SPRING W. Cheng, Ed.
Internet-Draft China Mobile
Updates: 8754 (if approved) C. Filsfils
Intended status: Standards Track Cisco Systems, Inc.
Expires: 10 August 2025 Z. Li
Huawei Technologies
B. Decraene
Orange
F. Clad, Ed.
Cisco Systems, Inc.
6 February 2025

Compressed SRv6 Segment List Encoding (CSID)
draft-ietf-spring-srv6-srh-compression-23

Abstract

Segment Routing over IPv6 (SRv6) is the instantiation of Segment Routing (SR) on the IPv6 dataplane. This document specifies new flavors for the SRv6 endpoint behaviors defined in RFC 8986, which enable the compression of an SRv6 segment list. Such compression significantly reduces the size of the SRv6 encapsulation needed to steer packets over long segment lists.

This document updates RFC 8754 by allowing a Segment List entry in the Segment Routing Header (SRH) to be either an IPv6 address, as specified in RFC 8754, or a REPLACE-CSID container in packed format, as specified in this document.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Segment Routing (SR) architecture [RFC8402] describes two data plane instantiations of SR: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6).

SRv6 Network Programming [RFC8986] builds upon the IPv6 Segment Routing Header (SRH) [RFC8754] to define a framework for constructing a network program with topological and service segments.

Some SRv6 applications such as strict path traffic engineering may require long segment lists. Compressing the encoding of these long segment lists in the packet header can significantly reduce the header size. This document specifies new flavors to the SRv6 endpoint behaviors defined in [RFC8986] that enable a compressed encoding of the SRv6 segment list. This document also specifies new SRv6 endpoint behaviors to preserve the compression efficiency in multi-domain environments.

The SRv6 endpoint behaviors defined in this document leverage the SRv6 data plane defined in [RFC8754] and [RFC8986], and are compatible with the SRv6 control plane extensions for IS-IS [RFC9352], OSPF [RFC9513], and BGP [RFC9252].

This document updates [RFC8754] by allowing a Segment List entry in the SRH to be either an IPv6 address, as specified in [RFC8754], or a

REPLACE-CSID container in packed format, as specified in Section 4.2.

2. Terminology

This document leverages the terms defined in [RFC8402], [RFC8754], and [RFC8986], in particular segment, segment list, Segment Identifier (SID), SID list, SR policy, prefix segment, adjacency segment, SRH, SR domain, SR source node, SR segment endpoint node, transit node, SRv6 endpoint behavior, flavor, SID block, locator, function, and argument. The reader is assumed to be familiar with this terminology.

This document introduces the following new terms:

* Locator-Block: The most significant bits of a SID locator that represent the SRv6 SID block. The Locator-Block is referred to as "B" in Section 3.1 of [RFC8986].

* Locator-Node: The least significant bits of a SID locator that identify the SR segment endpoint node instantiating the SID. The Locator-Node is referred to as "N" in Section 3.1 of [RFC8986].

* Compressed-SID (CSID): A compressed encoding of a SID. The CSID includes the Locator-Node and Function bits of the SID being compressed. If either constituent of the SID is empty (zero length), then the same applies to its CSID encoding.

* CSID container: A 128-bit IPv6 address that functions as a container holding a list of one or more CSIDs, and the Argument (if any) of the last CSID.

* CSID sequence: A group of one or more consecutive SID list entries encoding the common Locator-Block and at least one CSID container.

* Compressed SID list: A segment list encoding that reduces the packet header length thanks to one or more CSID sequences. A compressed SID list also contains zero, one, or more uncompressed SIDs.

* Global Identifiers Block (GIB): The pool of CSID values available for global allocation.

* Local Identifiers Block (LIB): The pool of CSID values available for local allocation.

In this document, the length of each constituent part of a SID is referred to as follows.

* LBL is the Locator-Block length of the SID.

* LNL is the Locator-Node length of the SID.

* FL is the Function length of the SID.

* AL is the Argument length of the SID.

In addition, the Locator-Node and Function length (LNFL) is the sum of the Locator-Node length and the Function length of the SID. It is also referred to as the CSID length.

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Basic Concepts

In an SR domain, all SRv6 SIDs instantiated from the same Locator-Block share the same most significant bits. In addition, when the combined length of the SRv6 SID Locator, Function, and Argument is smaller than 128 bits, the least significant bits of the SID are padded with zeros. The compressed segment list encoding seeks to decrease the packet header length by avoiding the repetition of the same Locator-Block and reducing the use of padding bits.

Building upon and fully compatible with the mechanisms specified in [RFC8754] and [RFC8986], the compressed segment list encoding leverages a SID list compression logic at the SR source node (see Section 6) in combination with new flavors of the SRv6 endpoint behaviors that process the compressed SID list (see Section 4).

An SR source node constructs and compresses the SID list depending on the SIDs instantiated on each SR segment endpoint node that the packet is intended to traverse, as well as its own compression capabilities. The resulting compressed SID list is a combination of CSID sequences, for the SIDs that the SR source node was able to compress, and uncompressed SIDs, which could not be compressed. In case the SR source node is able to compress all the SIDs in the SID list, the compressed SID list comprises only CSID sequences (one or more), and no uncompressed SIDs. Conversely, the compressed SID list comprises only uncompressed SIDs when the SR source is unable to compress any of the constituent SIDs.

4. SR Segment Endpoint Flavors

This section defines two SR segment endpoint flavors, NEXT-CSID and REPLACE-CSID, for the End, End.X, End.T, End.B6.Encaps, End.B6.Encaps.Red, and End.BM behaviors of [RFC8986].

This section also defines a REPLACE-CSID flavor for the End.DX6, End.DX4, End.DT6, End.DT4, End.DT46, End.DX2, End.DX2V, End.DT2U, and End.DT2M behaviors of [RFC8986]. A counterpart NEXT-CSID flavor is not defined for these behaviors: since any SID can be the last element of a CSID sequence compressed using the NEXT-CSID flavor (see

Section 4.1) and the aforementioned SRv6 endpoint behaviors are always in the last position in a SID list, there is no need for any modification of the behaviors defined in [RFC8986].

Future documents may extend the applicability of the NEXT-CSID and REPLACE-CSID flavors to other SRv6 endpoint behaviors (see Section 11).

The use of these flavors, either individually or in combination, enables the compressed segment list encoding.

The NEXT-CSID flavor and the REPLACE-CSID flavor both leverage the SID Argument to determine the next SID to be processed, but employ different SID list compression schemes. With the NEXT-CSID flavor, each CSID container is a fully formed SRv6 SID with the common Locator-Block for all the CSIDs in the CSID container, a Locator-Node and Function that are those of the first CSID, and an Argument carrying the subsequent CSIDs. With the REPLACE-CSID flavor, only the first element in a CSID sequence is a fully formed SRv6 SID. It has the common Locator-Block for all the CSIDs in the CSID sequence, and a Locator-Node and Function that are those of the first CSID. The remaining elements in the CSID sequence are CSID containers carrying the subsequent CSIDs without the Locator-Block.

Regardless of which flavor is used, the IPv6 address carried in the Destination Address field of the IPv6 header is a valid SRv6 SID conforming to [RFC9602].

In the remainder of this document, the term "a SID of this document" refers to any End, End.X, End.T, End.B6.Encaps, End.B6.Encaps.Red, or End.BM SID with the NEXT-CSID or the REPLACE-CSID flavor, and with any combination of Penultimate Segment Pop (PSP), Ultimate Segment Pop (USP), and Ultimate Segment Decapsulation (USD) flavor, or any End.DX6, End.DX4, End.DT6, End.DT4, End.DT46, End.DX2, End.DX2V, End.DT2U, or End.DT2M with the REPLACE-CSID flavor. All the SRv6 endpoint behaviors introduced in this document are listed in Table 1 at the end of the document.

In the remainder of this document, the terms "NEXT-CSID flavor SID" and "REPLACE-CSID flavor SID" refer to any SID of this document with the NEXT-CSID flavor and with the REPLACE-CSID flavor, respectively.

4.1. NEXT-CSID Flavor

A CSID sequence compressed using the mechanism of the NEXT-CSID flavor comprises one or more CSID containers. Each CSID container is a fully formed 128-bit SID structured as shown in Figure 1. It carries a Locator-Block followed by a series of CSIDs. The Locator-Node and Function of the CSID container are those of the first CSID, and its Argument is the contiguous series of subsequent CSIDs. The second CSID is encoded in the most significant bits of the CSID container Argument, the third CSID is encoded in the bits of the Argument that immediately follow the second CSID, and so on. When all CSIDs have the same length, a CSID container can carry up to K CSIDs, where K is computed as floor((128-LBL)/LNFL) (floor(x) is the greatest integer less than or equal to x [GKP94]). Each CSID container for NEXT-CSID is independent, such that contiguous CSID containers in a CSID sequence can be considered as separate CSID sequences.

When a CSID sequence compressed using the NEXT-CSID flavor comprises at least two CSIDs, the last CSID in the sequence is not required to have the NEXT-CSID flavor. It can be bound to any SRv6 endpoint behavior, including [RFC8986] behaviors and REPLACE-CSID flavor, as long as the updated destination address resulting from the processing of the previous CSID in the sequence is a valid form for that last SID. Line S12 of the first pseudocode in Section 6.2 provides sufficient conditions to ensure this property.

```
+------------------------------------------------------------------+
|     Locator-Block      |Loc-Node|          Argument              |
|                        |Function|                                |
+------------------------------------------------------------------+
 <----------------------> <------> <------------------------------>
          LBL               LNFL                 AL
```

Figure 1: Structure of a NEXT-CSID flavor SID (scaled for a 48-bit Locator- Block, 16-bit combined Locator-Node and Function, and 64-bit Argument)

Figure 2 illustrates a compressed SID list as could be produced by an SR source node steering a packet into an SR policy with a SID list of eight NEXT-CSID flavor SIDs. All SIDs in this example have a 48-bit Locator-Block, 16-bit combined Locator-Node and Function, and 64-bit Argument. The SR source node compresses the SR policy SID list as a compressed SID list of two CSID containers. The first CSID container carries a Locator-Block and the first five CSIDs. The second CSID container carries a Locator-Block and the sixth, seventh, and eighth CSIDs. Since the SR source node does not use the second CSID container at full capacity, it sets the 32 least significant bits to

zero. The SR source node sets the IPv6 Destination Address (DA) with the value of the first CSID container and the first element of the SRH Segment List with the value of the second CSID container. Without reduced SRH (Section 4.1.1 of [RFC8754]), the SR source node also writes the first CSID container as the second element of the SRH Segment List.

Note that the CSIDs within a given CSID container appear in forward order to leverage the longest-prefix match IP forwarding, while the entries in the SRH Segment List appear in reversed order of their processing, as specified in Section 4.1 of [RFC8754].

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+         Locator-Block         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |           1st CSID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           2nd CSID            |           3rd CSID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           4th CSID            |           5th CSID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      First CSID container

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+         Locator-Block         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |           6th CSID            |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          7th CSID             |          8th CSID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               0                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      Second CSID container
```

Figure 2: Compressed SID list of eight NEXT-CSID flavor SIDs with a 48-bit Locator-Block, 16-bit combined Locator-Node and Function, and 64-bit Argument

An implementation MUST support a 32-bit Locator-Block length (LBL) and a 16-bit CSID length (LNFL) for NEXT-CSID flavor SIDs, and MAY support any additional Locator-Block and CSID length.

The Argument length (AL) for NEXT-CSID flavor SIDs is equal to 128-LBL-LNFL.

When processing an IPv6 packet that matches a Forwarding Information Base (FIB) entry locally instantiated as a SID with the NEXT-CSID flavor, the SR segment endpoint node applies the procedure specified in the following subsection that corresponds to the SID behavior. If the SID also has the PSP, USP, or USD flavor, the procedure is modified as described in Section 4.1.7.

An SR segment endpoint node instantiating a SID of this document with the NEXT-CSID flavor MUST accept any Argument value for that SID.

At a high level, for any SID with the NEXT-CSID flavor, the SR segment endpoint node determines the next SID of the SID list as follows. If the Argument value of the active SID is non-zero, the SR segment endpoint node constructs the next SID from the active SID by copying the entire SID Argument value to the bits that immediately follow the Locator-Block, thus overwriting the active SID Locator-Node and Function with those of the next CSID, and filling the least significant LNFL bits of the Argument with zeros. Otherwise (if the Argument value is 0), the SR segment endpoint node copies the next 128-bit Segment List entry from the SRH to the Destination Address field of the IPv6 header.

4.1.1. End with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End SID with the NEXT-CSID flavor, the procedure described in Section 4.1 of [RFC8986] is executed with the following modifications.

The below pseudocode is inserted between lines S01 and S02 of the SRH processing in Section 4.1 of [RFC8986]. In addition, this pseudocode is executed before processing any extension header that is not an SRH, a Hop-by-Hop header or a Destination Options header, or before processing the upper-layer header, whichever comes first.

```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.   Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.
N09. }
```

> Notes:
>
> * DA.Argument identifies the value contained in the bits [(LBL+LNFL)..127] in the Destination Address of the IPv6 header.
>
> * The value in the Segments Left field of the SRH is not modified when DA.Argument in the received packet has a non-zero value.

A rendering of the complete pseudocode is provided in Appendix A.1.

4.1.2. End.X with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.X SID with the NEXT-CSID flavor, the procedure described in Section 4.2 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.1.1 of this document is modified by replacing line N08 as shown below.

```
N08.   Submit the packet to the IPv6 module for transmission to the
         new destination via a member of J.
```

> Note: the variable J is defined in Section 4.2 of [RFC8986].

The resulting pseudocode is inserted between lines S01 and S02 of the

SRH processing in Section 4.1 of [RFC8986] after applying the modification described in Section 4.2 of [RFC8986]. In addition, this pseudocode is executed before processing any extension header that is not an SRH, a Hop-by-Hop header or a Destination Options header, or before processing the upper-layer header, whichever comes first.

A rendering of the complete pseudocode is provided in Appendix A.2.

4.1.3. End.T with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.T SID with the NEXT-CSID flavor, the procedure described in Section 4.3 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.1.1 of this document is modified by replacing line N08 as shown below.

```
N08.1.   Set the packet's associated FIB table to T.
N08.2.   Submit the packet to the egress IPv6 FIB lookup for
           transmission to the new destination.
```

> Note: the variable T is defined in Section 4.3 of [RFC8986].

The resulting pseudocode is inserted between lines S01 and S02 of the SRH processing in Section 4.1 of [RFC8986] after applying the modification described in Section 4.3 of [RFC8986]. In addition, this pseudocode is executed before processing any extension header that is not an SRH, a Hop-by-Hop header or a Destination Options header, or before processing the upper-layer header, whichever comes first.

A rendering of the complete pseudocode is provided in Appendix A.3.

4.1.4. End.B6.Encaps with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.B6.Encaps SID with the NEXT-CSID flavor, the procedure described in Section 4.13 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.1.1 of this document is modified by replacing line N08 as shown below.


```
N08.1.   Push a new IPv6 header with its own SRH containing B.
N08.2.   Set the outer IPv6 SA to A.
N08.3.   Set the outer IPv6 DA to the first SID of B.
N08.4.   Set the outer Payload Length, Traffic Class, Flow Label,
           Hop Limit, and Next Header fields.
N08.5.   Submit the packet to the egress IPv6 FIB lookup for
           transmission to the next destination.
```

> Note: the variables A and B, as well as the values of the Payload Length, Traffic Class, Flow Label, Hop Limit, and Next Header are defined in Section 4.13 of [RFC8986].

The resulting pseudocode is inserted between lines S01 and S02 of the SRH processing in Section 4.13 of [RFC8986]. In addition, this pseudocode is executed before processing any extension header that is not an SRH, a Hop-by-Hop header or a Destination Options header, or before processing the upper-layer header, whichever comes first.

A rendering of the complete pseudocode is provided in Appendix A.4.

Similar to the base End.B6.Encaps SID defined in Section 4.13 of [RFC8986], the NEXT-CSID flavor variant updates the Destination Address field of the inner IPv6 header to the next SID in the original segment list before encapsulating the packet with the segment list of SR Policy B. At the endpoint of SR Policy B, the encapsulation is removed and the inner packet is forwarded towards the exposed destination address, which already contains the next SID in the original segment list.

4.1.5. End.B6.Encaps.Red with NEXT-CSID

This is an optimization of the End.B6.Encaps with NEXT-CSID behavior.

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.B6.Encaps.Red SID with the NEXT-CSID flavor, the procedure described in Section 4.1.4 of this document is executed with the modifications in Section 4.14 of [RFC8986].

4.1.6. End.BM with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.BM SID with the NEXT-CSID flavor, the procedure described in Section 4.15 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.1.1 of this document is modified by replacing line N08 as shown below.


```
N08.1.   Push the MPLS label stack for B.
N08.2.   Submit the packet to the MPLS engine for transmission.
```

> Note: the variable B is defined in Section 4.15 of [RFC8986].

The resulting pseudocode is inserted between lines S01 and S02 of the SRH processing in Section 4.15 of [RFC8986]. In addition, this pseudocode is executed before processing any extension header that is not an SRH, a Hop-by-Hop header or a Destination Options header, or before processing the upper-layer header, whichever comes first.

A rendering of the complete pseudocode is provided in Appendix A.5.

4.1.7. Combination with PSP, USP and USD flavors

PSP: The PSP flavor defined in Section 4.16.1 of [RFC8986] is unchanged when combined with the NEXT-CSID flavor.

USP: The USP flavor defined in Section 4.16.2 of [RFC8986] is unchanged when combined with the NEXT-CSID flavor.

USD: The USP flavor defined in Section 4.16.3 of [RFC8986] is unchanged when combined with the NEXT-CSID flavor.

4.2. REPLACE-CSID Flavor

A CSID sequence compressed using the mechanism of the REPLACE-CSID flavor starts with a CSID container in fully formed 128-bit SID format. The Locator-Block of this SID is the common Locator-Block for all the CSIDs in the CSID sequence, its Locator-Node and Function are those of the first CSID, and its Argument carries the index of the current CSID in the current CSID container. The Argument value is initially 0. When more segments are present in the segment list, the CSID sequence continues with one or more CSID containers in packed format carrying the series of subsequent CSIDs. Each container in packed format is a 128-bit Segment List entry split into K "positions" of LNFL bits, where K is computed as floor(128/LNFL). If LNFL does not divide into 128 perfectly, a zero pad is added in the least significant bits of the CSID container to fill the bits left over. The second CSID in the CSID sequence is encoded in the least significant bit position of the first CSID container in packed format (position K-1), the third CSID is encoded in position K-2, and so on.

The last CSID in the CSID sequence is not required to have the REPLACE-CSID flavor. It can be bound to any SRv6 endpoint behavior, including [RFC8986] behaviors and NEXT-CSID flavor, as long as it meets the conditions defined in Section 6.

The structure of a SID with the REPLACE-CSID flavor is shown in Figure 3. The same structure is also that of the CSID container for REPLACE-CSID in fully formed 128-bit SID format.

```
+-------------------------------------------------------------------+
|     Locator-Block      |  Locator-Node  |         Argument        |
|                        |   + Function   |                         |
+-------------------------------------------------------------------+
 <----------------------> <---------------> <----------------------->
           LBL                  LNFL                   AL
```

Figure 3: Structure of a REPLACE-CSID flavor SID (scaled for a 48-bit Locator- Block, 32-bit combined Locator-Node and Function, and 48-bit Argument)

The structure of a CSID container for REPLACE-CSID in packed format is shown in Figure 4.

```
+-------------------------------------------------------------------+
|  Fourth CSID   |   Third CSID   |  Second CSID   |   First CSID   |
|  (position 0)  |  (position 1)  |  (position 2)  |  (position 3)  |
+-------------------------------------------------------------------+
 <--------------> <--------------> <--------------> <-------------->
       LNFL             LNFL             LNFL             LNFL
```

Figure 4: Structure of a CSID container for REPLACE-CSID using a 32-bit CSID length (K = 4)

Figure 5 illustrates a compressed SID list as could be produced by an SR source node steering a packet into an SR policy SID list of seven REPLACE-CSID flavor SIDs. All SIDs in this example have a 48-bit Locator-Block, 32-bit combined Locator-Node and Function, and 48-bit Argument. The SR source node compresses the SR policy SID list as a compressed SID list of three CSID containers. The first CSID container is in fully formed 128-bit SID format. It carries a Locator-Block, the first CSID, and the argument value zero. The second and third CSID containers are in packed format. The second CSID container carries the second, third, fourth, and fifth CSIDs. The third CSID container carries the sixth and seventh CSIDs. Since the SR source node does not use the third CSID container at full capacity, it sets the 64 least significant bits to zero. The SR source node sets the IPv6 DA with the value of the first CSID container, sets the first element in the SRH Segment List with the value of the third CSID container, and sets the second element of the SRH Segment List with the value of the second CSID container (the elements in the SRH Segment List appear in reversed order of their processing, as specified in Section 4.1 of [RFC8754]). Without reduced SRH, the SR source node also writes the first CSID container as the third element of the SRH Segment List.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+         Locator-Block         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |           1st CSID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      1st CSID continued       |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               0               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      First CSID container

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           5th CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           4th CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           3rd CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           2nd CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      Second CSID container

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                               0                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           7th CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           6th CSID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      Third CSID container
```

Figure 5: Compressed SID list of seven REPLACE-CSID flavor SIDs with a 48-bit Locator-Block, 32-bit combined Locator-Node and Function, and 48-bit Argument

This document updates [RFC8754] by allowing each entry in the SRH Segment List to be either an IPv6 address or a REPLACE-CSID container in packed format. The SRv6 endpoint behaviors specified herein ensure that this entry is never copied as is to the IPv6 header and that the Destination Address field of the IPv6 header is always a valid SRv6 SID conforming to [RFC9602].

The REPLACE-CSID flavor SIDs support any Locator-Block length (LBL), depending on the needs of the operator, as long as it does not exceed 128-LNFL-ceiling(log_2(128/LNFL)) (ceiling(x) is the least integer greater than or equal to x [GKP94]), so that enough bits remain available for the CSID and Argument. A Locator-Block length of 48, 56, 64, 72, or 80 bits is recommended for easier reading in operation.

This document defines the REPLACE-CSID flavor for 16-bit and 32-bit CSID lengths (LNFL). An implementation MUST support a 32-bit CSID length for REPLACE-CSID flavor SIDs.

The Argument length (AL) for REPLACE-CSID flavor SIDs is equal to 128-LBL-LNFL. The index value is encoded in the least significant X bits of the Argument, where X is computed as ceiling(log_2(128/LNFL)).

When processing an IPv6 packet that matches a FIB entry locally instantiated as a SID with the REPLACE-CSID flavor, the SR segment endpoint node applies the procedure specified in the following subsection that corresponds to the SID behavior. If the SID also has the PSP, USP, or USD flavor, the procedure is modified as described in Section 4.2.8.

At a high level, at the start of a CSID sequence using the REPLACE-CSID flavor, the first CSID container in fully formed 128-bit SID format is copied to the Destination Address of the IPv6 header. Then, for any SID with the REPLACE-CSID flavor, the SR segment endpoint node determines the next SID of the SID list as follows. When an SRH is present, the SR segment endpoint node decrements the index value in the Argument of the active SID if the index value is not 0 or, if it is 0, decrements the Segments Left value in the SRH and sets the index value in the Argument of the active SID to K-1. The updated index value indicates the position of the next CSID within the CSID container in packed format at the "Segment List" index "Segments Left" in the SRH. The SR segment endpoint node then constructs the next SID by copying this next CSID to the bits that immediately follow the Locator-Block in the Destination Address field of the IPv6 header, thus overwriting the active SID Locator-Node and Function with those of the next CSID. If no SRH is present, the SR segment endpoint node ignores the index value in the SID Argument

(except End.DT2M, see Section 4.2.7) and processes the upper-layer header as per [RFC8986]. The CSID sequence ends with a last CSID in the last CSID container that does not have the REPLACE-CSID flavor, or with the special CSID value 0, or when reaching the end of the segment list, whichever comes first.

4.2.1. End with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End SID with the REPLACE-CSID flavor, the SRH processing described in Section 4.1 of [RFC8986] is executed with the following modifications.

Line S02 of SRH processing in Section 4.1 of [RFC8986] is replaced as follows.

```
S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
           Segment List[0][DA.Arg.Index-1] == 0)) {
```

Lines S09 to S15 are replaced by the following pseudo code.

```
R01. If (DA.Arg.Index != 0) {
R02.   If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
R03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.
R04.   }
R05.   Decrement DA.Arg.Index by 1.
R06.   If (Segment List[Segments Left][DA.Arg.Index] == 0) {
R07.     Decrement Segments Left by 1.
R08.     Decrement IPv6 Hop Limit by 1.
R09.     Update IPv6 DA with Segment List[Segments Left]
R10.     Submit the packet to the egress IPv6 FIB lookup for
          transmission to the new destination.
R11.   }
R12. } Else {
R13.   If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
R14.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.
R15.   }
R16.   Decrement Segments Left by 1.
R17.   Set DA.Arg.Index to (floor(128/LNFL) - 1).
R18. }
R19. Decrement IPv6 Hop Limit by 1.
R20. Write Segment List[Segments Left][DA.Arg.Index] into the bits
       [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
       header.
R21. Submit the packet to the egress IPv6 FIB lookup for
       transmission to the new destination.
```

> Notes:
>
> * DA.Arg.Index identifies the value contained in the bits
>   [(128-ceiling(log_2(128/LNFL)))..127] in the Destination

```
   |        Address of the IPv6 header.
   |
   |     *  Segment List[Segments Left][DA.Arg.Index] identifies the
   |        value contained in the bits
   |        [DA.Arg.Index*LNFL..(DA.Arg.Index+1)*LNFL-1] in the SRH
   |        Segment List entry at index Segments Left.
```

The upper-layer header processing described in Section 4.1.1 of [RFC8986] is unchanged.

A rendering of the complete pseudocode is provided in Appendix A.6.

4.2.2. End.X with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.X SID with the REPLACE-CSID flavor, the procedure described in Section 4.2 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.2.1 of this document is modified by replacing lines R10 and R21 as shown below.

```
R10. Submit the packet to the IPv6 module for transmission to the
       new destination via a member of J.

R21. Submit the packet to the IPv6 module for transmission to the
       new destination via a member of J.
```

| Note: the variable J is defined in Section 4.2 of [RFC8986].

The SRH processing in Section 4.2 of [RFC8986] is replaced with the resulting pseudocode. The upper-layer header processing is unchanged.

A rendering of the complete pseudocode is provided in Appendix A.7.

4.2.3. End.T with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.T SID with the REPLACE-CSID flavor, the procedure described in Section 4.3 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.2.1 of this document is modified by replacing lines R10 and R21 as shown below.

```
R10.1. Set the packet's associated FIB table to T.
R10.2. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the new destination.

R21.1. Set the packet's associated FIB table to T.
R21.2. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the new destination.
```

| Note: the variable T is defined in Section 4.3 of [RFC8986].

The SRH processing in Section 4.3 of [RFC8986] is replaced with the resulting pseudocode. The upper-layer header processing is unchanged.

A rendering of the complete pseudocode is provided in Appendix A.8.

4.2.4. End.B6.Encaps with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.B6.Encaps SID with the REPLACE-CSID flavor, the procedure described in Section 4.13 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.2.1 of this document is modified by replacing lines R10 and R21 as shown below.

```
R10.1. Push a new IPv6 header with its own SRH containing B.
R10.2. Set the outer IPv6 SA to A.
R10.3. Set the outer IPv6 DA to the first SID of B.
R10.4. Set the outer Payload Length, Traffic Class, Flow Label,
         Hop Limit, and Next Header fields.
R10.5. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.

R21.1. Push a new IPv6 header with its own SRH containing B.
R21.2. Set the outer IPv6 SA to A.
R21.3. Set the outer IPv6 DA to the first SID of B.
R21.4. Set the outer Payload Length, Traffic Class, Flow Label,
         Hop Limit, and Next Header fields.
R21.5. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.
```

> Note: the variables A and B, as well as the values of the Payload Length, Traffic Class, Flow Label, Hop Limit, and Next Header are defined in Section 4.13 of [RFC8986].

The SRH processing in Section 4.13 of [RFC8986] is replaced with the resulting pseudocode. The upper-layer header processing is unchanged.

A rendering of the complete pseudocode is provided in Appendix A.9.

4.2.5. End.B6.Encaps.Red with REPLACE-CSID

This is an optimization of the End.B6.Encaps with REPLACE-CSID behavior.

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.B6.Encaps.Red SID with the REPLACE-CSID flavor, the procedure described in Section 4.2.4 of this document is executed with the modifications in Section 4.14 of [RFC8986].

4.2.6. End.BM with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.BM SID with the REPLACE-CSID flavor, the procedure described in Section 4.15 of [RFC8986] is executed with the following modifications.

The pseudocode in Section 4.2.1 of this document is modified by replacing lines R10 and R21 as shown below.

```
R10.1. Push the MPLS label stack for B.
R10.2. Submit the packet to the MPLS engine for transmission.

R21.1. Push the MPLS label stack for B.
R21.2. Submit the packet to the MPLS engine for transmission.
```

> Note: the variable B is defined in Section 4.15 of [RFC8986].

The SRH processing in Section 4.15 of [RFC8986] is replaced with the resulting pseudocode. The upper-layer header processing is unchanged.

A rendering of the complete pseudocode is provided in Appendix A.10.

4.2.7. End.DX and End.DT with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.DX6, End.DX4, End.DT6, End.DT4, End.DT46, End.DX2, End.DX2V, or End.DT2U SID with the REPLACE-CSID flavor, the corresponding procedure described in Sections 4.4 through 4.11 of [RFC8986] is executed.

These SIDs differ from those defined in [RFC8986] by the presence of an Argument as part of the SID structure. The Argument value is ignored by the SR segment endpoint node.

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.DT2M SID with the REPLACE-CSID flavor, the procedure described in Section 4.12 of [RFC8986] is executed with the following modification.

For any End.DT2M SID with the REPLACE-CSID flavor, the value of Arg.FE2 is 16-bit long. The SR segment endpoint node obtains the value Arg.FE2 from the 16 most significant bits of DA.Argument if DA.Arg.Index is zero, or from the 16 least significant bits of the next position in the current CSID container (Segment List[Segments Left][DA.Arg.Index-1]) otherwise (DA.Arg.Index is non-zero).

4.2.8. Combination with PSP, USP, and USD flavors

PSP: When combined with the REPLACE-CSID flavor, the additional PSP flavor instructions defined in Section 4.16.1.2 of [RFC8986] are inserted after lines R09 and R20 of the pseudocode in Section 4.2.1, and the first line of the inserted instructions after R20 is modified as follows.

```
R20.1.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
                 Segment List[0][DA.Arg.Index-1] == 0)) {
```

> Note: Segment List[Segments Left][DA.Arg.Index-1] identifies the value contained in the bits [(DA.Arg.Index-1)*LNFL..DA.Arg.Index*LNFL-1] in the SRH Segment List entry at index Segments Left.

USP: When combined with the REPLACE-CSID flavor, the line S03 of the pseudocode in Section 4.2.1 are substituted by the USP flavor instructions S03.1 to S03.4 defined in Section 4.16.2 of [RFC8986].

Note that S03 is shown in the complete pseudocode in Appendix A.6.

USD: The USD flavor defined in Section 4.16.3 of [RFC8986] is unchanged when combined with the REPLACE-CSID flavor.

5. CSID Allocation

The CSID value of 0 is reserved. It is used to indicate the end of a CSID container.

In order to efficiently manage the CSID numbering space, a deployment may divide it into two non-overlapping sub-spaces: a Global Identifiers Block (GIB) and a Local Identifiers Block (LIB).

The CSID values that are allocated from the GIB have a global semantic within the Locator-Block, while those that are allocated from the LIB have a local semantic on an SR segment endpoint node and within the scope of the Locator-Block.

The concept of LIB is applicable to SRv6 and specifically to its NEXT-CSID and REPLACE-CSID flavors. The shorter the CSID, the more benefit the LIB brings.

The opportunity to use these sub-spaces, their size, and their CSID allocation policy depends on the CSID length relative to the size of the network (e.g., number of nodes, links, service routes). Some guidelines for a typical deployment scenario are provided in the below subsections.

5.1. Global CSID

A global CSID is a CSID allocated from the GIB.

A global CSID identifies a segment defined at the Locator-Block level. The tuple (Locator-Block, CSID) identifies the same segment across all nodes of the SR domain. A typical example is a prefix segment bound to the End behavior.

A node can have multiple global CSIDs under the same Locator-Block (e.g., one per IGP flexible algorithm ([RFC9350])). Multiple nodes may share the same global CSID (e.g., anycast [RFC4786]).

5.2. Local CSID

A local CSID is a CSID allocated from the LIB.

A local CSID identifies a segment defined at the node level and within the scope of a particular Locator-Block. The tuple (Locator-Block, CSID) identifies a different segment on each node of the SR domain. A typical example is a non-routed Adjacency segment bound to the End.X behavior.

Let N1 and N2 be two different physical nodes of the SR domain and I a local CSID value, N1 may allocate value I to SID S1 and N2 may allocate the same value I to SID S2.

5.3. Recommended Installation of CSIDs in FIB

Section 4.3 of [RFC8754] defines how an SR segment endpoint node identifies a locally instantiated SRv6 SID. To ensure that any valid argument value is accepted, an SR segment endpoint node instantiating a NEXT-CSID or REPLACE-CSID flavor SID should install a corresponding FIB entry that matches only the Locator and Function parts of the SID (i.e., with a prefix length of LBL + LNL + FL).

In addition, an SR segment endpoint node instantiating NEXT-CSID flavor SIDs from both GIB and LIB may install combined "Global + Local" FIB entries to match a sequence of global and local CSIDs in a single longest prefix match (LPM) lookup.

For example, let us consider an SR segment endpoint node 10 instantiating the following two NEXT-CSID flavor SIDs according to the CSID length, Locator-Block length, and GIB/LIB recommendations in this section.

* The SID 2001:db8:b1:10:: bound to the End behavior with the NEXT-CSID flavor is instantiated from GIB with

  - Locator-Block length (LBL) = 48 (Locator-Block value 0x20010db800b1),

  - Locator-Node length (LNL) = 16 (Locator-Node value 0x0010),

  - Function length (FL) = 0, and

  - Argument length (AL) = 64.

* The SID 2001:db8:b1:f123:: bound to the End.X behavior for its local IGP adjacency 123 with the NEXT-CSID flavor is instantiated from LIB with

  - Locator-Block length (LBL) = 48 (Locator-Block value 0x20010db800b1),

  - Locator-Node length (LNL) = 0,

  - Function length (FL) = 16 (Function value 0xf123), and

  - Argument length (AL) = 64.

For SID 2001:db8:b1:10::, Node 10 would install the FIB entry 2001:db8:b1:10::/64 bound the End SID with the NEXT-CSID flavor.

For SID 2001:db8:b1:f123::, Node 10 would install the FIB entry 2001:db8:b1:f123::/64 bound the End.X SID for adjacency 123 with the NEXT-CSID flavor.

In addition, Node 10 may also install the combined FIB entry 2001:db8:b1:10:f123::/80 bound the End.X SID for adjacency 123 with the NEXT-CSID flavor.

As another example, let us consider an SR segment endpoint node 20 instantiating the following two REPLACE-CSID flavor SIDs according to the CSID length, Locator-Block length, and GIB/LIB recommendations in this section.

* 2001:db8:b2:20:1:: from GIB with Locator-Block length (LBL) = 48, Locator-Node length (LNL) = 16, Function length (FL) = 16, Argument length (AL) = 48, and bound to the End behavior with the REPLACE-CSID flavor.

* 2001:db8:b2:20:123:: from GIB with Locator-Block length (LBL) = 48, Locator-Node length (LNL) = 16, Function length (FL) = 16,

Argument length (AL) = 48, and bound to the End.X behavior for its local IGP adjacency 123 with the REPLACE-CSID flavor.

For SID 2001:db8:b2:20:1::, Node 20 would install the FIB entry 2001:db8:b2:20:1::/80 bound the End SID with the REPLACE-CSID flavor.

For SID 2001:db8:b2:20:123::, Node 20 would install the FIB entry 2001:db8:b2:20:123::/80 bound the End.X SID for adjacency 123 with the REPLACE-CSID flavor.

6. SR Source Node

An SR source node may learn from a control plane protocol (see Section 8) or local configuration the SIDs that it can use in a segment list, along with their respective SRv6 endpoint behavior, structure, and any other relevant attribute (e.g., the set of L3 adjacencies associated with an End.X SID).

6.1. SID Validation for Compression

As part of the compression process or as a preliminary step, the SR source node MUST validate the SID structure of each SID of this document in the segment list. The SR source node does so regardless of whether the segment list is explicitly configured, locally computed, or advertised by a controller (e.g., via BGP [I-D.ietf-idr-sr-policy-safi] or PCEP [RFC9603]).

A SID structure is valid for compression if it meets all the following conditions.

* The Locator-Block length is not 0.

* The sum of the Locator-Node length and Function length is not 0.

* The Argument length is equal to 128-LBL-LNL-FL.

When compressing a SID list, the SR source node MUST treat an invalid SID structure as unknown. A SID with an unknown SID structure is incompressible.

Section 8 discusses how the SIDs of this document and their structure can be advertised to the SR source node through various control plane protocols. The SID structure may also be learned through configuration or other management protocols. The details of such mechanisms are outside the scope of this document.

6.2. Segment List Compression

An SR source node MAY compress a SID list when it includes NEXT-CSID and/or REPLACE-CSID flavor SIDs to reduce the packet header length.

It is out of the scope of this document to describe the mechanism through which an uncompressed SID list is derived, since such mechanism may include a wide range of considerations independent of compression (e.g., minimizing a specific metric, excluding certain links, or providing a loop-free fast-reroute path). As a general guidance for implementation or future specification, such a mechanism should aim to select the combination of SIDs that would result in the shortest compressed SID list. For example, by selecting a CSID flavor SID over an equivalent non-CSID flavor SID or by consistently selecting SIDs of the same CSID flavor within each routing domain.

The SID list that the SR source node pushes onto the packet MUST comply with the rules in Section 6.3 and Section 6.4 and express the same list of segments as the original SID list. If these rules are not followed, the packet may get dropped or misrouted.

If an SR source node chooses to compress the SID list, one method is described below for illustrative purposes. Any other method producing a compressed SID list of equal or shorter length than the uncompressed SID list MAY be used.

This method walks the uncompressed SID list and compresses each series of consecutive NEXT-CSID flavor SIDs and each series of consecutive REPLACE-CSID flavor SIDs.

* When the compression method encounters a series of one or more consecutive compressible NEXT-CSID flavor SIDs, it compresses the series as follows. A SID with the NEXT-CSID flavor is compressible if its structure is known to the SR source node and its Argument value is 0.


```
S01. Initialize a NEXT-CSID container equal to the first SID in the
       series, and initialize the remaining capacity of the CSID
       container to the AL of that SID
S02. For each subsequent SID in the series {
S03.   If the current SID Locator-Block matches that of the CSID
         container and the current SID LNFL is lower than or equal to
         the remaining capacity of the NEXT-CSID container {
S04.     Copy the current SID Locator-Node and Function to the most
           significant remaining Argument bits of the NEXT-CSID
           container and decrement the remaining capacity by LNFL
S05.   } Else {
S06.     Push the NEXT-CSID container onto the compressed SID list
S07.     Initialize a new NEXT-CSID container equal to the current
           SID in the series, and initialize the remaining capacity
           of the NEXT-CSID container to the AL of that SID
```

```
   S08.   } // End If
   S09. } // End For
   S10. If at least one SID remains in the uncompressed SID list
          (following the series of compressible NEXT-CSID flavor SIDs){
   S11.   Set S to the next SID in the uncompressed SID list
   S12.   If S is advertised with a SID structure, and the Locator-Block
            of S matches that of the NEXT-CSID container, and the sum of
            the Locator-Node, Function, and Argument length of S is
            lower than or equal to the remaining capacity of the CSID
            container {
   S13.     Copy the Locator-Node, Function, and Argument of S to the
              most significant remaining Argument bits of the CSID
              container
   S14.   } // End If
   S15. } // End If
   S16. Push the NEXT-CSID container onto the compressed SID list
```

* When the compression method encounters a series of REPLACE-CSID flavor SIDs of the same CSID length in the uncompressed SID list, it compresses the series as per the following high-level pseudo code. A compression checking function ComCheck(F, S) is defined to check if two SIDs F and S share the same SID structure and Locator-Block value, and if S has either no Argument or an Argument with value 0. If the check passes, then ComCheck(F,S) returns true.


```
   S01. Initialize a REPLACE-CSID container in full SID format equal to
          the first SID in the series
   S02. Push the REPLACE-CSID container onto the compressed SID list
   S03. Initialize a new REPLACE-CSID container in packed format if
          there are more than one SIDs, and initialize the remaining
          capacity of the REPLACE-CSID container to 128 bits
   S04. For each subsequent SID in the uncompressed SID list {
   S05.   Set S to the current SID in the uncompressed SID list
   S06.   If ComCheck(First SID, S) {
   S07.     If the LNFL of S is lower than or equal to
              the remaining capacity of the REPLACE-CSID container {
   S08.       Copy the Locator-Node and Function of S to the least
                significant remaining bits of the REPLACE-CSID container
                and decrement the remaining capacity by LNFL  // Note
   S09.     } Else {
   S10.       Push the REPLACE-CSID container onto the compressed SID
                list
   S11.       Initialize a new REPLACE-CSID container in packed format
                with all bits set to 0
   S12.       Copy the Locator-Node and Function of S to the least
                significant remaining bits of the REPLACE-CSID container
                and decrement the remaining capacity by LNFL  // Note
   S13.     }
   S14.     If S is not a REPLACE-CSID flavor SID, then break
   S15.   } Else {
   S16.     Break
   S17.   } // End If
   S18. } // End For
```

S19. Push the REPLACE-CSID container (if it is not empty) onto the compressed SID list

> Note: When the last CSID is an End.DT2M SID with the REPLACE-CSID flavor, if there is 0 or at least two CSID positions left in the current REPLACE-CSID container, the CSID is encoded as described above and the value of the Arg.FE2 argument is placed in the 16 least significant bits of the next CSID position. Otherwise (if there is only one CSID position left in the current REPLACE-CSID container), the current REPLACE-CSID container is pushed onto the SID list (the value of the CSID position 0 remains zero) and the End.DT2M SID with the REPLACE-CSID flavor is encoded in full SID format with the value of the Arg.FE2 argument in the 16 most significant bits of the SID Argument.

* In all remaining cases (i.e., when the compression method encounters a SID in the uncompressed SID list that is not handled by any of the previous subroutines), it pushes this SID as is onto the compressed SID list.

Regardless of how a compressed SID list is produced, the SR source node writes it in the IPv6 packet as described in Sections 4.1 and 4.1.1 of [RFC8754]. The text is reproduced below for reference.

> A source node steers a packet into an SR Policy. If the SR Policy results in a Segment List containing a single segment, and there is no need to add information to the SRH flag or add TLV; the DA is set to the single Segment List entry, and the SRH MAY be omitted.
>
> When needed, the SRH is created as follows:
>
> The Next Header and Hdr Ext Len fields are set as specified in [RFC8200].
>
> The Routing Type field is set to 4.
>
> The DA of the packet is set with the value of the first segment.
>
> The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment, and so on.
>
> The Segments Left field is set to n-1, where n is the number of elements in the SR Policy.
>
> The Last Entry field is set to n-1, where n is the number of elements in the SR Policy.
>
> TLVs (including HMAC) may be set according to their specification.
>
> The packet is forwarded toward the packet's Destination Address (the first segment).
>
> When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.
>
> A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to n-2, where n is the number of elements in the SR Policy.

6.3. Rules for segment lists containing NEXT-CSID flavor SIDs

1. If a Destination Options header would follow an SRH with a segment list of more than one segment compressed as a single NEXT-CSID container, the SR source node MUST NOT omit the SRH.

2. When the last Segment List entry (index 0) in the SRH is a NEXT-CSID container representing more than one segment and the segment S preceding the first segment of this NEXT-CSID container in the segment list has the PSP flavor, then the PSP operation is performed at the SR segment endpoint node of S. If the PSP behavior should instead be performed at the penultimate segment along the path, then the SR source node MUST NOT compress the ultimate SID of the SID list into a NEXT-CSID container.

3. If a Destination Options header would follow an SRH with a last Segment List entry being a NEXT-CSID container representing more than one segment, the SR source node MUST ensure that the PSP operation is not performed before the penultimate SR segment endpoint node along the path.

4. When the Argument of a NEXT-CSID container is not used to full capacity, the remaining least significant bits of that Argument MUST be set to 0.

6.4. Rules for segment lists containing REPLACE-CSID flavor SIDs

1. All SIDs compressed in a REPLACE-CSID sequence MUST share the same Locator-Block and the same compression scheme.

2. All SIDs except the last one in a CSID sequence for REPLACE-CSID MUST have the REPLACE-CSID flavor. If the last REPLACE-CSID container is fully filled (i.e., the last CSID is at position 0 in the REPLACE-CSID container) and the last SID in the CSID sequence is not the last segment in the segment list, the last SID in the CSID sequence MUST NOT have the REPLACE-CSID flavor.

3. When a REPLACE-CSID flavor CSID is present as the last SID in a container that is not the last Segment List entry (index 0) in the SRH, the next element in the SID list MUST be a REPLACE-CSID container in packed format carrying at least one CSID.

The SR source node determines the compression scheme of REPLACE-CSID flavor SIDs as follows.

When receiving a SID advertisement for a REPLACE-CSID flavor SID with LNL=16, FL=0, AL=128-LBL-LNFL, and the value of the Argument is all 0, the SR source node marks both the SID and its locator as using 16-bit compression. All other SIDs allocated from this locator with LNL=16, FL=16, AL=128-LBL-LNFL, and the value of the Argument is all 0 are also marked as using 16-bit compression. When receiving a SID advertisement for a REPLACE-CSID flavor SID with LNFL=32, AL=128-LBL-LNFL, and the value of the Argument is all 0, the SR source node marks both the SID and its locator as using 32-bit compression.

6.5. Upper-Layer Checksums

The Destination Address used in the IPv6 pseudo-header (Section 8.1 of [RFC8200]) is that of the ultimate destination.

At the SR source node, that address will be the Destination Address as it is expected to be received by the ultimate destination. When the last element in the compressed SID list is a CSID container, this address can be obtained from the last element in the uncompressed SID list or by repeatedly applying the segment behavior as described in Section 9.4. This applies regardless of whether an SRH is present in the IPv6 packet or omitted.

At the ultimate destination(s), that address will be in the Destination Address field of the IPv6 header.

7. Inter-Domain Compression

Some SRv6 traffic may need to cross multiple routing domains, such as different Autonomous Systems (ASes) or different routing areas within an SR domain. Different routing domains may use different addressing schema and Locator-Blocks.

A property of a CSID sequence is that all CSIDs in the sequence share the same Locator-Block. Therefore, a segment list that spans multiple routing domains using different Locator-Blocks may need a separate CSID sequence for each domain.

This section defines a solution to improve the efficiency of CSID compression in multi-domain environments by enabling a CSID sequence to combine CSIDs having different Locator-Blocks.

The solution leverages two new SRv6 endpoint behaviors, "Endpoint with SRv6 Locator-Block Swap" ("End.LBS" for short) and "Endpoint with L3 cross-connect and SRv6 Locator-Block Swap" ("End.XLBS" for short), that enable modifying the Locator-Block for the next CSID in the CSID sequence at the routing domain boundary.

7.1. End.LBS: Locator-Block Swap

The End.LBS behavior is a variant of the End behavior that modifies the Locator-Block of the active CSID sequence. This document defines the End.LBS behavior with the NEXT-CSID flavor and the End.LBS behavior with the REPLACE-CSID flavor.

An End.LBS SID is used to transition to a new Locator-Block when the routing domain boundary is on the SR segment endpoint node.

Each instance of an End.LBS SID is associated with a target Locator-Block B2/m, where B2 is an IPv6 address prefix and m is the associated prefix length. The original and target Locator-Blocks can have different prefix lengths as long as the new Destination Address formed by combining the target Locator-Block with the Locator-Node, Function, and Argument as described in the pseudocodes of Section 7.1.1 and Section 7.1.2 is a valid IPv6 address. The target Locator-Block is a local property of the End.LBS SID on the SR segment endpoint node.

> Note: a local SID property is an attribute associated with the

| SID when it is instantiated on the SR segment endpoint node.
| When the SR segment endpoint node identifies the destination
| address of a received packet as a locally instantiated SID, it
| also retrieves any local property associated with this SID.
| Other examples of local SID properties include the set of L3
| adjacencies of an End.X SID (Section 4.2 of [RFC8986]) and the
| lookup table of an End.DT6 SID (Section 4.6 of [RFC8986]).

The means by which an SR source node learns the target Locator-Block associated with an End.LBS SID are outside the scope of this document. As examples, it could be learned via configuration or signaled by a controller.

7.1.1. End.LBS with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.LBS SID with the NEXT-CSID flavor and associated with the target Locator-Block B2/m, the SR segment endpoint node applies the procedure specified in Section 4.1.1 with the lines N05 to N06 replaced as follows.

```
N05.1. Initialize an IPv6 address A equal to B2.
N05.2. Copy DA.Argument into the bits [m..(m+AL-1)] of A.
N06.   Copy A to the Destination Address of the IPv6 header.
```

7.1.2. End.LBS with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.LBS SID with the REPLACE-CSID flavor and associated with the target Locator-Block B2/m, the SR segment endpoint node applies the procedure specified in Section 4.2.1 with the line R20 replaced as follows.


```
R20.1. Initialize an IPv6 address A equal to B2.
R20.2. Write Segment List[Segments Left][DA.Arg.Index] into the bits
          [m..m+LNFL-1] of A.
R20.3. Write DA.Arg.Index into the bits
          [(128-ceiling(log_2(128/LNFL)))..127] of A.
R20.4. Copy A to the Destination Address of the IPv6 header.
```

7.2. End.XLBS: L3 Cross-Connect and Locator-Block Swap

The End.XLBS behavior is a variant of the End.X behavior that modifies the Locator-Block of the active CSID sequence. This document defines the End.XLBS behavior with the NEXT-CSID flavor and the End.XLBS behavior with the REPLACE-CSID flavor.

An End.XLBS SID is used to transition to a new Locator-Block when the routing domain boundary is on a link adjacent to the SR segment endpoint node.

Each instance of an End.XLBS SID is associated with a target Locator-Block B2/m and a set, J, of one or more L3 adjacencies. The original and target Locator-Blocks can have different prefix lengths as long as the new Destination Address formed by combining the target Locator-Block with the Locator-Node, Function, and Argument as described in the pseudocodes of Section 7.2.1 and Section 7.2.2 is a valid IPv6 address. The target Locator-Block and set of adjacencies are local properties of the End.XLBS SID on the SR segment endpoint node.

The means by which an SR source node learns the target Locator-Block associated with an End.XLBS SID are outside the scope of this document. As examples, it could be learned via configuration or signaled by a controller.

7.2.1. End.XLBS with NEXT-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.XLBS SID with the NEXT-CSID flavor and associated with the target Locator-Block B2/m, the SR segment endpoint node applies the procedure specified in Section 4.1.2 with the lines N05 to N06 (of the pseudocode in Section 4.1.1) replaced as follows.

```
N05.1. Initialize an IPv6 address A equal to B2.
N05.2. Copy DA.Argument into the bits [m..(m+AL-1)] of A.
N06.   Copy A to the Destination Address of the IPv6 header.
```


7.2.2. End.XLBS with REPLACE-CSID

When processing an IPv6 packet that matches a FIB entry locally instantiated as an End.XLBS SID with the REPLACE-CSID flavor and associated with the target Locator-Block B2/m, the SR segment endpoint node applies the procedure specified in Section 4.2.2 with the line R20 (of the pseudocode in Section 4.2.1) replaced as follows.

```
R20.1. Initialize an IPv6 address A equal to B2.
R20.2. Write Segment List[Segments Left][DA.Arg.Index] into the bits
          [m..m+LNFL-1] of A.
R20.3. Write DA.Arg.Index into the bits
          [(128-ceiling(log_2(128/LNFL)))..127] of A.
R20.4. Copy A to the Destination Address of the IPv6 header.
```

8. Control Plane

Section 8 of [RFC8986] provides an overview of the control plane protocols used for signaling of the SRv6 endpoint behaviors introduced by that document, including the base SRv6 endpoint behaviors that are extended in the present document.

The CSID-flavored behaviors introduced by this document are advertised in the same manner as their base SRv6 endpoint behaviors using the SRv6 extensions for various routing protocols, such as

* IS-IS [RFC9352]

* OSPFv3 [RFC9513]

* BGP [RFC9252], [RFC9514], [I-D.ietf-idr-sr-policy-safi]

* BGP-LS [I-D.ietf-idr-bgp-ls-sr-policy]

* PCEP [RFC9603]

The SR segment endpoint node MUST set the SID Argument bits to 0 when advertising a locally instantiated SID of this document in the routing protocol (e.g., IS-IS [RFC9352], OSPF [RFC9513], or BGP-LS [RFC9514]).

Signaling the SRv6 SID Structure is REQUIRED for all the SIDs introduced in this document. It is used by an SR source node to compress a SID list as described in Section 6. The node initiating the SID advertisement MUST set the length values in the SRv6 SID Structure to match the format of the SID on the SR segment endpoint node. For example, for a SID of this document instantiated from a /48 SRv6 SID block and a /64 Locator, and having a 16-bit Function, the SRv6 SID Structure advertisement carries the following values.

* Locator-Block length: 48

* Locator-Node length: 16

* Function length: 16

* Argument length: 48 (= 128-48-16-16)

A local CSID may be advertised in the control plane individually and/or in combination with a global CSID instantiated on the same SR segment endpoint node, with the End behavior, and the same Locator-Block and flavor as the local CSID. A combined global and local CSID is advertised as follows.

* The SID Locator-Block is that shared by the global and local CSIDs

* The SID Locator-Node is that of global CSID

* The SID Function is that of the local CSID

* The SID Argument length is equal to 128-LBL-LNL-FL and the SID Argument value is 0

* All other attributes of the SID (e.g., SRv6 endpoint behavior or algorithm) are those of the local CSID

The local CSID combined advertisement is needed in particular for control plane protocols mandating that the SID is a subnet of a locator advertised in the same protocol (e.g., Section 8 of [RFC9352] and Section 9 of [RFC9513] for advertising Adjacency SIDs in IS-IS and OSPFv3, respectively).

For a segment list computed by a controller and signaled to an SR source node (e.g., via BGP [I-D.ietf-idr-sr-policy-safi] or PCEP [RFC9603]), the controller provides the ordered segment list comprising the uncompressed SIDs, with their respective behavior and structure, to the SR source node. The SR source node may then compress the SID list as described in Section 6.

When a node that does not support this specification receives an advertisement of a SID of this document, it handles it as described in the corresponding control plane specification (e.g., Sections 7.2, 8.1, and 8.2 of [RFC9352], Sections 8, 9.1, and 9.2 of [RFC9513], and Section 3.1 of [RFC9252]).

9. Operational Considerations

9.1. Flavor, Block, and CSID Length

SRv6 is intended for use in a variety of networks that require different prefix lengths and SID numbering spaces. Each of the two flavors introduced in this document comes with its own recommendations for Locator-Block and CSID length, as specified in Section 4.1 and Section 4.2. These flavors are best suited for different environments, depending on the requirements of the network. For instance, larger CSID lengths may be more suitable for networks requiring ample SID numbering space, while smaller CSID lengths are better for compression efficiency. The two compression flavors allow the compressed segment list encoding to adapt to a range of requirements, with support for multiple compression levels. Network operators can choose the flavor that best suits their use case, deployment design, and network scale.

Both CSID flavors can coexist in the same SR domain, on the same SR segment endpoint node, and even in the same segment list. However, operators should generally avoid instantiating SIDs of different CSID flavors within the same routing domain or Locator-Block since these SIDs have different length and allocation recommendations (see Section 4.1, Section 4.2, and Section 9.2). In a multi-domain deployment, different flavors may be used in different routing domains of the SR domain.

A deployment should use consistent Locator-Block lengths and CSID lengths for all SIDs within a routing domain. Heterogeneous lengths, while possible, may impact the compression efficiency.

The compressed segment list encoding works with various Locator-Block allocations. For example, each routing domain within the SR domain can be allocated a /48 Locator-Block from a global IPv6 block available to the operator, or from a prefix allocated to SRv6 SIDs as discussed in Section 5 of [RFC9602].

9.2. GIB/LIB Usage

GIB and LIB usage is a local implementation and/or configuration decision, however, some guidelines for determining usage for specific SRv6 endpoint behaviors and recommendations are provided.

The GIB number space is shared among all SR segment endpoint nodes using SRv6 locators under a Locator-Block space. The more SIDs assigned from this space, per node, the faster it is exhausted. Therefore, its use is prioritized for global segments, such as SIDs that identify a node.

The LIB number space is unique per node. Each node can fully utilize the entire LIB number space without consideration of assignments at other nodes. Therefore, its use is prioritized for local segments, such as SIDs that identify services (of which there may be many) at nodes, cross-connects, or adjacencies.

While a longer CSID length permits more flexibility in which SRv6 endpoint behaviors may be assigned from the GIB; it also reduces the compression efficiency.

Given the previous Locator-Block and CSID length recommendations, the following GIB/LIB usage is recommended:

* NEXT-CSID:

  - GIB: End

  - LIB: End.X, End.T, End.DT4/6/46/2U/2M, End.DX4/6/2/2V (including large-scale pseudowire), End.B6.Encaps, End.B6.Encaps.Red, End.BM, End.LBS, and End.XLBS

* REPLACE-CSID:

  - GIB: End, End.X, End.T, End.DT4/6/46/2U/2M, End.DX4/6/2/2V, End.B6.Encaps, End.B6.Encaps.Red, End.BM, End.LBS, and End.XLBS

  - LIB: End.DX2/2V for large-scale pseudowire

Any other allocation is possible but may lead to a suboptimal use of the CSID numbering space.

9.3. Pinging a SID

An SR source node may ping an SRv6 SID by sending an ICMPv6 echo request packet destined to the SRv6 SID. The SR source node may ping the target SID with a SID list comprising only that target SID, or with a longer one that comprises two or more SIDs. In that case, the target SID is the last element in the SID list. This operation is illustrated in Appendix A.1.2 of [RFC9259].

When pinging a SID of this document the SR source node MUST construct the IPv6 packet as described in Section 6, including computing the ICMPv6 checksum as described in Section 6.5.

In particular, when pinging a SID of this document with a SID list comprising only the target SID, the SR source node places the SID with Argument value 0 in the destination address of the ICMPv6 echo request and computes the ICMPv6 checksum using this SID as the destination address in the IPv6 pseudo-header. The Argument value 0 allows the SID SR segment endpoint node (Section 4) to identify itself as the ultimate destination of the packet and process the ICMPv6 payload. Therefore, any existing IPv6 ping implementation can originate ICMP echo requests to a NEXT-CSID or REPLACE-CSID flavor SID with a SID list comprising only the target SID, provided that the user ensures that the SID Argument is 0.

9.4. ICMP Error Processing

When an IPv6 node encounters an error while processing a packet, it may report that error by sending an IPv6 error message to the packet source with an enclosed copy of the invoking packet. For the source of an invoking packet to process the ICMP error message, the ultimate destination address of the IPv6 header may be required.

Section 5.4 of [RFC8754] defines the logic that an SR source node follows to determine the ultimate destination of an invoking packet containing an SRH.

For an SR source node that supports the compressed segment list encoding defined in this document, the logic to determine the ultimate destination is generalized as follows.

* If the destination address of the invoking IPv6 packet matches a known SRv6 SID, modify the invoking IPv6 packet by applying the SRv6 endpoint behavior associated with the matched SRv6 SID;

* Repeat until the application of the SRv6 endpoint behavior would result in the processing of the upper-layer header.

The destination address of the resulting IPv6 packet may be used as the ultimate destination of the invoking IPv6 packet.

Since the SR source node that needs to determine the ultimate destination is the same node that originally built the SID list in the invoking packet, it can perform this operation for all the SIDs in the packet.

10. Implementation Status

This section is to be removed before publishing as an RFC.

RFC-Editor: Please clean up the references cited by this section before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

This section is provided in compliance with the SPRING working group policies ([SPRING-WG-POLICIES]).

10.1. Cisco Systems

Cisco Systems reported the following implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1) and the SR source node efficient SID list encoding (Section 6) for NEXT-CSID flavor SIDs. These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities.

* Cisco NCS 540 Series routers running IOS XR 7.3.x or above [IMPL-CISCO-NCS540]

* Cisco NCS 560 Series routers running IOS XR 7.6.x or above [IMPL-CISCO-NCS560]

* Cisco NCS 5500 Series routers running IOS XR 7.3.x or above [IMPL-CISCO-NCS5500]

* Cisco NCS 5700 Series routers running IOS XR 7.5.x or above [IMPL-CISCO-NCS5700]

* Cisco 8000 Series routers running IOS XR 7.5.x or above [IMPL-CISCO-8000]

* Cisco ASR 9000 Series routers running IOS XR 7.5.x or above [IMPL-CISCO-ASR9000]

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID flavor.

This report was last updated on January 11, 2023.

10.2. Huawei Technologies

Huawei Technologies reported the following implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities.

* Huawei ATN8XX,ATN910C,ATN980B routers running VRPV800R021C00 or above.

* Huawei CX600-M2 routers running VRPV800R021C00 or above.

* Huawei NE40E,ME60-X1X2,ME60-X3X8X16 routers running VRPV800R021C00 or above.

* Huawei NE5000E,NE9000 routers running VRPV800R021C00 or above.

* Huawei NCE-IP Controller running V1R21C00 or above.

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the REPLACE-CSID flavor.

This report was last updated on January 11, 2023.

10.3. Nokia

Nokia reported the following implementations ([IMPL-NOKIA-20.10]) of the SR segment endpoint node NEXT-CSID flavor (Section 4.1). These are used as part of its shortest path forwarding (in algorithm 0 and Flex-Algo), remote and TI-LFA repair tunnel, and Traffic Engineering functionalities.

* Nokia 7950 XRS 20/20e routers running SROS Release 22.10 or above
* Nokia 7750 SR-12e routers running SROS Release 22.10 or above
* Nokia 7750 SR-7/12 routers running SROS Release 22.10 or above
* Nokia 7750 SR-7s/14s routers running SROS Release 22.10 or above
* Nokia 7750 SR-1/1s/2s routers running SROS Release 22.10 or above

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID flavor.

This report was last updated on February 3, 2023.

10.4. Arrcus

Arrcus reported the following implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1). These are used as part of its SRv6 shortest path forwarding (in algorithm 0 and Flex-Algo), TI-LFA, micro-loop avoidance and Traffic Engineering functionalities.

* Arrcus running on Ufi Space routers S9510-28DC, S9710-76D, S9600-30DX and S9700-23D with ArcOS v5.2.1 or above
* Arrcus running n Ufi Space routers S9600-72XC and S9700-53DX with ArcOS v5.1.1D or above
* Arrcus running on Quanta router IXA and IXAE with ArcOS v5.1.1D or above

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID flavor.

This report was last updated on March 11, 2023.

10.5. Juniper Networks

Juniper Networks reported the following implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1). These are used as part of its SRv6 shortest path forwarding (in algorithm 0 and Flex-Algo), TI-LFA, micro-loop avoidance, and Traffic Engineering functionalities.

Juniper release 23.3 onwards supports this functionality.

At the time of this report, all the implementations listed above are in development and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID flavor.

This report was last updated on May 30, 2023.

10.6. Marvell

Marvell reported support in the Marvell Prestera Packet Processor for the SR segment endpoint node NEXT-CSID flavor (Section 4.1) and REPLACE-CSID flavor (Section 4.2).

This report was last updated on February 15, 2023.

10.7. Broadcom

Broadcom reported the following implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1) and REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities. All implementation of the following list is in general availability for customers using BCM SDK 6.5.26 or above.

* 88850 (Jericho2c+) series

* 88690 (Jericho2) series

* 88800 (Jericho2c) series

* 88480 (Qunran2a) series

* 88280 (Qunran2u) series

* 88295 (Qunran2n) series

* 88830 (Jericho2x) series

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID and REPLACE-CSID flavors.

For 78900 (Tomahawk) series-related support, please contact the Broadcom team.

This report was last updated on February 21, 2023.

10.8. ZTE Corporation

ZTE Corporation reported the following implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities.

* ZTE M6000-18S(BRAS), M6000-8S Plus(BRAS) routers running V5.00.10.09 or above.

* ZTE M6000-18S(SR), M6000-8S Plus(SR) routers running V5.00.10.80 or above.

* ZTE T8000-18 routers running V5.00.10.07 or above.

This report was last updated on March 29, 2023.

10.9. New H3C Technologies

New H3C Technologies reported the following implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities.

* H3C CR16000-F, SR8800-X routers running Version 7.1.075 or above.

* H3C CR18000, CR19000 routers running Version 7.1.071 or above.

This report was last updated on March 29, 2023.

10.10. Ruijie Network

Ruijie Network reported the following implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities.

* RUIJIE RG-N8018-R, RG-N8010-R routers running N8000-R_RGOS 12.8(3)B0801 or above.

This report was last updated on March 29, 2023.

10.11. Ciena

Ciena reported the following implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1). These are used as part of its shortest path forwarding (in algorithm 0 and Flex-Algo), remote and TI-LFA repair tunnel, and Traffic Engineering functionalities.

The following platforms support implementation of the above.

* Ciena 5162, 5164, 5166, 5168 routers running SAOS 10.10 or above

* Ciena 8110, 8112, 8190 routers running SAOS 10.10 or above

At the time of this report, all the implementations listed above are in production and follow the specification in the latest version of this document, including all the "MUST" and "SHOULD" clauses for the NEXT-CSID flavor.

This report was last updated on February 6, 2024.

10.12. Centec

Centec reported the following implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2). These are used as part of its SRv6 TI-LFA, micro-loop avoidance, and traffic engineering functionalities. All implementation of the following list is in general availability for customers using Centec SDK 5.6.8 or above.

* CTC7132 (TsingMa) Series

* CTC8180 (TsingMa.MX) Series

This report was last updated on February 14, 2024.

10.13. Open-Source

The authors found the following open-source implementations of the SR segment endpoint node NEXT-CSID flavor (Section 4.1).

* The Linux kernel, version 6.1 [IMPL-OSS-LINUX]

* The Software for Open Networking in the Cloud (SONiC), version 202212 [IMPL-OSS-SONIC], and Switch Abstraction Interface (SAI), version 1.9.0 [IMPL-OSS-SAI]

* The Vector Packet Processor (VPP), version 20.05 [IMPL-OSS-VPP]

* A generic P4 implementation [IMPL-OSS-P4]

The authors found the following open-source implementations of the SR segment endpoint node REPLACE-CSID flavor (Section 4.2).

* ONOS and P4 Programmable Switch based [IMPL-OSS-ONOS]

* Open SRv6 Project [IMPL-OSS-OPEN-SRV6]

This section was last updated on January 11, 2023.

10.14. Interoperability Reports

10.14.1. EANTC 2024

In April 2024, the European Advanced Networking Test Center (EANTC) successfully validated multiple implementations of SRv6 NEXT-CSID flavor (a.k.a., SRv6 uSID) [EANTC-24].

The participating vendors included Arista, Ciena, Cisco, Ericsson, H3C, Huawei, Juniper, Keysight, Nokia, and ZTE.

10.14.2. Bell Canada / Ciena 2023

Bell Canada is currently evaluating interoperability between Ciena and Cisco implementations of the NEXT-CSID flavor defined in this document. Further information will be added to this section when the evaluation is complete.

10.14.3. EANTC 2023

In April 2023, the European Advanced Networking Test Center (EANTC) successfully validated multiple implementations of SRv6 NEXT-CSID flavor (a.k.a., SRv6 uSID) [EANTC-23].

The participating vendors included Arista, Arrcus, Cisco, Huawei, Juniper, Keysight, Nokia, and Spirent.

10.14.4. China Mobile 2020

In November 2020, China Mobile successfully validated multiple interoperable implementations of the NEXT-CSID and REPLACE-CSID flavors defined in this document.

This testing covered two different implementations of the SRv6 endpoint flavors defined in this document:

* Hardware implementation in Cisco ASR 9000 running IOS XR

* Software implementation in Cisco IOS XRv9000 virtual appliance

* Hardware implementation in Huawei NE40E and NE5000E running VRP

The interoperability testing consisted of a packet flow sent by an SR source node N0 via an SR traffic engineering policy with a segment list <S1, S2, S3, S4, S5, S6, S7>, where S1..S7 are SIDs instantiated on SR segment endpoint nodes N1..N7, respectively.

```
N0 --- N1 --- N2 --- N3 --- N4 --- N5 --- N6 --- N7
      (S1)   (S2)   (S3)   (S4)   (S5)   (S6)   (S7)
```

* N0 is a generic packet generator.

* N1, N2, and N3 are Huawei routers.

* N4, N5, and N6 are Cisco routers.

* N7 is a generic traffic generator acting as a packet receiver.

The SR source node N0 steers the packets onto the SR policy by setting the IPv6 destination address and creating an SRH (as described in Section 4.1 of [RFC8754]) using a compressed segment list encoding. The length of the compressed segment list encoding varies for each scenario.

All SR segment endpoint nodes execute a variant of the End behavior: regular End behavior (as defined in Section 4.1 of [RFC8986]), End behavior with NEXT-CSID flavor, and End behavior with REPLACE-CSID flavor. The variant being used at each SR segment endpoint node varies for each scenario.

The interoperability was validated for the following scenarios:

Scenario 1:

* S1 and S2 are associated with the End behavior with the REPLACE-CSID flavor

* S3 is associated with the regular End behavior (no flavor)

* S4, S5, and S6 are associated with the End behavior with the NEXT-CSID flavor

* The SR source node imposes a compressed segment list encoding of 3 SIDs.

Scenario 2:

* S1, S2..., S6 are associated with the End behavior with the NEXT-CSID flavor

* The SR source node imposes a compressed segment list encoding of 2 SIDs.

Scenario 3:

* S1, S2..., S6 are associated with the End behavior with the REPLACE-CSID flavor

* The SR source node imposes a compressed segment list encoding of 3 SIDs.

11. Applicability to other SRv6 Endpoint Behaviors

Future documents may extend the applicability of the NEXT-CSID and REPLACE-CSID flavors to other SRv6 endpoint behaviors.

For an SRv6 endpoint behavior that can be used before the last position of a segment list, a CSID flavor is defined by reproducing the same logic as described in Section 4.1 and Section 4.2 of this document to determine the next SID in the SID list.

12. Security Considerations

Section 8 of [RFC8402] discusses the security considerations for Segment Routing.

Section 5 of [RFC8754] describes the intra-SR-domain deployment model and how to secure it. Section 7 of [RFC8754] describes the threats applicable to SRv6 and how to mitigate them.

Section 9 of [RFC8986] discusses the security considerations applicable to the SRv6 network programming framework, as well as the SR source node and SR segment endpoint node behaviors that it defines.

This document introduces two new flavors for some of the SRv6 endpoint behaviors defined in [RFC8986] and a method by which an SR source node may leverage the SIDs of these flavors to produce a compressed segment list encoding.

This document also introduces two new SRv6 endpoint behaviors, End.LBS and End.XLBS, to preserve the efficiency of CSID compression in multi-domain environments.

An SR source node constructs an IPv6 packet with a compressed segment list encoding as defined in Sections 3.1 and 4.1 of [RFC8754] and Section 5 of [RFC8986]. The paths that an SR source node may enforce using a compressed segment list encoding are the same, from a topology and service perspective, as those that an SR source node could enforce using the SIDs of [RFC8986].

An SR segment endpoint node processes an IPv6 packet matching a locally instantiated SID as defined in [RFC8986], with the pseudocode modifications in Section 4 of this document. These modifications change how the SR segment endpoint node determines the next SID in the packet, but not the semantic of either the active or the next SID. For example, an adjacency segment instantiated with the End.X behavior remains an adjacency segment regardless of whether it uses the base End.X behavior defined in Section 4.2 of [RFC8986] or a CSID flavor of that behavior. This document does not introduce any new SID semantic.

Any other transit node processes the packet as described in Section 4.2 of [RFC8754].

This document defines a new method of encoding the SIDs inside a SID list at the SR source node (Section 6) and decoding them at the SR segment endpoint node (Section 4 and Section 7), but it does not change how the SID list itself is encoded in the IPv6 packet nor the semantic of any segment that it comprises. Therefore, this document is subject to the same security considerations that are discussed in [RFC8402], [RFC8754], and [RFC8986].

13. IANA Considerations

13.1. SRv6 Endpoint Behaviors

This I-D. requests the IANA to update the reference of the following registrations from the "SRv6 Endpoint Behaviors" registry under the top-level "Segment Routing" registry-group (https://www.iana.org/assignments/segment-routing/) with the RFC number of this document once it is published, and transfer change control to the IETF.

Value	Description	Reference
43	End with NEXT-CSID	This I-D.
44	End with NEXT-CSID & PSP	This I-D.
45	End with NEXT-CSID & USP	This I-D.
46	End with NEXT-CSID, PSP & USP	This I-D.
47	End with NEXT-CSID & USD	This I-D.
48	End with NEXT-CSID, PSP & USD	This I-D.
49	End with NEXT-CSID, USP & USD	This I-D.

50	End with NEXT-CSID, PSP, USP & USD	This I-D.
52	End.X with NEXT-CSID	This I-D.
53	End.X with NEXT-CSID & PSP	This I-D.
54	End.X with NEXT-CSID & USP	This I-D.
55	End.X with NEXT-CSID, PSP & USP	This I-D.
56	End.X with NEXT-CSID & USD	This I-D.
57	End.X with NEXT-CSID, PSP & USD	This I-D.
58	End.X with NEXT-CSID, USP & USD	This I-D.
59	End.X with NEXT-CSID, PSP, USP & USD	This I-D.
85	End.T with NEXT-CSID	This I-D.
86	End.T with NEXT-CSID & PSP	This I-D.

87	End.T with NEXT-CSID & USP	This I-D.
88	End.T with NEXT-CSID, PSP & USP	This I-D.
89	End.T with NEXT-CSID & USD	This I-D.
90	End.T with NEXT-CSID, PSP & USD	This I-D.
91	End.T with NEXT-CSID, USP & USD	This I-D.
92	End.T with NEXT-CSID, PSP, USP & USD	This I-D.
93	End.B6.Encaps with NEXT-CSID	This I-D.
94	End.B6.Encaps.Red with NEXT-CSID	This I-D.
95	End.BM with NEXT-CSID	This I-D.
96	End.LBS with NEXT-CSID	This I-D.
97	End.XLBS with NEXT-CSID	This I-D.
101	End with REPLACE-CSID	This I-D.
102	End with REPLACE-CSID & PSP	This I-D.
103	End with REPLACE-CSID & USP	This I-D.
104	End with REPLACE-CSID, PSP & USP	This I-D.
105	End.X with REPLACE-CSID	This I-D.
106	End.X with REPLACE-CSID & PSP	This I-D.
107	End.X with REPLACE-CSID & USP	This I-D.
108	End.X with REPLACE-CSID, PSP & USP	This I-D.

109	End.T with REPLACE-CSID	This I-D.
110	End.T with REPLACE-CSID & PSP	This I-D.
111	End.T with REPLACE-CSID & USP	This I-D.
112	End.T with REPLACE-CSID, PSP & USP	This I-D.
114	End.B6.Encaps with REPLACE-CSID	This I-D.

115	End.BM with REPLACE-CSID	This I-D.
116	End.DX6 with REPLACE-CSID	This I-D.
117	End.DX4 with REPLACE-CSID	This I-D.
118	End.DT6 with REPLACE-CSID	This I-D.
119	End.DT4 with REPLACE-CSID	This I-D.
120	End.DT46 with REPLACE-CSID	This I-D.
121	End.DX2 with REPLACE-CSID	This I-D.
122	End.DX2V with REPLACE-CSID	This I-D.
123	End.DT2U with REPLACE-CSID	This I-D.
124	End.DT2M with REPLACE-CSID	This I-D.
127	End.B6.Encaps.Red with REPLACE-CSID	This I-D.
128	End with REPLACE-CSID & USD	This I-D.
129	End with REPLACE-CSID, PSP & USD	This I-D.
130	End with REPLACE-CSID, USP & USD	This I-D.
131	End with REPLACE-CSID, PSP, USP & USD	This I-D.
132	End.X with REPLACE-CSID & USD	This I-D.
133	End.X with REPLACE-CSID, PSP & USD	This I-D.
134	End.X with REPLACE-CSID, USP & USD	This I-D.
135	End.X with REPLACE-CSID, PSP, USP & USD	This I-D.
136	End.T with REPLACE-CSID & USD	This I-D.
137	End.T with REPLACE-CSID, PSP & USD	This I-D.
138	End.T with REPLACE-CSID, USP & USD	This I-D.
139	End.T with REPLACE-CSID, PSP, USP & USD	This I-D.
140	End.LBS with REPLACE-CSID	This I-D.

141	End.XLBS with REPLACE-CSID	This I-D.

Table 1: Registration List

14. Acknowledgements

The authors would like to thank Kamran Raza, Xing Jiang, YuanChao Su, Han Li, Yisong Liu, Martin Vigoureux, Joel Halpern, and Tal Mizrahi for their insightful feedback and suggestions.

The authors would also like to thank Andrew Alston, Linda Dunbar, Adrian Farrel, Boris Hassanov, Alvaro Retana, and Gunter Van de Velde for their thorough review of this document.

15. References

15.1. Normative References

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

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

15.2. Informative References

[EANTC-23] European Advanced Networking Test Center (EANTC), "Multi-Vendor MPLS SDN Interoperability Test Report 2023", 18 April 2023, <https://eantc.de/wp-content/uploads/2023/04/EANTC-InteropTest2023-TestReport.pdf>.

[EANTC-24] European Advanced Networking Test Center (EANTC), "Multi-

Vendor MPLS SDN Interoperability Test Report 2024", April 2024, <https://eantc.de/wp-content/uploads/2023/12/EANTC-MPLSSDNInterop2024-TestReport-v1.3.pdf>.

[GKP94] Graham, R., Knuth, D., and O. Patashnik, "Concrete Mathematics: A Foundation for Computer Science", ISBN 9780201558029, 1994.

[I-D.ietf-idr-bgp-ls-sr-policy]
Previdi, S., Talaulikar, K., Dong, J., Gredler, H., and J. Tantsura, "Advertisement of Segment Routing Policies using BGP Link-State", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-ls-sr-policy-10, 9 December 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-ls-sr-policy-10>.

[I-D.ietf-idr-sr-policy-safi]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-sr-policy-safi-11, 31 January 2025, <https://datatracker.ietf.org/api/v1/doc/document/draft-ietf-idr-sr-policy-safi/>.

[IMPL-CISCO-8000]
Cisco Systems, "Segment Routing Configuration Guide for Cisco 8000 Series Routers", 4 November 2022, <https://www.cisco.com/c/en/us/td/docs/iosxr/cisco8000/segment-routing/75x/b-segment-routing-cg-cisco8000-75x/configuring-segment-routing-over-ipv6-srv6-micro-sids.html>.

[IMPL-CISCO-ASR9000]
Cisco Systems, "Segment Routing Configuration Guide for Cisco ASR 9000 Series Routers", 6 November 2022, <https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k-r7-5/segment-routing/configuration/guide/b-segment-routing-cg-asr9000-75x/configure-srv6-micro-sid.html>.

[IMPL-CISCO-NCS540]
Cisco Systems, "Segment Routing Configuration Guide for Cisco NCS 540 Series Routers", 2 November 2022, <https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5xx/segment-routing/73x/b-segment-routing-cg-73x-ncs540/configure-srv6.html>.

[IMPL-CISCO-NCS5500]
Cisco Systems, "Segment Routing Configuration Guide for Cisco NCS 5500 Series Routers", 6 November 2022, <https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5500/segment-routing/73x/b-segment-routing-cg-ncs5500-73x/configure-srv6-micro-sid.html>.

[IMPL-CISCO-NCS560]
Cisco Systems, "Segment Routing Configuration Guide for Cisco NCS 560 Series Routers", 14 October 2022, <https://www.cisco.com/c/en/us/td/docs/iosxr/ncs560/segment-routing/76x/b-segment-routing-cg-76x-ncs560/m-configure-srv6-usid-ncs5xx.html>.

[IMPL-CISCO-NCS5700]
Cisco Systems, "Segment Routing Configuration Guide for Cisco NCS 5700 Series Routers", 6 November 2022, <https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5500/segment-routing/75x/b-segment-routing-cg-ncs5500-75x/configure-srv6-micro-sid.html>.

[IMPL-NOKIA-20.10]
Nokia, "Segment Routing and PCE User Guide", December 2022, <https://documentation.nokia.com/sr/22-10/books/Segment%20Routing%20and%20PCE%20User%20Guide/segment-rout-with-ipv6-data-plane-srv6.html>.

[IMPL-OSS-LINUX]
Abeni, P., "Add NEXT-CSID support for SRv6 End behavior", 20 September 2022, <https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git/commit/?id=cec9d59e89362809f17f2d854faf52966216da13>.

[IMPL-OSS-ONOS]
Open Networking Foundation, "Stratum CMCC G-SRv6 Project", 24 March 2021, <https://wiki.opennetworking.org/display/COM/Stratum+CMCC+G-SRv6+Project>.

[IMPL-OSS-OPEN-SRV6]
"Open SRv6 Project", n.d., <http://opensrv6.org.cn/en/srv6-2/>.

[IMPL-OSS-P4]
Salsano, S. and A. Tulumello, "SRv6 uSID (micro SID) implementation on P4", 3 January 2021, <https://github.com/netgroup/p4-srv6-usid>.

[IMPL-OSS-SAI]
Agrawal, A., "Added new behaviors to support uSID instruction", 8 June 2021, <https://github.com/opencomputeproject/SAI/pull/1231/commits/02e58d95ad966ca9efc24eb9e0c0fa10b21de2a4>.

[IMPL-OSS-SONIC]
Shah, S. and R. Sudarshan, "SONiC uSID", 21 August 2022, <https://github.com/sonic-net/SONiC/blob/master/doc/srv6/SRv6_uSID.md>.

[IMPL-OSS-VPP]
FD.io, "Srv6 cli reference", n.d., <https://s3-docs.fd.io/vpp/23.02/cli-reference/clis/clicmd_src_vnet_srv6.html>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC9252] Dawra, G., Ed., Talaulikar, K., Ed., Raszuk, R., Decraene, B., Zhuang, S., and J. Rabadan, "BGP Overlay Services Based on Segment Routing over IPv6 (SRv6)", RFC 9252, DOI 10.17487/RFC9252, July 2022, <https://www.rfc-editor.org/info/rfc9252>.

[RFC9259] Ali, Z., Filsfils, C., Matsushima, S., Voyer, D., and M. Chen, "Operations, Administration, and Maintenance (OAM) in Segment Routing over IPv6 (SRv6)", RFC 9259, DOI 10.17487/RFC9259, June 2022, <https://www.rfc-editor.org/info/rfc9259>.

[RFC9350] Psenak, P., Ed., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", RFC 9350, DOI 10.17487/RFC9350, February 2023, <https://www.rfc-editor.org/info/rfc9350>.

[RFC9352] Psenak, P., Ed., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over the IPv6 Data Plane", RFC 9352, DOI 10.17487/RFC9352, February 2023, <https://www.rfc-editor.org/info/rfc9352>.

[RFC9513] Li, Z., Hu, Z., Talaulikar, K., Ed., and P. Psenak, "OSPFv3 Extensions for Segment Routing over IPv6 (SRv6)", RFC 9513, DOI 10.17487/RFC9513, December 2023, <https://www.rfc-editor.org/info/rfc9513>.

[RFC9514] Dawra, G., Filsfils, C., Talaulikar, K., Ed., Chen, M., Bernier, D., and B. Decraene, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing over IPv6 (SRv6)", RFC 9514, DOI 10.17487/RFC9514, December 2023, <https://www.rfc-editor.org/info/rfc9514>.

[RFC9602] Krishnan, S., "Segment Routing over IPv6 (SRv6) Segment Identifiers in the IPv6 Addressing Architecture", RFC 9602, DOI 10.17487/RFC9602, October 2024, <https://www.rfc-editor.org/info/rfc9602>.

[RFC9603] Li, C., Ed., Kaladharan, P., Sivabalan, S., Koldychev, M., and Y. Zhu, "Path Computation Element Communication Protocol (PCEP) Extensions for IPv6 Segment Routing", RFC 9603, DOI 10.17487/RFC9603, July 2024, <https://www.rfc-editor.org/info/rfc9603>.

[SPRING-WG-POLICIES] SPRING Working Group Chairs, "SPRING Working Group Policies", 14 October 2022, <https://wiki.ietf.org/en/group/spring/WG_Policies>.

Appendix A. Complete pseudocodes

The content of this section is purely informative rendering of the pseudocodes of [RFC8986] with the modifications in this document. This rendering may not be used as a reference.

A.1. End with NEXT-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End SID with the NEXT-CSID flavor:

```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address
           with Code 0 (Hop limit exceeded in transit),
           interrupt packet processing, and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.   Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.
N09. }
S02. If (Segments Left == 0) {
S03.   Stop processing the SRH, and proceed to process the next
         header in the packet, whose type is identified by
         the Next Header field in the routing header.
S04. }
S05. If (IPv6 Hop Limit <= 1) {
S06.   Send an ICMP Time Exceeded message to the Source Address
         with Code 0 (Hop limit exceeded in transit),
         interrupt packet processing, and discard the packet.
S07. }
S08. max_LE = (Hdr Ext Len / 2) - 1
S09. If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.   Send an ICMP Parameter Problem to the Source Address
         with Code 0 (Erroneous header field encountered)
         and Pointer set to the Segments Left field,
         interrupt packet processing, and discard the packet.
S11. }
S12. Decrement IPv6 Hop Limit by 1.
S13. Decrement Segments Left by 1.
S14. Update IPv6 DA with Segment List[Segments Left].
S15. Submit the packet to the egress IPv6 FIB lookup for
       transmission to the new destination.
```

Before processing the Upper-Layer header or any IPv6 extension header other than Hop-by-Hop or Destination Options of a packet matching a FIB entry locally instantiated as an End SID with the NEXT-CSID flavor:


```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
```

```
              interrupt packet processing and discard the packet.
   N04.   }
   N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
            Destination Address.
   N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
            zero.
   N07.   Decrement IPv6 Hop Limit by 1.
   N08.   Submit the packet to the egress IPv6 FIB lookup for
            transmission to the next destination.
   N09. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End SID with the NEXT-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
            with Code 4 (SR Upper-layer Header Error)
            and Pointer set to the offset of the Upper-Layer header,
            interrupt packet processing, and discard the packet.
   S05. }
```

A.2. End.X with NEXT-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.X SID with the NEXT-CSID flavor:


```
   N01. If (DA.Argument != 0) {
   N02.   If (IPv6 Hop Limit <= 1) {
   N03.     Send an ICMP Time Exceeded message to the Source Address
              with Code 0 (Hop limit exceeded in transit),
              interrupt packet processing, and discard the packet.
   N04.   }
   N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
            Destination Address.
   N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
            zero.
   N07.   Decrement IPv6 Hop Limit by 1.
   N08.   Submit the packet to the IPv6 module for transmission to the
            new destination via a member of J.
   N09. }
   S02. If (Segments Left == 0) {
   S03.   Stop processing the SRH, and proceed to process the next
            header in the packet, whose type is identified by
```

```
         the Next Header field in the routing header.
S04. }
S05. If (IPv6 Hop Limit <= 1) {
S06.   Send an ICMP Time Exceeded message to the Source Address
         with Code 0 (Hop limit exceeded in transit),
         interrupt packet processing, and discard the packet.
S07. }
S08. max_LE = (Hdr Ext Len / 2) - 1
S09. If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.   Send an ICMP Parameter Problem to the Source Address
         with Code 0 (Erroneous header field encountered)
         and Pointer set to the Segments Left field,
         interrupt packet processing, and discard the packet.
S11. }
S12. Decrement IPv6 Hop Limit by 1.
S13. Decrement Segments Left by 1.
S14. Update IPv6 DA with Segment List[Segments Left].
S15. Submit the packet to the IPv6 module for transmission
       to the new destination via a member of J.
```

Before processing the Upper-Layer header or any IPv6 extension header other than Hop-by-Hop or Destination Options of a packet matching a FIB entry locally instantiated as an End.X SID with the NEXT-CSID flavor:


```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.   Submit the packet to the IPv6 module for transmission to the
         new destination via a member of J.
N09. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.X SID with the NEXT-CSID flavor:

```
S01. If (Upper-Layer header type is allowed by local configuration) {
S02.   Proceed to process the Upper-Layer header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address
         with Code 4 (SR Upper-layer Header Error)
         and Pointer set to the offset of the Upper-Layer header,
         interrupt packet processing, and discard the packet.
S05. }
```

A.3. End.T with NEXT-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.T SID with the NEXT-CSID flavor:


```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address
           with Code 0 (Hop limit exceeded in transit),
           interrupt packet processing, and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.1. Set the packet's associated FIB table to T.
N08.2. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the new destination.
N09. }
S02. If (Segments Left == 0) {
S03.   Stop processing the SRH, and proceed to process the next
         header in the packet, whose type is identified by
         the Next Header field in the routing header.
S04. }
S05. If (IPv6 Hop Limit <= 1) {
S06.   Send an ICMP Time Exceeded message to the Source Address
         with Code 0 (Hop limit exceeded in transit),
         interrupt packet processing, and discard the packet.
S07. }
S08. max_LE = (Hdr Ext Len / 2) - 1
S09. If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.   Send an ICMP Parameter Problem to the Source Address
         with Code 0 (Erroneous header field encountered)
         and Pointer set to the Segments Left field,
         interrupt packet processing, and discard the packet.
S11. }
S12. Decrement IPv6 Hop Limit by 1.
S13. Decrement Segments Left by 1.
S14. Update IPv6 DA with Segment List[Segments Left].
S15.1. Set the packet's associated FIB table to T.
S15.2. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the new destination.
```

Before processing the Upper-Layer header or any IPv6 extension header other than Hop-by-Hop or Destination Options of a packet matching a FIB entry locally instantiated as an End.T SID with the NEXT-CSID flavor:

```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.1. Set the packet's associated FIB table to T.
N08.2. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the new destination.
N09. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.T SID with the NEXT-CSID flavor:

```
S01. If (Upper-Layer header type is allowed by local configuration) {
S02.   Proceed to process the Upper-Layer header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address
         with Code 4 (SR Upper-layer Header Error)
         and Pointer set to the offset of the Upper-Layer header,
         interrupt packet processing, and discard the packet.
S05. }
```

A.4. End.B6.Encaps with NEXT-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID with the NEXT-CSID flavor:

```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.1. Push a new IPv6 header with its own SRH containing B.
N08.2. Set the outer IPv6 SA to A.
N08.3. Set the outer IPv6 DA to the first SID of B.
N08.4. Set the outer Payload Length, Traffic Class, Flow Label,
         Hop Limit, and Next Header fields.
N08.5. Submit the packet to the egress IPv6 FIB lookup for
         transmission to the next destination.
N09. }
S02. If (Segments Left == 0) {
S03.   Stop processing the SRH, and proceed to process the next
         header in the packet, whose type is identified by
         the Next Header field in the routing header.
S04. }
S05. If (IPv6 Hop Limit <= 1) {
S06.   Send an ICMP Time Exceeded message to the Source Address
         with Code 0 (Hop limit exceeded in transit),
         interrupt packet processing, and discard the packet.
S07. }
S08. max_LE = (Hdr Ext Len / 2) - 1
S09. If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.   Send an ICMP Parameter Problem to the Source Address
         with Code 0 (Erroneous header field encountered)
         and Pointer set to the Segments Left field,
         interrupt packet processing, and discard the packet.
S11. }
S12. Decrement IPv6 Hop Limit by 1.
S13. Decrement Segments Left by 1.
S14. Update IPv6 DA with Segment List[Segments Left].
S15. Push a new IPv6 header with its own SRH containing B.
S16. Set the outer IPv6 SA to A.
S17. Set the outer IPv6 DA to the first SID of B.
S18. Set the outer Payload Length, Traffic Class, Flow Label,
       Hop Limit, and Next Header fields.
S19. Submit the packet to the egress IPv6 FIB lookup for
       transmission to the new destination.
```


Before processing the Upper-Layer header or any IPv6 extension header other than Hop-by-Hop or Destination Options of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID with the NEXT-CSID flavor:

```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
```

```
   N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
            zero.
   N07.   Decrement IPv6 Hop Limit by 1.
   N08.1. Push a new IPv6 header with its own SRH containing B.
   N08.2. Set the outer IPv6 SA to A.
   N08.3. Set the outer IPv6 DA to the first SID of B.
   N08.4. Set the outer Payload Length, Traffic Class, Flow Label,
            Hop Limit, and Next Header fields.
   N08.5. Submit the packet to the egress IPv6 FIB lookup for
            transmission to the next destination.
   N09. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID with the NEXT-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
            with Code 4 (SR Upper-layer Header Error)
            and Pointer set to the offset of the Upper-Layer header,
            interrupt packet processing, and discard the packet.
   S05. }
```

A.5. End.BM with NEXT-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.BM SID with the NEXT-CSID flavor:


```
   N01. If (DA.Argument != 0) {
   N02.   If (IPv6 Hop Limit <= 1) {
   N03.     Send an ICMP Time Exceeded message to the Source Address,
              Code 0 (Hop limit exceeded in transit),
              interrupt packet processing and discard the packet.
   N04.   }
   N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
            Destination Address.
   N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
            zero.
   N07.   Decrement IPv6 Hop Limit by 1.
   N08.1. Push the MPLS label stack for B.
   N08.2. Submit the packet to the MPLS engine for transmission.
   N09. }
   S02. If (Segments Left == 0) {
   S03.   Stop processing the SRH, and proceed to process the next
            header in the packet, whose type is identified by
            the Next Header field in the routing header.
   S04. }
   S05. If (IPv6 Hop Limit <= 1) {
   S06.   Send an ICMP Time Exceeded message to the Source Address
            with Code 0 (Hop limit exceeded in transit),
            interrupt packet processing, and discard the packet.
   S07. }
   S08. max_LE = (Hdr Ext Len / 2) - 1
   S09. If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
```

```
S10.   Send an ICMP Parameter Problem to the Source Address
         with Code 0 (Erroneous header field encountered)
         and Pointer set to the Segments Left field,
         interrupt packet processing, and discard the packet.
S11. }
S12. Decrement IPv6 Hop Limit by 1.
S13. Decrement Segments Left by 1.
S14. Update IPv6 DA with Segment List[Segments Left].
S15. Push the MPLS label stack for B.
S16. Submit the packet to the MPLS engine for transmission.
```

Before processing the Upper-Layer header or any IPv6 extension header other than Hop-by-Hop or Destination Options of a packet matching a FIB entry locally instantiated as an End.BM SID with the NEXT-CSID flavor:


```
N01. If (DA.Argument != 0) {
N02.   If (IPv6 Hop Limit <= 1) {
N03.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
N04.   }
N05.   Copy DA.Argument into the bits [LBL..(LBL+AL-1)] of the
         Destination Address.
N06.   Set the bits [(LBL+AL)..127] of the Destination Address to
         zero.
N07.   Decrement IPv6 Hop Limit by 1.
N08.1. Push the MPLS label stack for B.
N08.2. Submit the packet to the MPLS engine for transmission.
N09. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.BM SID with the NEXT-CSID flavor:

```
S01. If (Upper-Layer header type is allowed by local configuration) {
S02.   Proceed to process the Upper-Layer header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address
         with Code 4 (SR Upper-layer Header Error)
         and Pointer set to the offset of the Upper-Layer header,
         interrupt packet processing, and discard the packet.
S05. }
```

A.6. End with REPLACE-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End SID with the REPLACE-CSID flavor:

```
   S01. When an SRH is processed {
   S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
              Segment List[0][DA.Arg.Index-1] == 0)) {
   S03.     Stop processing the SRH, and proceed to process the next
              header in the packet, whose type is identified by
              the Next Header field in the routing header.
   S04.   }
   S05.   If (IPv6 Hop Limit <= 1) {
   S06.     Send an ICMP Time Exceeded message to the Source Address,
              Code 0 (Hop limit exceeded in transit),
              interrupt packet processing and discard the packet.
   S07.   }
   S08.   max_LE = (Hdr Ext Len / 2) - 1
   R01.   If (DA.Arg.Index != 0) {
   R02.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
   R03.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R04.     }
   R05.     Decrement DA.Arg.Index by 1.
   R06.     If (Segment List[Segments Left][DA.Arg.Index] == 0) {
   R07.       Decrement Segments Left by 1.
   R08.       Decrement IPv6 Hop Limit by 1.
   R09.       Update IPv6 DA with Segment List[Segments Left]
   R10.       Submit the packet to the egress IPv6 FIB lookup for
               transmission to the new destination.
   R11.     }
   R12.   } Else {
   R13.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
   R14.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R15.     }
   R16.     Decrement Segments Left by 1.
   R17.     Set DA.Arg.Index to (128/LNFL - 1).
   R18.   }
   R19.   Decrement IPv6 Hop Limit by 1.
   R20.   Write Segment List[Segments Left][DA.Arg.Index] into the bits
            [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
            header.
   R21.   Submit the packet to the egress IPv6 FIB lookup for
            transmission to the new destination.
   S16. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End SID with the REPLACE-CSID flavor:

```
S01. If (Upper-Layer header type is allowed by local configuration) {
S02.   Proceed to process the Upper-Layer header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address
          with Code 4 (SR Upper-layer Header Error)
          and Pointer set to the offset of the Upper-Layer header,
          interrupt packet processing, and discard the packet.
S05. }
```

A.7. End.X with REPLACE-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.X SID with the REPLACE-CSID flavor:


```
S01. When an SRH is processed {
S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
           Segment List[0][DA.Arg.Index-1] == 0)) {
S03.     Stop processing the SRH, and proceed to process the next
           header in the packet, whose type is identified by
           the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
```

```
              Code 0 (Hop limit exceeded in transit),
              interrupt packet processing and discard the packet.
   S07.   }
   S08.   max_LE = (Hdr Ext Len / 2) - 1
   R01.   If (DA.Arg.Index != 0) {
   R02.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
   R03.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R04.     }
   R05.     Decrement DA.Arg.Index by 1.
   R06.     If (Segment List[Segments Left][DA.Arg.Index] == 0) {
   R07.       Decrement Segments Left by 1.
   R08.       Decrement IPv6 Hop Limit by 1.
   R09.       Update IPv6 DA with Segment List[Segments Left]
   R10.       Submit the packet to the IPv6 module for transmission to
                the new destination via a member of J.
   R11.     }
   R12.   } Else {
   R13.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
   R14.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R15.     }
   R16.     Decrement Segments Left by 1.
   R17.     Set DA.Arg.Index to (128/LNFL - 1).
   R18.   }
   R19.   Decrement IPv6 Hop Limit by 1.
   R20.   Write Segment List[Segments Left][DA.Arg.Index] into the bits
            [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
            header.
   R21.   Submit the packet to the IPv6 module for transmission to the
            new destination via a member of J.
   S16. }
```


When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.X SID with the REPLACE-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
             with Code 4 (SR Upper-layer Header Error)
             and Pointer set to the offset of the Upper-Layer header,
             interrupt packet processing, and discard the packet.
   S05. }
```

A.8. End.T with REPLACE-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.T SID with the REPLACE-CSID flavor:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
           Segment List[0][DA.Arg.Index-1] == 0)) {
S03.     Stop processing the SRH, and proceed to process the next
           header in the packet, whose type is identified by
           the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
R01.   If (DA.Arg.Index != 0) {
R02.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
R03.       Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field,
             interrupt packet processing and discard the packet.
R04.     }
R05.     Decrement DA.Arg.Index by 1.
R06.     If (Segment List[Segments Left][DA.Arg.Index] == 0) {
R07.       Decrement Segments Left by 1.
R08.       Decrement IPv6 Hop Limit by 1.
R09.       Update IPv6 DA with Segment List[Segments Left]
R10.1.     Set the packet's associated FIB table to T.
R10.2.     Submit the packet to the egress IPv6 FIB lookup for
             transmission to the new destination.
R11.     }
R12.   } Else {
R13.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
R14.       Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field,
             interrupt packet processing and discard the packet.
```

```
   R15.     }
   R16.     Decrement Segments Left by 1.
   R17.     Set DA.Arg.Index to (128/LNFL - 1).
   R18.   }
   R19.   Decrement IPv6 Hop Limit by 1.
   R20.   Write Segment List[Segments Left][DA.Arg.Index] into the bits
            [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
            header.
   R21.1. Set the packet's associated FIB table to T.
   R21.2. Submit the packet to the egress IPv6 FIB lookup for
            transmission to the new destination.
   S16. }
```


When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.T SID with the REPLACE-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
             with Code 4 (SR Upper-layer Header Error)
             and Pointer set to the offset of the Upper-Layer header,
             interrupt packet processing, and discard the packet.
   S05. }
```

A.9. End.B6.Encaps with REPLACE-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID with the REPLACE-CSID flavor:

```
   S01. When an SRH is processed {
   S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
              Segment List[0][DA.Arg.Index-1] == 0)) {
   S03.     Stop processing the SRH, and proceed to process the next
              header in the packet, whose type is identified by
              the Next Header field in the routing header.
   S04.   }
   S05.   If (IPv6 Hop Limit <= 1) {
   S06.     Send an ICMP Time Exceeded message to the Source Address,
              Code 0 (Hop limit exceeded in transit),
              interrupt packet processing and discard the packet.
   S07.   }
   S08.   max_LE = (Hdr Ext Len / 2) - 1
   R01.   If (DA.Arg.Index != 0) {
   R02.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
   R03.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R04.     }
   R05.     Decrement DA.Arg.Index by 1.
   R06.     If (Segment List[Segments Left][DA.Arg.Index] == 0) {
   R07.       Decrement Segments Left by 1.
   R08.       Decrement IPv6 Hop Limit by 1.
   R09.       Update IPv6 DA with Segment List[Segments Left]
   R10.1.     Push a new IPv6 header with its own SRH containing B.
   R10.2.     Set the outer IPv6 SA to A.
   R10.3.     Set the outer IPv6 DA to the first SID of B.
   R10.4.     Set the outer Payload Length, Traffic Class, Flow Label,
                Hop Limit, and Next Header fields.
```

```
   R10.5.     Submit the packet to the egress IPv6 FIB lookup for
                transmission to the next destination.
   R11.     }
   R12.   } Else {
   R13.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
   R14.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R15.     }
   R16.     Decrement Segments Left by 1.
   R17.     Set DA.Arg.Index to (128/LNFL - 1).
   R18.   }
   R19.   Decrement IPv6 Hop Limit by 1.
   R20.   Write Segment List[Segments Left][DA.Arg.Index] into the bits
            [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
            header.
   R21.1. Push a new IPv6 header with its own SRH containing B.
   R21.2. Set the outer IPv6 SA to A.
   R21.3. Set the outer IPv6 DA to the first SID of B.
   R21.4. Set the outer Payload Length, Traffic Class, Flow Label,
            Hop Limit, and Next Header fields.
   R21.5. Submit the packet to the egress IPv6 FIB lookup for
            transmission to the next destination.
   S16. }
```

When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID with the REPLACE-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
             with Code 4 (SR Upper-layer Header Error)
             and Pointer set to the offset of the Upper-Layer header,
             interrupt packet processing, and discard the packet.
   S05. }
```

A.10. End.BM with REPLACE-CSID

When processing the SRH of a packet matching a FIB entry locally instantiated as an End.BM SID with the REPLACE-CSID flavor:


```
   S01. When an SRH is processed {
   S02.   If (Segments Left == 0 and (DA.Arg.Index == 0 or
              Segment List[0][DA.Arg.Index-1] == 0)) {
   S03.     Stop processing the SRH, and proceed to process the next
              header in the packet, whose type is identified by
```

```
              the Next Header field in the routing header.
   S04.   }
   S05.   If (IPv6 Hop Limit <= 1) {
   S06.     Send an ICMP Time Exceeded message to the Source Address,
              Code 0 (Hop limit exceeded in transit),
              interrupt packet processing and discard the packet.
   S07.   }
   S08.   max_LE = (Hdr Ext Len / 2) - 1
   R01.   If (DA.Arg.Index != 0) {
   R02.     If ((Last Entry > max_LE) or (Segments Left > Last Entry)) {
   R03.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R04.     }
   R05.     Decrement DA.Arg.Index by 1.
   R06.     If (Segment List[Segments Left][DA.Arg.Index] == 0) {
   R07.       Decrement Segments Left by 1.
   R08.       Decrement IPv6 Hop Limit by 1.
   R09.       Update IPv6 DA with Segment List[Segments Left]
   R10.1.     Push the MPLS label stack for B.
   R10.2.     Submit the packet to the MPLS engine for transmission.
   R11.     }
   R12.   } Else {
   R13.     If((Last Entry > max_LE) or (Segments Left > Last Entry+1)){
   R14.       Send an ICMP Parameter Problem to the Source Address,
                Code 0 (Erroneous header field encountered),
                Pointer set to the Segments Left field,
                interrupt packet processing and discard the packet.
   R15.     }
   R16.     Decrement Segments Left by 1.
   R17.     Set DA.Arg.Index to (128/LNFL - 1).
   R18.   }
   R19.   Decrement IPv6 Hop Limit by 1.
   R20.   Write Segment List[Segments Left][DA.Arg.Index] into the bits
            [LBL..LBL+LNFL-1] of the Destination Address of the IPv6
            header.
   R21.1. Push the MPLS label stack for B.
   R21.2. Submit the packet to the MPLS engine for transmission.
   S16. }
```


When processing the Upper-Layer header of a packet matching a FIB entry locally instantiated as an End.BM SID with the REPLACE-CSID flavor:

```
   S01. If (Upper-Layer header type is allowed by local configuration) {
   S02.   Proceed to process the Upper-Layer header
   S03. } Else {
   S04.   Send an ICMP Parameter Problem to the Source Address
             with Code 4 (SR Upper-layer Header Error)
             and Pointer set to the offset of the Upper-Layer header,
             interrupt packet processing, and discard the packet.
   S05. }
```

Contributors

Liu Aihua
ZTE Corporation
China

Email: liu.aihua@zte.com.cn

Dennis Cai
Alibaba
United States of America
Email: d.cai@alibaba-inc.com

Darren Dukes
Cisco Systems, Inc.
Canada
Email: ddukes@cisco.com

James N Guichard
Futurewei Technologies Ltd.
United States of America
Email: james.n.guichard@futurewei.com

Cheng Li
Huawei Technologies
China
Email: c.l@huawei.com

Robert Raszuk
NTT Network Innovations
United States of America

Email: robert@raszuk.net

Ketan Talaulikar
Cisco Systems, Inc.
India
Email: ketant.ietf@gmail.com

Daniel Voyer
Bell Canada
Canada
Email: daniel.voyer@bell.ca

Shay Zadok
Broadcom
Israel
Email: shay.zadok@broadcom.com

Authors' Addresses

Weiqiang Cheng (editor)
China Mobile
China
Email: chengweiqiang@chinamobile.com

Clarence Filsfils
Cisco Systems, Inc.

Belgium
Email: cf@cisco.com

Zhenbin Li
Huawei Technologies
China
Email: lizhenbin@huawei.com

Bruno Decraene
Orange
France
Email: bruno.decraene@orange.com

Francois Clad (editor)
Cisco Systems, Inc.
France
Email: fclad.ietf@gmail.com