Stream Number n : 16 bits (unsigned integer)

This is the stream number for which a limit is to be enforced.

Byte Limit n : 16 bits (unsigned integer)

This is the limit (in bytes) that the receiver (sending the chunk) is requesting that the sender (receiver of the chunk) enforce as the maximum amount of outstanding data permitted at any time on this stream, as per the rules in Section 4.5. Note that the value ‘0’ holds a special meaning described in Section 4.5.1

Valid Chunk Appearance

The Stream Flow Limit Change parameter may appear in the ASCONF chunk, the INIT, or the INIT-ACK chunk type. The inclusion of this parameter in the INIT or INIT-ACK can be used to indicate initial
3.2.4 Error Cause Indication

When reporting an error this response parameter is used to wrap one or more standard error causes normally found within an SCTP Operational Error or SCTP Abort (as defined in [RFC2960]). The Error Cause(s) follow the format defined in section 3.3.10 of [RFC2960].

Valid Chunk Appearance

The Error Cause Indication parameter may only appear in the ASCONF-ACK chunk type.

3.2.5 Set Primary IP Address

Address Parameter: TLV

This field contains an IPv4 or IPv6 address parameter as described in 3.3.2.1 of [RFC2960]. The complete TLV is wrapped within this parameter. It requests the receiver to mark the specified address as the primary address to send data to (see section 5.1.2 of [RFC2960]). The receiver MAY mark this as its primary upon receiving this request.

An example TLV requesting that the IPv4 address 10.1.1.1 be made the primary destination address would look as follows:

Valid Chunk Appearance
The Set Primary IP Address parameter may appear in the ASCONF Chunk, the INIT, or the INIT-ACK chunk type. The inclusion of this parameter in the INIT or INIT-ACK can be used to indicate an initial preference of primary address.

3.2.6 Success Indication

By default if a responding endpoint does not report an error for any requested TLV, a success is implicitly indicated. Thus a sender of a ASCONF-ACK MAY indicate complete success of all TLVs in an ASCONF by returning only the Chunk Type, Chunk Flags, Chunk Length (set to 8) and the Serial Number.

The responding endpoint MAY also choose to explicitly report a success for a requested TLV, by returning a success report ASCONF Parameter Response.

Valid Chunk Appearance

The Success Indication parameter may only appear in the ASCONF-ACK chunk type.

3.2.7 Stream Message Limit Change

Stream Number n : 16 bits (unsigned integer)
This is the stream number for which a limit is to be enforced.

Message Limit n : 16 bits (unsigned integer)
This is the limit (in messages) that the receiver (sending the chunk) is requesting that the sender (receiver of the chunk)
enforce as the maximum number of outstanding messages permitted at any time on this stream, as per the rules in Section 4.5. Note that the value '0' holds a special meaning described in Section 4.5.1.

Valid Chunk Appearance

The Stream Message Limit Change parameter may appear in the ASCONF chunk, the INIT, or the INIT-ACK chunk type. The inclusion of this parameter in the INIT or INIT-ACK can be used to indicate initial stream message limits.

3.2.8 Association Message Limit Change

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type =0xC008</td>
</tr>
<tr>
<td>Association Message Limit</td>
</tr>
</tbody>
</table>

Association Message Limit n : 32 bits (unsigned integer)

This is the limit (in messages) that the receiver (sending the chunk) is requesting that the sender (receiver of the chunk) enforce as the maximum number of outstanding messages permitted at any time on the association, as per the rules in Section 4.6. Note that the value 0 holds a special meaning described in Section 4.5.1

Valid Chunk Appearance

The Association Message Limit Change parameter may appear in the ASCONF Chunk, the INIT, or the INIT-ACK chunk type. The inclusion of this parameter in the INIT or INIT-ACK can be used to indicate an initial association message limit.

3.3 New Error Causes

Three new Error Causes are added to the SCTP Operational Errors, primarily for use in the ASCONF-ACK chunk.

<table>
<thead>
<tr>
<th>Cause Code</th>
<th>Value</th>
<th>Cause Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request to Delete Last Remaining IP Address.</td>
<td>0xC</td>
<td></td>
</tr>
<tr>
<td>Operation Refused Due to Resource Shortage.</td>
<td>0xD</td>
<td></td>
</tr>
<tr>
<td>Request to Delete Source IP Address.</td>
<td>0xE</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: New Error Causes
3.3.1 Error Cause: Request to Delete Last Remaining IP Address

Cause of error
---------------
Request to Delete Last Remaining IP address: The receiver of this error sent a request to delete the last IP address from its association with its peer. This error indicates that the request is rejected.

```
+--------------------------------+    +--------------------------------+    +--------------------------------+    +--------------------------------+
| Type = 0xC004 | Length = 20   | Cause=0x000C | Length = 16   | Type= 0xC002 | Length = 12   | Type=0x0005 | Length = 8    |
|Cause Code=0x000C | Cause Length=Variable |
| TLV-Copied-From-ASCONF |
/\                  /\                  /\                  |
\                     \                     \                     |
+--------------------------------+    +--------------------------------+    +--------------------------------+    +--------------------------------+

An example of a failed delete in an Error Cause TLV would look as follows in the response ASCONF-ACK message:
```

3.3.2 Error Cause: Operation Refused Due to Resource Shortage

Cause of error
---------------
This error cause is used to report a failure by the receiver to perform the requested operation due to a lack of resources. The entire TLV that is refused is copied from the ASCONF-REQ into the error cause.

```
+--------------------------------+    +--------------------------------+    +--------------------------------+    +--------------------------------+
| Type = 0xC004 | Length = 20   | Cause=0x000C | Length = 16   | Type= 0xC002 | Length = 12   | Type=0x0005 | Length = 8    |
|Cause Code=0x000D | Cause Length=Variable |
| TLV-Copied-From-ASCONF |
/\                  /\                  /\                  |
\                     \                     \                     |
+--------------------------------+    +--------------------------------+    +--------------------------------+    +--------------------------------+
```

An example of a failed addition in an Error Cause TLV would look as follows in the response ASCONF-ACK message:
3.3.3 Error Cause: Request to Delete Source IP Address

Cause of error
------------------
Request to Delete Source IP Address: The receiver of this error sent a request to delete the source IP address of the ASCONF message. This error indicates that the request is rejected.

An example of a failed delete in an Error Cause TLV would look as follows in the response ASCONF-ACK message:

IMPLEMENTATION NOTE: It is unlikely that an endpoint would source a packet from the address being deleted, unless the endpoint does not do proper source address selection.

4. Procedures

This section will lay out the specific procedures for address/stream configuration change chunk type and its processing.
4.1 ASCONF Chunk Procedures

When an endpoint has an ASCONF signaled change to be sent to the remote endpoint it should do the following:

A1) Create an ASCONF Chunk as defined in section 3.1.1. The chunk should contain all of the TLV(s) of information necessary to be sent to the remote endpoint, and unique correlation identities for each request.

A2) A serial number should be assigned to the Chunk. The serial number should be a monotonically increasing number. All serial numbers are defined to be initialized at the start of the association to the same value as the Initial TSN.

A3) If no ASCONF Chunk is outstanding (un-acknowledged) with the remote peer, AND there is less than cwnd bytes of user data outstanding, send the chunk.

A4) Start a T-4 RTO timer, using the RTO value of the selected destination address (normally the primary path; see [RFC2960] section 6.4 for details).

A5) When the ASCONF-ACK that acknowledges the serial number last sent arrives, stop the T-4 RTO timer, and clear the appropriate association and destination error counters as defined in [RFC2960] section 8.1 and 8.2.

A6) Process all of the TLVs within the ASCONF-ACK to find out particular status information returned to the various requests that were sent. Use the Correlation IDs to correlate the request and the responses.

A7) If an error response is received for a TLV parameter, all TLVs with no response before the failed TLV are considered successful if not reported. All TLVs after the failed response are considered unsuccessful unless a specific success indication is present for the parameter.

A8) If there is no response(s) to specific TLV parameter(s), and no failures are indicated, then all request(s) are considered successful.

If the T-4 RTO timer expires the endpoint should do the following:

B1) Increment the error counters and perform path failure detection on the appropriate destination address as defined in [RFC2960] section 8.1 and 8.2. Note error counters include the per destination error counter as well as the overall association error counter.

B2) Increment the association error counters and perform endpoint failure detection on the association as defined in [RFC2960] section 8.1 and 8.2. Note error counters include the per destination error counter as well as the overall association error counter.
B3) Back-off the destination address RTO timer to which the ASCONF chunk was sent by doubling the RTO timer value.

B4) Re-transmit the ASCONF Chunk last sent and if possible choose an alternate destination address (please refer to [RFC2960] section 6.4.1). An endpoint MUST NOT add new parameters to this chunk, it MUST be the same (including its serial number) as the last ASCONF sent.

B5) Restart the T-4 RTO timer. Note that if a different destination is selected, then the RTO used will be that of the new destination address.

Note: the total number of re-transmissions is limited by B2 above. If the maximum is reached, the association will fail and enter a CLOSED state (see [RFC2960] section 6.4.1 for details).

4.1.1 Congestion Control of ASCONF Chunks

In defining the ASCONF Chunk transfer procedures, it is essential that these transfers MUST NOT cause congestion within the network. To achieve this, we place these restrictions on the transfer of ASCONF Chunks:

R1) One and only one ASCONF Chunk MAY be in transit and unacknowledged at any one time. If a sender, after sending an ASCONF chunk, decides it needs to transfer another ASCONF Chunk, it MUST wait until the ASCONF-ACK Chunk returns from the previous ASCONF Chunk before sending a subsequent ASCONF. Note this restriction binds each side, so at any time two ASCONF may be in-transit on any given association (one sent from each endpoint).

R2) A ASCONF MUST NOT be sent if there is no room in the current cwnd. If there is room in the cwnd of the destination network, the Chunk may be sent regardless of the value of rwnd.

R3) A ASCONF may be bundled with any other chunk type (except other ASCONF Chunks) as long as the source address in the IP header of the packet is already a part of the association. If the ASCONF chunk is using an alternate source address as the source in the IP header, then NO other chunks may be bundled with the ASCONF chunk.

R4) A ASCONF-ACK may be bundled with any other chunk type except other ASCONF-ACKs.

R5) Both ASCONF and ASCONF-ACK chunks MUST NOT be sent in any SCTP state except ESTABLISHED.

R6) An ASCONF MUST NOT be larger than the path MTU of the destination.

R7) An ASCONF-ACK SHOULD not be larger than the path MTU. In some circumstances a ASCONF-ACK may exceed the path MTU and in such
a case IP fragmentation must be used.

If the sender of an ASCONF Chunk receives a Operational Error indicating that the ASCONF chunk type is not understood, then the sender MUST not send subsequent ASCONF Chunks to the peer. The endpoint should also inform the upper layer application that the peer endpoint does not support any of the extensions detailed in this document.

4.2 Upon reception of an ASCONF Chunk.

When an endpoint receives an ASCONF Chunk from the remote peer special procedures MAY be needed to identify the association the ASCONF Chunk is associated with. To properly find the association the following procedures should be followed:

L1) Use the source address and port number of the sender to attempt to identify the association (i.e. use the same method defined in [RFC2960] used for all other SCTP chunks ). If found proceed to rule L5.

L2) If the association is not found, use the address found in the Address Bytes field combined with the port number found in the SCTP common header. If found proceed to rule L4.

L3) If neither L1 or L2 locates the association, treat the chunk as an Out Of The Blue chunk as defined in [RFC2960].

L4) Verify that no other chunk is bundled with the ASCONF chunk. If other chunks are bundled with the ASCONF Chunk then the receiver MUST silently discard the ASCONF chunk.

L5) Follow the normal rules to validate the SCTP verification tag found in [RFC2960].

After identification and verification of the association, the following should be performed to properly process the ASCONF Chunk:

C1) Compare the value of the serial number to the value the endpoint stored in a new association variable ‘Peer-Serial-Number’. This value MUST be initialized to the Initial TSN value minus 1.

C2) If the value found in the serial number is equal to the the (‘Peer-Serial-Number’ + 1), the endpoint should:

V1) Process the TLVs contained within the Chunk performing the appropriate actions as indicated by each TLV type. The TLVs MUST be processed in order within the Chunk. For example, if the sender puts 3 TLVs in one chunk, the first TLV (the one closest to the Chunk Header) in the Chunk MUST be processed first. The next TLV in the chunk (the middle one) MUST be processed second and finally the
last TLV in the Chunk MUST be processed last.

V2) In processing the chunk, the receiver should build a response message with the appropriate error TLVs, as specified in the Parameter type bits for any ASCONF Parameter it does not understand. To indicate an unrecognized parameter, parameter type 8 as defined in in the INIT-ACK in 3.3.3 of [RFC2960] should be used. The endpoint may also use the response to carry rejections for other reasons such as resource shortages etc using the Error Cause TLV and an appropriate error condition.

Note: a positive response is implied if no error is indicated by the sender.

V3) All error responses MUST copy the ASCONF-Request Correlation ID field received in the ASCONF, from the TLV being responded to, into the ASCONF-Request Correlation ID field. The ASCONF-Request Correlation ID always precedes the request TLV.

V4) After processing the entire Chunk, it MUST send all TLVs for both unrecognized parameters and any other status TLVs inside the ASCONF-ACK chunk that acknowledges the arrival and processing of the ASCONF Chunk.

V5) Update the ‘Peer-Serial-Number’ to the value found in the serial number field.

C3) If the value found in the serial number is equal to the value stored in the ‘Peer-Serial-Number’, the endpoint should:

X1) Parse the ASCONF Chunk TLVs but the endpoint MUST NOT take any action on the TLVs parsed (since it has already performed these actions).

X2) Build a response message with the appropriate response TLVs as specified in the ASCONF Parameter type bits, for any parameter it does not understand or could not process.

X3) After parsing the entire Chunk, it MUST send any response TLV errors and status with an ASCONF-ACK chunk acknowledging the arrival and processing of the ASCONF Chunk.

X4) The endpoint MUST NOT update its ‘Peer-Serial-Number’.

IMPLEMENTATION NOTE: As an optimization a receiver may wish to save the last ASCONF-ACK for some predetermined period of time and instead of re-processing the ASCONF (with the same serial number) it may just re-transmit the ASCONF-ACK. It may wish to use the arrival of a new serial number to discard the previously saved ASCONF-ACK or any other means it may choose to expire the saved ASCONF-ACK.

C4) Otherwise, the ASCONF Chunk is discarded since it must be either a stale packet or from an attacker. A receiver of such a packet MAY log the event for security purposes.
C5) In both cases C2 and C3 the ASCONF-ACK MUST be sent back to the source address contained in the IP header of the ASCONF being responded to.

4.3 General rules for address manipulation

When building TLV parameters for the ASCONF Chunk that will add or delete IP addresses the following rules should be applied:

D1) When adding an IP address to an association, the IP address is NOT considered fully added to the association until the ASCONF-ACK arrives. This means that until such time as the ASCONF containing the add is acknowledged the sender MUST NOT use the new IP address as a source for ANY SCTP chunks besides an ASCONF Chunk. The receiver of the add IP address request may use the address as a destination immediately.

D2) After the ASCONF-ACK of an IP address add arrives, the endpoint MAY begin using the added IP address as a source address for any type of SCTP chunk.

D3a) If an endpoint receives an Error Cause TLV indicating that the IP address Add or IP address Deletion parameters was not understood, the endpoint MUST consider the operation failed and MUST NOT attempt to send any subsequent Add or Delete requests to the peer.

D3b) If an endpoint receives an Error Cause TLV indicating that the IP address Set Primary IP Address parameter was not understood, the endpoint MUST consider the operation failed and MUST NOT attempt to send any subsequent Set Primary IP Address requests to the peer.

D4) When deleting an IP address from an association, the IP address MUST be considered a valid destination address for the reception of SCTP packets until the ASCONF-ACK arrives and MUST NOT be used as a source address for any subsequent packets. This means that any datagrams that arrive before the ASCONF-ACK destined to the IP address being deleted MUST be considered part of the current association. One special consideration is that ABORT chunks arriving destined to the IP address being deleted MUST be ignored (see Section 4.3.1 for further details).

D5) An endpoint MUST NOT delete its last remaining IP address from an association. In other words if an endpoint is NOT multi-homed it MUST NOT use the delete IP address. Or if an endpoint sends multiple requests to delete IP addresses it MUST NOT delete all of the IP addresses that the peer has listed for the requester.

D6) An endpoint MUST NOT set a IP header source address for an SCTP packet holding the ASCONF Chunk to be the same as an address being deleted by the ASCONF Chunk.
D7) If a request is received to delete the last remaining IP address of a peer endpoint, the receiver MUST send an Error Cause TLV with the error cause set to the new error code ‘Request to Delete Last Remaining IP Address’. The requested delete MUST NOT be performed or acted upon, other than to send the ASCONF-ACK.

D8) If a request is received to delete an IP address which is also the source address of the IP packet which contained the ASCONF chunk, the receiver MUST reject this request. To reject the request the receiver MUST send an Error Cause TLV set to the new error code ‘Request to Delete Source IP Address’ (unless Rule D5 has also been violated, in which case the error code ‘Request to Delete Last Remaining IP Address’ is sent).

D9) If an endpoint receives an ADD IP address request and does not have the local resources to add this new address to the association, it MUST return an Error Cause TLV set to the new error code ‘Operation Refused Due to Resource Shortage’.

D10) If an endpoint receives an ‘Out of Resource’ error in response to its request to ADD an IP address to an association, it must either ABORT the association or not consider the address part of the association. In other words if the endpoint does not ABORT the association, it must consider the add attempt failed and NOT use this address and treat SCTP packets destined to the address as Out Of The Blue packets.

D11) When an endpoint receiving an ASCONF to add an IP address sends an ‘Out of Resource’ in its response, it MUST also fail any subsequent add or delete requests bundled in the ASCONF. The receiver MUST NOT reject an ADD and then accept a subsequent DELETE of an IP address in the same ASCONF Chunk. In other words, once a receiver begins failing any ADD or DELETE request, it must fail all subsequent ADD or DELETE requests contained in that single ASCONF.

D12) When an endpoint receives a request to delete an IP address that is the current primary address, it is an implementation decision as to how that endpoint chooses the new primary address.

D13) When an endpoint receives a valid request to DELETE an IP address the endpoint MUST consider the address no longer as part of the association. It MUST NOT send SCTP packets for the association to that address and it MUST treat subsequent packets received from that address as Out Of The Blue.

During the time interval between sending out the ASCONF and receiving the ASCONF-ACK it MAY be possible to receive DATA chunks out of order. The following examples illustrate these problems:

Endpoint-A
----------
ASCONF[Add-IP:X]------------------------->
//--ASCONF-ACK

Endpoint-Z
----------
In the above example we see a new IP address (X) being added to the Endpoint-A. However due to packet re-ordering in the network a new DATA chunk is sent and arrives at Endpoint-A before the ASCONF-ACK confirming the add of the address to the association.

A similar problem exists with the deletion of an IP address as follows:

```
Endpoint-A                  Endpoint-Z
----------                  ----------
/--------------------------/
|                         |
| New DATA:                |
| /                       |
|   Destination           |
| /                       |
|   IP:X                  |
|<--------------------------/   

ASCONF [DEL-IP:X]---------/-------------------->
<--------------------------------------------->
/                        /                          ASCONF-ACK
|                        |
|<--------------------------/                        |
|                                        |
```

In this example we see a DATA chunk destined to the IP:X (which is about to be deleted) arriving after the deletion is complete. For the ADD case an endpoint SHOULD consider the newly adding IP address valid for the association to receive data from during the interval when awaiting the ASCONF-ACK. The endpoint MUST NOT source data from this new address until the ASCONF-ACK arrives but it may receive out of order data as illustrated and MUST NOT treat this data as an OOTB datagram (please see [RFC2960] section 8.4). It MAY drop the data silently or it MAY consider it part of the association but it MUST NOT respond with an ABORT.

For the DELETE case, an endpoint MAY respond to the late arriving DATA packet as an OOTB datagram or it MAY hold the deleting IP address for a small period of time as still valid. If it treats the DATA packet as an OOTB the peer will silently discard the ABORT (since by the time the ABORT is sent the peer will have removed the IP address from this association). If the endpoint elects to hold the IP address valid for a period of time, it MUST NOT hold it valid longer than 2 RTO intervals for the destination being removed.

4.3.1 A special case for OOTB ABORT chunks

Another case worth mentioning is illustrated below:

```
Endpoint-A                  Endpoint-Z
----------                  ----------
/--------------------------/
|                         |
| New DATA:                |
| /                       |
|   Source IP:X           |
|<--------------------------/   
```
For this case, during the deletion of an IP address, an Abort MUST be ignored if the destination address of the Abort message is that of a destination being deleted.

4.3.2 A special case for changing an address.

In some instances the sender may only have one IP address in an association that is being renumbered. When this occurs, the sender may not be able to send to the peer the appropriate ADD/DELETE pair and use the old address as a source in the IP header. For this reason the sender MUST fill in the Address Bytes field with an address that is part of the association (in this case the one being deleted). This will allow the receiver to locate the association without using the source address found in the IP header. Such an SCTP packet MUST NOT be bundled with any other chunk.

The receiver of such an ASCONF chunk MUST NOT process the SCTP packet if any other chunks are contained inside the SCTP packet. The receiver MUST always first use the source address found in the IP header in looking up the association. The receiver should attempt to use the address found in the Address Bytes field only if the lookup fails using the source address from the IP header. The receiver MUST NOT reply to the source address of the packet in this special case, but to the new address that was added by the ASCONF (since the old address is no longer a part of the association after processing).

4.4 Setting of the primary address

A sender of this option may elect to send this combined with a deletion or addition of an address. A sender SHOULD only send a set primary request to an address that is already considered part of the association. In other words if a sender combines a set primary with an add of a new IP address the set primary will be discarded unless the add request is to be processed BEFORE the set primary (i.e. it precedes the set primary).

A request to set primary MAY also appear in a INIT or INIT-ACK chunk. This can give advice to the peer endpoint as to which of its addresses the sender of the INIT or INIT-ACK would like to be used as the primary address.

The request to set an address as the primary path is an option the receiver SHOULD perform. It is considered advice to the receiver of the best destination address to use in sending SCTP packets (in the
requester’s view). If a request arrives that asks the receiver to set an address as primary that does not exist, the receiver should NOT honor the request, leaving its existing primary address unchanged.

4.5 Stream Flow Limit and Message Limit Procedures

A stream in SCTP is an uni-directional logical channel established from one to another associated SCTP endpoint, within which all user messages are delivered in sequence except for those submitted to the un-ordered delivery service which may arrive out of sequence. Since each stream is uni-directional and no feedback mechanism exists to limit a sender, it is possible for one unique stream to monopolize all of the transport level receiver window space. The mechanism defined here attempts to alleviate this problem by allowing the receiver side to communicate to the sender a limit on how much outstanding data may be sent within a particular stream.

The procedures defined here are broken down into two sides:

- The stream receiver side or peer requesting the limit. And,
- The stream sender side or peer that MUST honor the limit request.

The receiver’s side is mainly involved with sending the request to the peer. The sender’s side is where the actual flow or message limit will be enforced. Note that the stream receiver is the endpoint that sends the ASCONF, INIT or INIT-ACK message (see Section 4.5.1), whereas the stream sender is the endpoint that receives the ASCONF, INIT or INIT-ACK message (see Section 4.5.2).

4.5.1 Stream receiver side procedures

The receiver side SCTP requests byte or message limits in response to an upper layer request. An upper layer may request, via an API interface, that a byte or message limit be imposed on all or a subset of the active streams that send data to the upper layer receiver, or that a message limit be imposed on the association. The basis on which the upper layer determines these limits is outside the scope of this document.

Any time during an association that limits are requested of the SCTP endpoint by the upper layer, the receiver side SHOULD create an ASCONF Chunk and attach to a Stream Flow Limit Change, Stream Message Limit Change, or Association Message Limit Change parameter as appropriate. These parameter types MAY also be placed in an INIT or INIT-ACK chunk at the beginning of an association to request initial values for the appropriate limits.

The Stream Flow Limit Change and Stream Message Limit Changes parameters contain a sequence of one or more pairs, each of which consists of a specific stream number, and a byte or message limit to be applied to that stream.
If the receiver wishes to remove the flow limit or message limit for a specific stream, it may do so by placing the special value ‘0’ in the Flow Limit or Message Limit field. Once acknowledged by the peer endpoint the receiver should consider the limit in place.

In the case of flow or message limits contained within an INIT chunk, any such limit is considered acknowledged with the arrival of the INIT-ACK, provided that the peer indicates that it understands the requested limit by NOT placing an ‘unrecognized parameter’ error in the INIT-ACK.

Similarly, for flow or message limits contained within an INIT-ACK chunk, any such limit is considered acknowledged with the arrival of the cookie, provided that the peer indicates that it understands the requested limit by NOT placing an ‘unrecognized parameter’ error in the cookie.

To send initial limits, ASCONF chunks are NOT bundled with the INIT or INIT-ACK. Instead the TLV is added to the variable parameters section of the INIT or INIT-ACK.

Note that the parameter type field upper two bits dictates that any parameter not understood should be skipped and reported to the sender with an Operational Error. With this in mind we make the following rules for the sender of the request:

21) If an Operational Error is received that indicates that the ‘Stream Byte Limit Request’ is not understood, the sender of the limit request MUST not send subsequent limit requests. The endpoint SHOULD also inform the upper layer application that the peer endpoint does not support this feature.

22) If an Operational Error is received that indicates that the ‘Stream Message Limit Request’ is not understood, the sender of the limit request MUST not send subsequent limit requests. The endpoint SHOULD also inform the upper layer application that the peer endpoint does not support this feature.

4.5.2 Stream Sender side procedures

When a ‘Stream Byte Limit Request’ or ‘Stream Message Limit Request’ is received the sender MUST record each limit with its appropriate stream.

After a limit is set on a stream the sender MUST obey the following rules when sending to the peer on that stream:

S1) When the upper layer application attempts to send to the peer on a stream, check
   - the number of outstanding bytes sent to that stream
     (those TSNs in queue to be sent, which the Cumulative TSN Acknowledgment has not passed, on this stream) versus the limit set
for that stream (The last received limit for this stream is henceforth termed the current limit).
- the number of outstanding messages sent on that stream (for which not all TSNs are passed by the Cumulative TSN Acknowledgment) versus the limit for this stream.

S2a) If the number of outstanding bytes is greater than or equal to the current limit, the SCTP endpoint MUST reject the request and NOT queue the data for transmit. Instead it SHOULD return an error to the sending application.

S2b) If the number of outstanding messages is greater or equal to the current limit, the SCTP endpoint MUST reject the request and NOT queue the data for transmit. Instead it SHOULD return an error to the sending application.

S3a) If the number of outstanding bytes is less than the current limit, validate that the data to be sent plus the number of outstanding bytes is smaller than or equal to this limit. If the user data plus the number of outstanding bytes is smaller than or equal to the current limit accept the data for transmit and queue the user data (increasing the number of outstanding data bytes on this stream). If the user data plus the number of outstanding bytes is larger than the current limit for this stream, the SCTP endpoint MUST reject the request and NOT queue the data for transmit and instead SHOULD return an error to the application.

S3b) If the number of outstanding messages is less than the current limit, accept the data for transmit and queue the user data (increasing the number of outstanding messages on this stream).

S4) Any time a stream limit is updated to the value of 0, consider this indication to mean no limit is in effect for this stream.

S5) Any stream number NOT mentioned in a limit request MUST be left unchanged. In other words failure to mention a stream in a limit request leaves the un-mentioned stream unchanged.

S6) If a stream limit is reduced and the stream already exceeds the stream limit, no changes are made with respect to the outstanding data. New data request MUST be rejected however, until the streams limit will allow the sending of data (rules S2 and S3 above).

NOTE: Stream limits do NOT change the underlying SCTP rwnd and its usage as defined in [RFC2960]. The association MUST still honor the rwnd when sending to the peer endpoint as defined in [RFC2960].

4.5.3 ULP considerations on the use of SCTP flow limit facility

A side-effect of rule S3 in section 4.5.2 is that an upper limit is imposed on the size of messages that may be sent to any stream where a flow limit is in place. Once a flow limit is in effect,
if the sending Upper Layer Protocol (ULP) wishes to send a message that is larger than that permitted by the imposed stream limit, the ULP will need to provide a mechanism for fragmentation and re-assembly.

This ULP mechanism is in addition to any fragmentation and re-assembly that may be provided by SCTP. It is the sole responsibility of the ULP to handle the case of a single user message being larger than the stream byte limit, if applicable.

4.6 Association Message Limit Procedures

Using the stream flow/message limit functionality described in 4.5 it is possible for a receiver to limit the sender in a way the receiver thinks is appropriate. For an overall (per association) byte based limit the receiver can make use of the rwnd field in SACK-chunks.

An overall message based limit is provided by the ‘Association Message Limit Request’. This can be useful to make better use of message oriented pools (e.g. mbufs) and to limit the delivery time for messages.

The procedures defined here are broken down into two sides:

- The receiver side or peer requesting the limit. And,
- the sender side or peer that MUST honor the limit request.

The receiver’s side is mainly involved with sending the request to the peer. The sender’s side is where the actual limitations and flow message limit will occur. Note in section 4.6.1 the receiver is the endpoint that sends the ASCONF, INIT or INIT-ACK message, in section 4.6.2 the sender side is the endpoint that receives the ASCONF, INIT or INIT-ACK message.

4.6.1 Receiver side procedures

The same rules as given in 4.5.1 for the stream limits apply to the association limit.

4.6.2 Sender side procedures

When an ‘Association Message Limit Request’ is received the sender MUST record this limit for the association.

After a limit is set for the association the sender MUST obey the following rules when sending to the peer on that stream:

S1) When the upper layer application attempts to send to the peer on a stream, check the number of outstanding messages sent on the association (for which not all TSNs are passed by the Cumulative TSN Acknowledgment) versus the limit for this association.
S2) If the number of outstanding messages is greater or equal to the current limit, the SCTP endpoint MUST reject the request and NOT queue the data for transmit. Instead it SHOULD return an error to the sending application.

S3) If the number of outstanding messages is less than the current limit, accept the data for transmit and queue the user data (increasing the number of outstanding messages on this association).

S4) Any time the association limit is updated to the value of 0, consider this indication to mean no limit is in effect for the Association.

5. Security Considerations

The ADD/DELETE of an IP address to an existing association does provide an additional mechanism by which existing associations can be hijacked. Where the attacker is able to intercept and or alter the packets sent and received in an association, the use of this feature MAY increase the ease with which an association may be overtaken. This threat SHOULD be considered when deploying a version of SCTP that makes use of this feature. The IP Authentication Header [RFC2402] SHOULD be used when the threat environment requires stronger integrity protections, but does not require confidentiality. It should be noted that in the base SCTP specification [RFC2960], if an attacker is able to intercept and or alter packets, even without this feature it is possible to hijack an existing association; please refer to Section 11 of RFC2960.

6. IANA considerations

This document defines the following new SCTP parameters, chunks and errors:

- Two new chunk types,
- Eight parameter types, and
- Three new SCTP error causes.

7. Acknowledgments

The authors wish to thank Jon Berger, John Loughney, Ivan Arias Rodriguez, Marshall Rose, and Chip Sharp for their invaluable comments.

8. Authors’ Addresses

Randall R. Stewart                      Tel: +1-815-477-2127
Cisco Systems, Inc.                     EMail: rrs@cisco.com
8745 W. Higgins Road, Suite 200
Chicago, Ill  60631
USA

Micheal A. Ramalho                      Tel: +1-732-809-0188
9. References


This Internet Draft expires in 6 months from May, 2001