Network Working Group
Internet-Draft
Expires: July 31, 2005

R. Stewart
M. Ramalho
Cisco Systems, Inc.
Q. Xie
Motorola, Inc.
M. Tuexen
Univ. of Applied Sciences Muenster
P. Conrad
University of Delaware
January 28, 2005

Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration
draft-ietf-tsvwg-addip-sctp-10.txt

Abstract

This document describes extensions to the Stream Control Transmission Protocol (SCTP) [RFC2960] that provides a method to reconfigure IP address information on an existing association.

Table of Contents

1. Introduction

To extend the utility and application scenarios of SCTP, this document introduces optional extensions that provide SCTP with the ability to:

1. reconfigure IP address information on an existing association.
2. set the remote primary path.
3. exchange adaptation layer information during association setup.

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

1. For computational or networking platforms that allow addition/removal of physical interface cards this feature can provide a graceful method to add to the interfaces of an existing association. For IPv6 this feature allows renumbering of existing associations.
2. This provides 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.
3. This feature can be used to extend the usability of SCTP without modifying it by allowing endpoints to exchange some information during association setup.

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 RFC2119 [2].

3. Additional Chunks and Parameters

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

- o Dynamic addition of IP Addresses to an association.
- o Dynamic deletion of IP Addresses from an association.
- o A request to set the primary address the peer will use when sending to an endpoint.

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.

Chunk Type	Chunk Name	
0xC1	Address Configuration Change Chunk	(ASCONF)
0x80	Address Configuration Acknowledgment	(ASCONF-ACK)

Table 1: Address 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 [6] section 3.2. Note that the upper two bits in the ASCONF Chunk are set to one. As defined in RFC2960 [6] 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 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 Type-Length-Value (TLV), as described in "3.2.1 Optional/Variable-length Parameter Format" in RFC2960 [6], for all variable parameters.

```
        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 = 0xC1   |  Chunk Flags  |      Chunk Length             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       Serial Number                           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                    Address Parameter                          |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                     ASCONF Parameter #1                       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \                                                               \
       /                             ....                              /
       \                                                               \
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                     ASCONF Parameter #N                       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

Address Parameter : 8 or 20 bytes (depending on type)

This field contains an address parameter, either IPv6 or IPv4, from RFC2960 [6]. The address is an address of the sender of the ASCONF chunk, the address MUST be considered part of the association by the peer endpoint (the receiver of the ASCONF chunk). This field may be used by the receiver of the ASCONF to help in finding the association. This parameter MUST be present in every ASCONF message i.e. it is a mandatory TLV parameter.

Note the host name address parameter is NOT allowed and MUST be ignored if received in any ASCONF message.

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 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.

```
        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 = 0x80   |  Chunk Flags  |      Chunk Length             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       Serial Number                           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                 ASCONF Parameter Response#1                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \                                                               \
       /                             ....                              /
       \                                                               \
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                 ASCONF Parameter Response#N                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 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 six new parameters added follow the format defined in section 3.2.1 of RFC2960 [6]. Table 2 and 3 describes the parameters.

Address Configuration Parameters	Parameter Type
Set Primary Address	0xC004
Adaption Layer Indication	0xC006

Table 2: Parameters that can be used in INIT/INIT-ACK chunk

Address Configuration Parameters	Parameter Type
Add IP Address	0xC001
Delete IP Address	0xC002
Set Primary Address	0xC004

Table 2: Parameters used in ASCONF Parameter

Address Configuration Parameters	Parameter Type
Error Cause Indication	0xC003
Success Indication	0xC005

Table 3: Parameters used in ASCONF Parameter Response

Any parameter that appears where it is not allowed (for example a 0xC002 parameter appearing within an INIT or INIT-ACK) MAY be responded to with an ABORT by the receiver of the invalid parameter.

3.2.1 Add IP Address

```
 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 = 0xC001          |    Length = Variable          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ASCONF-Request Correlation ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Address Parameter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ASCONF-Request Correlation ID: 32 bits

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 Response Correlation ID field of the ASCONF-ACK response parameter. The sender of the ASCONF can use this same value in the ASCONF-ACK to find which request the response is for.

Address Parameter: TLV

This field contains an IPv4 or IPv6 address parameter as described in 3.3.2.1 of RFC2960 [6]. 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 = 16   |
        +--------------------------------+
        |       C-ID = 0x01023474        |
        +--------------------------------+
        |  Type=5         | Length = 8    |
        +----------------+---------------+
        |       Value=0x0a010101         |
        +----------------+---------------+
```

Valid Chunk Appearance

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

3.2.2 Delete IP Address

```
    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 =0xC002           |    Length = Variable          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               ASCONF-Request Correlation ID                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Address Parameter                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ASCONF-Request Correlation ID: 32 bits

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 Response Correlation ID field of the ASCONF-ACK response parameter. The sender of the ASCONF can use this same value in the ASCONF-ACK to find which request the response is for.

Address Parameter: TLV

This field contains an IPv4 or IPv6 address parameter as described in 3.3.2.1 of RFC2960 [6]. 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 = 16   |
+--------------------------------+
|       C-ID = 0x01023476         |
+--------------------------------+
|  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 Error Cause Indication

```
 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 = 0xC003              |      Length = Variable        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ASCONF-Response Correlation ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Error Cause(s) or Return Info on Success         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ASCONF-Response Correlation ID: 32 bits

This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32 bit value from the ASCONF-Request Correlation ID into the ASCONF Response Correlation ID field so the peer can easily correlate the request to this response.

Error Cause(s): TLV(s)

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 [6]). The Error Cause(s) follow the format defined in section 3.3.10 of RFC2960 [6].

Valid Chunk Appearance

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

3.2.4 Set Primary IP Address

```
 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 =0xC004           |     Length = Variable         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ASCONF-Request Correlation ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Address Parameter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ASCONF-Request Correlation ID: 32 bits

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 Response Correlation ID field of the ASCONF-ACK response parameter. The sender of the ASCONF can use this same value in the ASCONF-ACK to find which request the response is for.

Address Parameter: TLV

This field contains an IPv4 or IPv6 address parameter as described in 3.3.2.1 of RFC2960 [6]. 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 [6]). 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:

```
+--------------------------------+
|  Type=0xC004    | Length = 16  |
+--------------------------------+
|       C-ID = 0x01023479        |
+--------------------------------+
|  Type=5         | Length = 8   |
+----------------+---------------+
|       Value=0x0a010101         |
+----------------+---------------+
```

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.

3.2.5 Success Indication

```
 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 = 0xC005          |      Length = 8               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ASCONF-Response Correlation ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

ASCONF-Response Correlation ID: 32 bits

This is an opaque integer assigned by the sender to identify each request parameter. The receiver of the ASCONF Chunk will copy this 32 bit value from the ASCONF-Request Correlation ID into the ASCONF Response Correlation ID field so the peer can easily correlate the request to this response.

Valid Chunk Appearance

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

3.2.6 Adaptation Layer Indication

```
 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 =0xC006           |    Length = 8                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Adaption Code point                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This parameter is specified for the communication of peer upper layer protocols. It is envisioned to be used for flow control and other adaptation layers that require an indication to be carried in the INIT and INIT-ACK. Each adaptation layer that is defined that wishes to use this parameter MUST specify a an adaption code point in an appropriate RFC defining its use and meaning. This parameter SHOULD NOT be examined by the receiving SCTP implementation and should be passed opaquely to the upper layer protocol.

Valid Chunk Appearance

The Adaptation Layer Indication parameter may appear in INIT or INIT-ACK chunk and SHOULD be passed to the receivers upper layer protocol. This parameter MUST NOT appear in a ASCONF chunk.

3.3 New Error Causes

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

Cause Code Value	Cause Code
0x0100	Request to Delete Last Remaining IP Address.
0x0101	Operation Refused Due to Resource Shortage.
0x0102	Request to Delete Source IP Address.
0x0103	Association Aborted due to illegal ASCONF-ACK
0x0104	Request refused - no authorization.

Table 4: 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.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=0x0100         |      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:

```
           +--------------------------------+
           | Type = 0xC003  | Length = 28   |
           +----------------+---------------+
           |       C-ID = 0x01023476        |
           +--------------------------------+
           |  Cause=0x0100  | Length = 20   |
           +----------------+---------------+
           |  Type= 0xC002  | Length = 16   |
           +----------------+---------------+
           |       C-ID = 0x01023476        |
           +--------------------------------+
           |   Type=0x0005  | Length = 8    |
           +----------------+---------------+
           |       Value=0x0A010101         |
           +----------------+---------------+
```

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 into the error cause.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     Cause Code=0x0101         |      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:

```
           +--------------------------------+
           | Type = 0xC003  | Length = 28   |
           +--------------------------------+
           |       C-ID = 0x01023474        |
           +--------------------------------+
           |  Cause=0x0101  | Length = 20   |
           +----------------+---------------+
           |  Type=0xC001   | Length = 16   |
           +--------------------------------+
           |       C-ID = 0x01023474        |
           +--------------------------------+
           |  Type=0x0005   | Length = 8    |
           +----------------+---------------+
           |       Value=0x0A010101         |
           +----------------+---------------+
```

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.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     Cause Code=0x0102         |      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:

```
+--------------------------------+
| Type = 0xC003  | Length = 28   |
+--------------------------------+
|       C-ID = 0x01023476        |
+--------------------------------+
|  Cause=0x0102  | Length = 20   |
+----------------+---------------+
|  Type=0xC002   | Length = 16   |
+----------------+---------------+
|       C-ID = 0x01023476        |
+--------------------------------+
|   Type=0x0005  | Length = 8    |
+----------------+---------------+
|       Value=0x0A010101         |
+----------------+---------------+
```

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.

3.3.4 Error Cause: Association Aborted due to illegal ASCONF-ACK

This error is to be included in an ABORT that is generated due to the reception of an ASCONF-ACK that was not expected but is larger than the current sequence number (see Section 4.3 Rule D0). Note that a sequence number is larger than the last acked sequence number if it is either the next sequence or no more than 2^^31-1 greater than the current sequence number.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=0x0103         |      Cause Length=4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.3.5 Error Cause: Request refused - no authorization.

Cause of error

This error cause may be included to reject a request based on local security policies.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |     Cause Code=0x0104         |      Cause Length=Variable    |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \                     TLV-Copied-From-ASCONF                    /
       /                                                               \
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4. Procedures

This section will lay out the specific procedures for address 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. The serial number MUST be initialized at the start of the association to the same value as the Initial TSN and every time a new ASCONF chunk is created it is incremented by one after assigning the serial number to the newly created chunk .

A3) If no ASCONF Chunk is outstanding (un-acknowledged) with the remote peer, 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 [6] 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 [6] 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.

A9) If the peer responds to an ASCONF with an ERROR chunk reporting that it did not recognize the ASCONF chunk type, the sender of the ASCONF MUST NOT send any further ASCONF chunks and MUST stop its T-4 timer.

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 [6] section 8.1 and 8.2.
B2) Increment the association error counters and perform endpoint failure detection on the association as defined in RFC2960 [6] section 8.1 and 8.2.
B3) Back-off the destination address RTO value to which the ASCONF chunk was sent by doubling the RTO timer value.
Note: The RTO value is used in the setting of all timer types for SCTP. Each destination address has a single RTO estimate.
B4) Re-transmit the ASCONF Chunk last sent and if possible choose an alternate destination address (please refer to RFC2960 [6] 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 into the CLOSED state (see RFC2960 [6] 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) An ASCONF may be bundled with any other chunk type (except other ASCONF Chunks).
R3) An ASCONF-ACK may be bundled with any other chunk type except other ASCONF-ACKs.
R4) Both ASCONF and ASCONF-ACK chunks MUST NOT be sent in any SCTP state except ESTABLISHED, SHUTDOWN-PENDING, SHUTDOWN-RECEIVED and SHUTDOWN-SENT.
R5) An ASCONF MUST NOT be larger than the path MTU of the destination.

R6) An ASCONF-ACK SHOULD not be larger than the path MTU. In some circumstances an ASCONF-ACK may exceed the path MTU and in such a case IP fragmentation should be used to transmit the chunk.

If the sender of an ASCONF Chunk receives an 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 [6] used for all other SCTP chunks). If found proceed to rule L4.
L2) If the association is not found, use the address found in the Address Parameter TLV 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 [6].
L4) Follow the normal rules to validate the SCTP verification tag found in RFC2960 [6].

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 ('Peer-Serial-Number' + 1), the endpoint MUST:

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, cause type 8 as defined in the ERROR in 3.3.10.8 of RFC2960 [6] 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 responses MUST copy the ASCONF-Request Correlation ID field received in the ASCONF parameter, from the TLV being responded to, into the ASCONF-Request Correlation ID field in the response parameter.

V4) After processing the entire Chunk, the receiver of the ASCONF 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'.

Note: the response to the retransmitted ASCONF MUST be the same as the original response. This MAY mean an implementation must keep state in order to respond with the same exact answer (including resource considerations that may have made the implementation refuse a request).

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:

D0) If an endpoint receives an ASCONF-ACK that is greater than or equal to the next serial number to be used but no ASCONF chunk is outstanding the endpoint MUST ABORT the association. Note that a sequence number is greater than if it is no more than 2^^31-1 larger than the current sequence number (using serial arithmetic).

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 packet except on carrying 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 without an add IP address preceding the delete parameter in the ASCONF chunk. 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 an 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 since its peer will 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                                     Endpoint-Z
    ----------                                     ----------
    ASCONF[Add-IP:X]------------------------------>
                                            /--ASCONF-ACK
                                           /
                                 /--------/---New DATA:
                                /        /    Destination
           <-------------------/        /     IP:X
                                       /
           <--------------------------/
```

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 [6] 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           \
                           \
    ASCONF-REQ[DEL-IP:X]----\------------------>
                             \        /---------ASCONF-ACK
                              \      /
                               \----/-----------> OOTB
    (Ignored <---------------------/-------------ABORT
     by rule D4)                  /
           <---------------------/
```

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 Parameter 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.

The receiver of such a chunk 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 reply to the source address of the packet in this case which is 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 an 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 prefer 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 requesters 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.

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 [3] 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 [6], 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].

Future versions of this document may require use of purpose built keys (pbk). A purpose built key mechanism assure that the endpoint adding or deleting IP addresses is most likely the same endpoint that the association started with aka the sender of the INIT or INIT-ACK.

6. IANA considerations

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

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

This document also defines a Adaption code point. The adaption code point is a 32 bit interger that is assigned by IANA through an IETF Consensus action as defined in RFC2434 [4].

7. Acknowledgments

The authors wish to thank Jon Berger, Greg Kendall, Seok Koh, Peter Lei, John Loughney, Ivan Arias Rodriguez, Renee Revis, Marshall Rose, and Chip Sharp for their invaluable comments.

The authors would also like to give special mention to Maria-Carmen Belinchon and Ian Rytina for there early contributions to this document and their thoughtful comments.

8. References

[1] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[4] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[5] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[6] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

Authors' Addresses

Randall R. Stewart
Cisco Systems, Inc.
4875 Forest Drive
Suite 200
Columbia, SC 29206
USA

Phone:
Email: rrs@cisco.com

Michael A. Ramalho
Cisco Systems, Inc.
1802 Rue de la Porte
Wall Township, NJ 07719-3784
USA

Phone: +1.732.449.5762
Email: mramalho@cisco.com

Qiaobing Xie
Motorola, Inc.
1501 W. Shure Drive, #2309
Arlington Heights, IL 60004
USA

Phone: +1-847-632-3028
Email: qxie1@email.mot.com

Michael Tuexen
Univ. of Applied Sciences Muenster
Stegerwaldstr. 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Phillip T. Conrad
University of Delaware
Department of Computer and Information Sciences
Newark, DE 19716
US

Phone: +1 302 831 8622
Email: conrad@acm.org
URI: http://www.cis.udel.edu/~pconrad

Appendix A. Abstract Address Handling

A.1 General remarks

The following text provides a working definition of the endpoint notion to discuss address reconfiguration. It is not intended to restrict implementations in any way, its goal is to provide as set of definitions only. Using these definitions should make a discussion about address issues easier.

A.2 Generalized endpoints

A generalized endpoint is a pair of a set of IP addresses and a port number at any given point of time. The precise definition is as follows:

A generalized endpoint gE at time t is given by

gE(t) = ({IP1, ..., IPn}, Port)

where {IP1, ..., IPn} is a non empty set of IP addresses.

Please note that the dynamic addition and deletion of IP-addresses described in this document allows the set of IP-addresses of a generalized endpoint to be changed at some point of time. The port number can never be changed.

The set of IP addresses of a generalized endpoint gE at a time t is defined as

Addr(gE)(t) = {IP1, ..., IPn}

if gE(t) = ({IP1, ..., IPn}, Port) holds at time t.

The port number of a generalized endpoint gE is defined as

Port(gE) = Port

if gE(t) = ({IP1, ..., IPn}, Port) holds at time t.

There is one fundamental rule which restricts all generalized endpoints:

For two different generalized endpoints gE' and gE'' with the same port number Port(gE') = Port(gE'') the address sets Addr(gE')(t) and Addr(gE'')(t) must be disjoint at every point of time.

A.3 Associations

Associations consists of two generalized endpoints and the two address sets known by the peer at any time. The precise definition is as follows:

An association A between to different generalized endpoints gE' and gE'' is given by

A = (gE', S', gE'', S'')

where S'(t) and S''(t) are set of addresses at any time t such that S'(t) is a non-empty subset of Addr(gE')(t) and S''(t) is a non-empty subset of Addr(gE'')(t).

If A = (gE', S', gE'', S'') is an association between the generalized endpoints gE' and gE'' the following notion is used:

Addr(A, gE') = S'   and Addr(A, gE'') = S''.

If the dependency on time is important the notion Addr(A, gE')(t) = S'(t) will be used.

If A is an association between gE' and gE'' then Addr(A, gE') is the subset of IP addresses of gE' which is known by gE'' and used by gE'.

Association establishment between gE' and gE'' can be seen as:

1. gE' and gE'' do exist before the association.
2. If an INIT has to be send from gE' to gE'' address scoping rules and other limitations are applied to calculate the subset S' from Addr(gE'). The addresses of S' are included in the INIT chunk.
3. If an INIT-ACK has to be send from gE'' to gE' address scoping rules and other limitations are applied to calculate the subset S'' from Addr(gE''). The addresses of S'' are included in the INIT-ACK chunk.
4. After the handshake the association A = (gE', S', gE'', S'') has been established.
5. Right after the association establishment Addr(A, gE') and Addr(A, gE'') are the addresses which have been seen on the wire during the handshake.

A.4 Relationship with RFC 2960

RFC2960 [6] defines the notion of an endpoint. This subsection will show that these endpoints are also (special) generalized endpoints.

RFC2960 [6] has no notion of address scoping or other address

handling limitations and provides no mechanism to change the addresses of an endpoint.

This means that an endpoint is simply a generalized endpoint which does not depend on the time. Neither the Port nor the address list changes.

During association setup no address scoping rules or other limitations will be applied. This means that for an association A between two endpoints gE' and gE'' the following is true:

Addr(A, gE') = Addr(gE') and Addr(A, gE'') = Addr(gE'').

A.5 Rules for address manipulation

The rules for address manipulation can now be stated in a simple way:

1. An address can be added to a generalized endpoint gE only if this address is not an address of a different generalized endpoint with the same port number.
2. An address can be added to an association A with generalized endpoint gE if it has been added to the generalized endpoint gE first. This means that the address must be an element of Addr(gE) first and then it can become an element of Addr(A, gE). But this is not necessary. If the association does not allow the reconfiguration of the addresses only Addr(gE) can be modified.
3. An address can be deleted from an association A with generalized endpoint gE as long as Addr(A, gE) stays non-empty.
4. An address can be deleted from an generalized endpoint gE only if it has been removed from all associations having gE as a generalized endpoint.

These rules simply make sure that the rules for the endpoints and associations given above are always fulfilled.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.