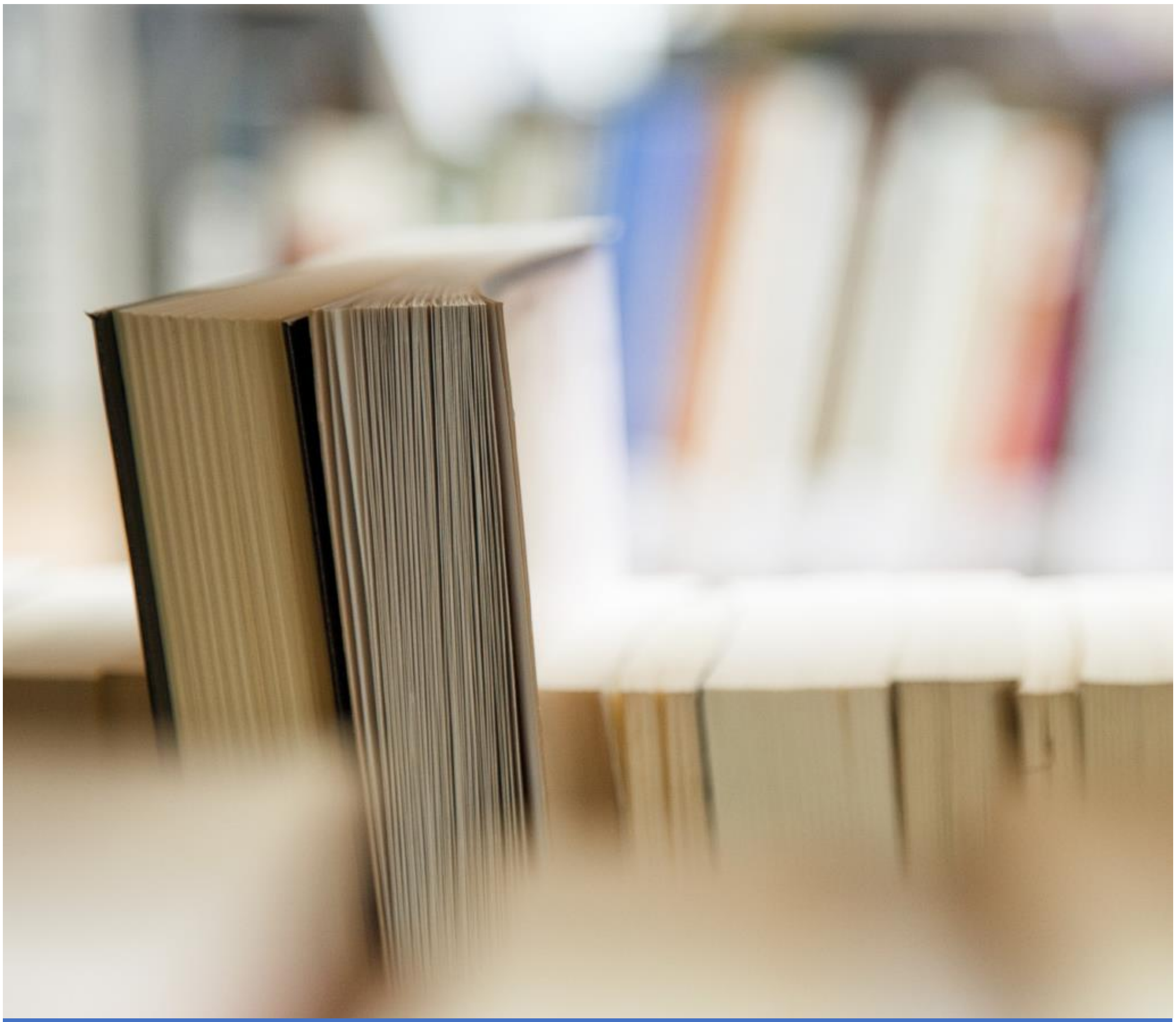


Implementing Segment Routing with SR-OS

Version 1.4

Colin Bookham 10/28/22 Nokia

Table of Contents

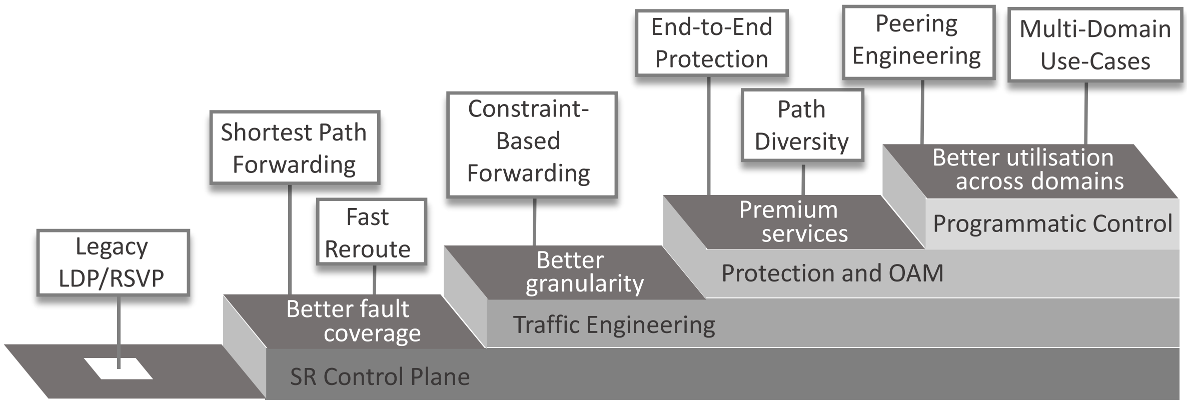
1 Introduction

The home of Segment Routing (SR) in the IETF is the Source Packet Routing in Networking (SPRING) Working Group. The primary objective for SR was to provide a mechanism to steer packets through a network without the need for per-path state information to be held at transit nodes. Additionally, this should be done using existing control, management, and data-plane protocols wherever possible. Since its conception SR has of course achieved this, and in the course of doing so has created a new realm of opportunities for network operators to increase their service offerings and improve network resilience.

SR can follow the shortest path ECMP-aware route through a network, or it can follow a specific set of instructions or segments providing the ability to source-route a packet. These instructions can be programmed using a centralised controller, or they can be programmed in a distributed manner with each SR-capable route autonomously computing the CSPF. When SR paths are programmed by a centralised controller additional path placement constraints over that of a typical CSPF can be applied, such as bidirectionality, disjointness, and the ability to reoptimise paths that fall outside of their intended objectives. SR offers a Fast Reroute capability using Loop-Free Alternates (LFA) to compute both link-protect and node-protect repair tunnels, again without any changes to existing control or data planes. It can be used with an MPLS data-plane (SR-MPLS) using an IPv4 or IPv6 control plane, or an IPv6 data-plane (SRv6). It can interwork with classic LDP in a fully integrated way offering an end-to-end heterogeneous service, or simply provide extended fast reroute coverage to LDP using SR-based repair tunnels. It is integrated with Flexible Algorithm from day one offering the ability to deliver traffic engineering without the need to impose a set of instructions or segments at the headend. It has an extensive OAM capability to assist in debugging and troubleshooting. SR is essentially a framework of powerful tools that continues to evolve.

When introducing SR, the benefits can be considered incremental, and can be added in an accumulative manner. Some very simple changes to a network with no additional control plane machinery can bring instant benefits. As confidence and knowledge grows additional SR capabilities can be introduced, further increasing network capabilities, which in turn can enhance product portfolios. This is shown in *Figure 1-1*, where simply enabling the advertisement of SR extensions into the IGP can deliver shortest path SR tunnels and increase fast reroute coverage for both SR and LDP tunnels. The introduction of traffic engineering with constraint-based forwarding brings further capability and can be implemented in a number of ways depending on an operators requirements. Traffic engineered paths can be source-routed and computed either locally by the headend router, or they can be computed by a centralised controller. Alternatively, they may not need to be source-routed and can be advertised and understood network-wide using Flexible-Algorithm. The introduction of end-to-end protection for traffic engineered paths further increases failure detection and reconvergence times, as well as offering the path diversity between primary and backup paths. Finally, SR can be introduced for use in single or multi-domain environments. The multi-domain case may be two domains of the same operator that need connectivity, such as a network running a seamless MPLS environment, or it may be an operator wishing to engineer how traffic leaves its domain when handing off to another operator. All of these capabilities have multiple options with regard to how they are implemented giving the operator the ability to pick the relevant tools from the toolset.

Figure 1-1: SR Introduction and Benefits



As the standardisation of SR has progressed, so has the feature support within SR-OS. That feature set has reached a point where it offers a rich and complete toolset for deployment of SR. The objective of this document is to explain the applications and extensions of SR, and to demonstrate how they are configured in SR-OS through illustration and example. The document assumes that the reader has a basic understanding of the concepts of SR but provides an overview of each application of SR to ensure that concepts and rationale are understood before demonstrating the SR-OS implementation of that application.

Document Structure

Although SR can be instantiated on either an MPLS or IPv6 data-plane, this document focuses largely on the former. This rationale for that is that SR-MPLS can be considered mature and fairly widely deployed, whereas SRv6 has seen significantly less take-up and has open that are still being addressed. While the document focuses on SR-MPLS, chapter 12 provides an overview of SRv6, describes the SR-OS implementation of SRv6 and details the current status of SRv6 in the IETF at the time of writing, notably around the use of compressed SID (C-SID).

The document predominantly uses IS-IS as an IGP and I make no apologies for that. A large part of the document is agnostic to the IGP. In chapters 3 and 4 I have covered shortest path SR for IS-IS and OSPF respectively because documenting the base SR extensions separately seems appropriate. However, there is a significant amount of content repeated and overlap between those chapters, so my recommendation would be to read just one of them. This is the only time the document separates OSPF and IS-IS in this manner. Elsewhere in the document I cover both protocols concurrently and identify how each supports a given feature.

The document deals generally with SR-OS and focuses on 7x50 products, notably, 7750, 7950, and 7250, and is based on SR-OS release 22.10.R1. As it is not an FP-based product some features may not be supported on the 7250 IXR, and I have attempted to call out which features are either not supported or are implemented differently owing to the underlying hardware. However, please be aware that the document is only as current as the time it was written, and feature parity evolves. To that end it is recommended that readers check the relevant Release Notes for an updated list of features not supported on the 7250 IXR.

I have tried to make each chapter readable as standalone so that a reader can go directly to chapters of specific interest, but to avoid repetition I frequently refer to topics that have been previously described in other chapters.

2 Preparing for SR

Before SR of any form can be enabled in SR-OS a block of MPLS labels needs to be assigned for use with SR-MPLS. In addition, to ensure that SR functions as expected network interfaces should be enabled for MPLS and RSVP. This section discusses those requirements.

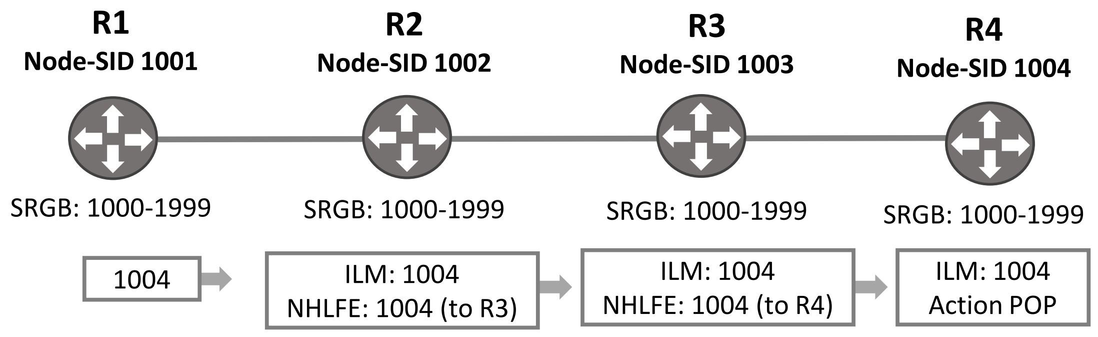
SR Label allocation

A range of MPLS labels needs to be set aside for SR operation at each SR router in the domain. If the entire label range is allocated to SR is assigned to a single SR domain that range can be considered the SR Global Block (SRGB). If the node spans multiple SR domains and allocates a subset of the range to each domain, then there is one SRGB for each domain. Later in this section I will illustrate how SR-OS allows for the assignment of label ranges to one or multiple SR domains. Prior to that, some consideration needs to be given the actual number of labels that should be assigned to SR, as well as the exact start and end numbers in the MPLS label block.

Indexing and Absolute SIDs

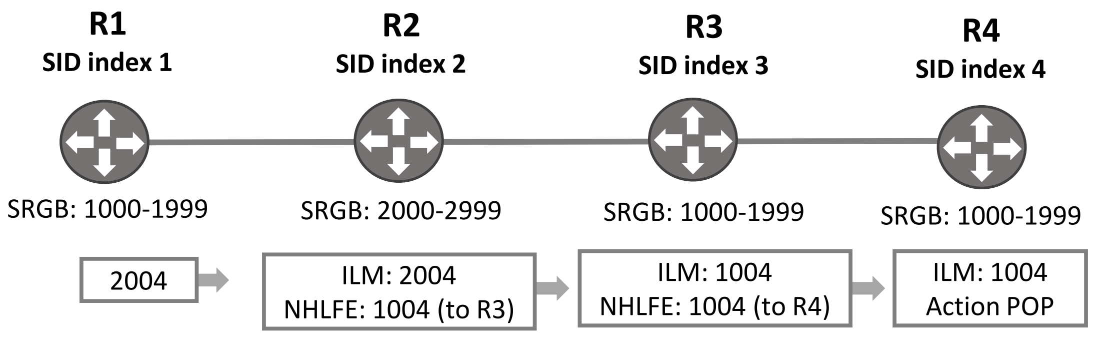
Not all routers may be capable of supporting the same SRGB. Allowed label ranges will vary across different vendor platforms, and also between different platforms of the same vendor because of the varying hardware capabilities. If a common SRGB can be found across all the devices in the SR domain then the values allocated to Node-SIDs can use what is termed as *absolute* values. This is shown in *Figure 2-1* where routers R1 through R4 have a common SRGB of 1000-1999. R4 advertises its Node-SID as 1004 and this value is learned by the remaining routers which each install an Incoming Label Map (ILM) entry for label 1004, with a corresponding next-hop Label Forwarding Entry (NHLFE) containing the next-hop forwarding information. When R1 wishes to forward traffic towards R4 it pushes on the label value of 1004. At R2 the ILM entry for label 1004 has an NHLFE of label 1004 with a next-hop of R3. At R3 the ILM entry for label 1004 has an NHLFE of 1004 with a next-hop of R4. This action at R3 assumes that Penultimate Hop Popping (PHP) is not signalled by R4, which is always the case in SR-OS. Finally, at R4 the ILM entry for label 1004 instructs the router to pop the packet.

Figure 2-1: SID Absolute Values



In SR-MPLS, the use of the same SRGB on all nodes within the SR domain is recommended. This is because the same label value represents the same globally unique segment at each SR node in the domain which eases operations and troubleshooting. If using a common SRGB is not possible, then *indexing* must be used. When indexing is used each router in the domain advertises its own SRGB start label and an offset label for the Node-SID called a *SID index*. Upstream routers then use the formula {SRGB start label + SID index} to obtain a local label for a downstream Node-SID. For example, in *Figure 2-2* routers R1, R3, and R4 have an SRGB of 1000-1999 while R2 has an SRGB of 2000-2999. R4 advertises its Node-SID as SID index 4 and an SRGB start label of 1000. When R1 wishes to forward traffic towards R4 the next-hop downstream router is R2, so R1 uses the sum {2000 + 4} representing R2's SRGB start label and R4's advertised SID index. R1 therefore imposes a label value of 2004. R2's downstream neighbor towards R4 is R3, so R2 uses the sum {1000 + 4} to derive an NHLFE label value of 1004 when forwarding traffic towards R3. Finally, R3 also calculates an NHLFE label value of 1004 when forwarding traffic towards R4, again assuming that PHP is not in use. At R4 the ILM entry for label 1004 instructs the router to pop the packet.

Figure 2-2: SID Indexing



When different SRGBs are in use, the forwarding behaviour is often compared to that of LDP in independent label distribution mode in as much as the label values can change on hops with different SRGB start labels. The notable difference is that LDP explicitly advertises the label that it expects to receive whereas with SR indexing the label value used to forward traffic to a downstream router is computed by the upstream router using the formula local label = start label + SID index.

How Many Labels do I need?

It's a good idea to have a coherent strategy for SID numbering from the outset, and to that end the first question operators should consider is just exactly how many labels do I need? When attempting to answer that question there are some fairly obvious factors such as the size of the network, but there are one or two not-so-obvious things that are also worth considering.

Every SR router in the SR domain requires a unique Node-SID. One of the technologies that SR can fully utilise to the maximum is Flexible-Algorithm [1]. Flexible-Algorithm, or *Flex-Algo*, provides a way for IGPs to compute constraint-based paths across a domain and can be a very powerful tool if a reasonably basic level of traffic-engineering is required. Flexible-

Algorithm is discussed in detail in chapter 10, but for now it's important to understand that each logical topology that Flex-Algo creates is given an identifier in the range 128-255. When Flex-Algo is used with SR-MPLS, each SR node that originates or terminates services needs to be assigned its own Node-SID for each of the Flex-Algo identifiers. Hence, if there are 100 SR nodes in the domain and there are three Flex-Algo topologies in use, the requirement would be 100 Node-SIDs for algorithm 0 (the base algorithm) and 300 Node-SIDs for the Flex-Algo algorithms.

In a similar vein, if a network runs dual-stack IPv4/IPv6 and there are aspirations to use SR-MPLS with an IPv6 control plane, then an IPv6 Node-SID will need to be allocated in addition to the IPv4 Node-SID.

Another consideration is how Adjacency-SIDs are allocated. By default, SR-OS allocates Adj-SIDs from the dynamic label range and they are non-persistent. That is, if a node restarts for some reason the allocated Adj-SIDs may well change. This might not be the desired behaviour if, for example, SR-TE LSPs are statically configured and/or signalled with Adj-SIDs as hops. SR-OS therefore also allows for Adj-SID labels to be statically assigned for individual adjacencies, and these labels are persistent. The label values for statically assigned Adj-SIDs are taken from a reserved SR Local Block (SRLB). If statically assigned Adj-SIDs are used, it offers the option of using either values that have local significance or globally unique values:

i. If global values are assigned to Adj-SIDs it has the potential to reduce the imposed label stack. Take the simple case of a TI-LFA backup tunnel to a Q-node that is adjacent to a P-node. By default, that backup tunnel would need two labels; a Node-SID label to the P-node followed by the Adj-SID to the Q-node. If globally unique Adj-SID values were in use however, the backup tunnel would only need a single label of the P-node to Q-node Adj-SID because that SID/label is globally unique. The use of globally unique Adj-SIDs in this manner is not something that is currently supported in SR-OS for SR-MPLS. Moreover, the fairly obvious disadvantage of using globally unique Adj-SID label values is that in large networks they could very quickly consume large chunks of the available MPLS label space.
ii. If locally significant values are used the advantage is that the same approach, methodology, and label space can be repeated on each and every node. The disadvantage is that additional labels will need to be imposed to target remote Adj-SIDs.

Start label

Once the anticipated number of labels required for SR has been derived then it is necessary to determine where in the MPLS label stack range the SR range should reside. In SR-OS SR labels are carved out of the dynamic range, which by default has a start label of 18432 and an end label that is platform dependent.

Output 2-1: Default SR-OS Label Ranges

```
*A:admin@7750# show router mpls-labels label-range

===============================================================================
Label Ranges
```

```
===============================================================================
Label Type      Start Label End Label   Aging       Available   Total
-------------------------------------------------------------------------------
Static          32          18431       -           18400       18400
Dynamic         18432       524287      0           505855      505856
    Seg-Route   0           0           -           0           0
```

To help decide where the SRGB should reside in that range it's worth making a couple of observations:

i. SR-OS allocates labels to MPLS clients (RSVP, LDP, BGP etc) and services from the high end of the platform range.
ii. There are very few networks that have a requirement for 18431 static labels.

With those points in mind, and if you are given freedom of choice and not forced to use a certain range to match other vendors ranges, the recommendation would be twofold. Firstly, start the SR block at the low end of the MPLS label range because it is much easier for the low end to be common (across Nokia and other vendor platforms), and secondly to recover some of the range allocated to static labels by default. In the following example I firstly reduce the size of the static label range to 32-11999. Although *Output 2-2* shows a configured static label range of 11968 this value allows for the additional 0-32 reserved labels, resulting in an end label of 11999. The example then configures the SR label range to be in the range 12000-*19999*.

Output 2-2: SR Label Allocation

```
    router "Base" {
        mpls-labels {
            static-label-range 11968
            sr-labels {
                start 12000
                end 19999
            }
        }
    }
```

The resulting label ranges are shown in *Output 2-3* with 8000 labels allocated to SR in the range 12000-19999. This range can be allocated to a single domain in global mode, or sub-divided into multiple domains in per-instance mode. This is illustrated in chapters 3 and 4 for IS-IS and OSPF respectively.

Output 2-3: Modified Label Ranges

```
A:admin@R1# show router mpls-labels label-range

===============================================================================
Label Ranges
===============================================================================
Label Type      Start Label End Label   Aging       Available   Total
-------------------------------------------------------------------------------
Static          32          11999       -           11968       11968
Dynamic         12000       524287      0           504287      512288
    Seg-Route   12000       19999       -           0           8000
```

MPLS and RSVP Interfaces

Any network interface required to support SR should be enabled under the **mpls** context. Equally, to ensure that traffic engineering parameters are correctly advertised interfaces running SR should also be enabled under the **rsvp** context. This requirement is largely historical as in the past it was the only place to enable traffic engineering for a given interface. However, SR, or more explicitly SR-TE, still requires this traffic engineering information as an ingress LSR computing a CSPF will not include non-TE links in the topology. To avoid encountering issues it is recommended to have every network interface defined in both MPLS and RSVP contexts, and to ensure that traffic engineering is enabled within all IGP instances.

Output 2-4: MPLS and RSVP Interfaces

```
        mpls {
            admin-state enable
            interface "link-to-R2" {
            }
            interface "link-to-R4" {
            }
        }
        rsvp {
            admin-state enable
            interface "link-to-R2" {
            }
            interface "link-to-R4" {
            }
        }
```

3 Shortest Path SR with IS-IS

Shortest Path SR refers to the use of a single Prefix-SID (Node-SID) in the SR-MPLS header representing the destination node. The Prefix-SID/Node-SID represents the shortest-path ECMP-aware path to the destination. Throughout this document, when a reference is made to IS-IS it refers explicitly to Multi-Topology ID 0 unless otherwise stated.

IS-IS Base SR Extensions

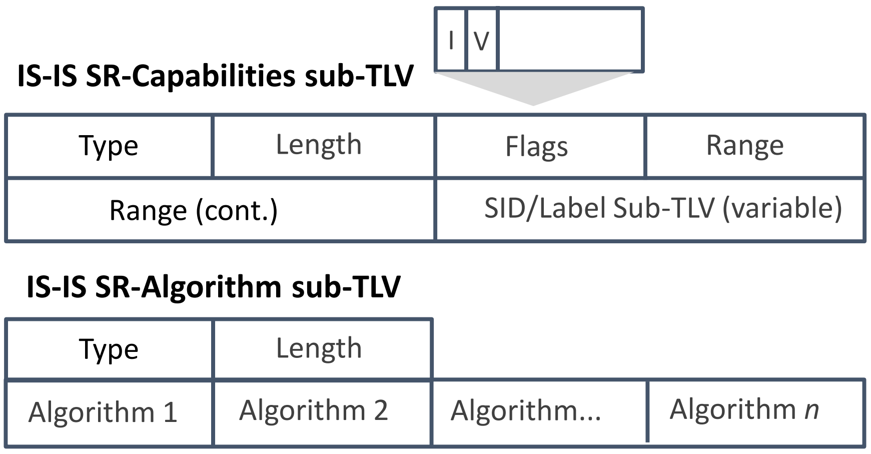
SR has no requirement for a dedicated MPLS control plane in the same way that RSVP or LDP does, and simply piggybacks SR segment information into the IGP for dissemination across the SR domain. This segment information includes Prefix-SIDs and Adjacency-SIDs that a given node owns as well as its SR data plane capabilities and label range used for SR. This section describes the main extensions to IS-IS for SR-MPLS and then illustrates how they are enabled in SR-OS.

Router Capabilities

IS-IS has extensions defined for advertising router capability information using the Capability TLV [3] with optional sub-TLVs. It is intended to indicate capabilities such as graceful restart, traffic engineering parameters, or IS-IS mesh group, and is extended to indicate a routers capability to support SR.

The Router Capability TLV defines a number of sub-TLVs for SR, including the SR-Capabilities sub-TLV and the SR-Algorithm sub-TLV. The SR-Capabilities sub-TLV contains a SID/label sub-TLV containing the first value of the SRGB, while the range contains the number of SID/Labels in that SRGB. The Flags field contains only two bits. The I-Flag when set indicates that the router is capable of processing SR-MPLS-encapsulated IPv4 packets, while the V-Flag when set indicates that the router is capable of processing SR-MPLS-encapsulated IPv6 packets. The SR-Algorithm sub-TLV is used to indicate the algorithms that a router supports when calculating reachability to other nodes or prefixes. In the base SR extensions only two algorithms are defined. Algorithm 0 uses the well-known shortest path algorithm based on link metric. Algorithm 1 uses a strict SPF algorithm based on link metric. It is identical to algorithm 0, but algorithm 1 requires that all nodes along the path adhere to the SPF routing decision and do not use any form of local policy to alter that decision. SR-OS does not currently support algorithm 1.

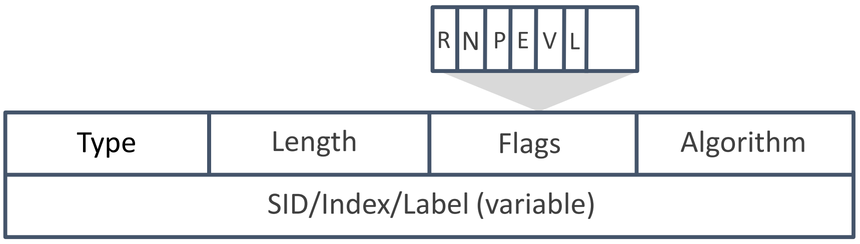
Figure 3-1: IS-IS Base SR Extensions to Capabilities TLV



Prefix-SID advertisement

The IS-IS Prefix-SID sub-TLV is used to advertise one or more globally unique Node-SIDs assigned to a given node. In SR-OS it is a sub-TLV of TLV-135 (Extended IPv4 Reachability) or TLV-236 (IPv6 IP Reachability). The format of the Prefix-SID sub-TLV is shown in *Figure 3-2*.

Figure 3-2: IS-IS Prefix-SID sub-TLV



The algorithm field contains the identifier of the algorithm the router uses to compute the reachability of the prefix to which the Prefix-SID is associated. As described above, the base algorithm described only two possible algorithms, but the introduction of Flexible-Algorithm [1] provides a significant increase in flexibility. Flexible-Algorithm is discussed in detail in chapter 10. There are a number of flags contained in the IS-IS Prefix-SID sub-TLV:

R-Flag	Readvertisement Flag. If set the prefix to which this Prefix-SID is attached has been propagated by the router from another IS-IS level.
N Flag	Node-SID Flag. If set, the Prefix-SID refers to the router identified by the prefix. This is typically a system or loopback address.
P-Flag	No-PHP (Penultimate Hop-Popping) flag. If set the penultimate hop must not pop the Prefix-SID before forwarding the packet to the advertising router. This is always set to one in SR-OS.
E-Flag	Explicit Null Flag. If set, any upstream neighbor of the Prefix-SID originator must replace the Prefix-SID with a Prefix-SID that has an Explicit Null value.

V-Flag	Value Flag. If set the Prefix-SID carries a value instead of an index. SR-OS always advertises Prefix-SIDs as an index.
L-Flag	Local flag. If set the value/index carried by the Prefix-SID has local significance.

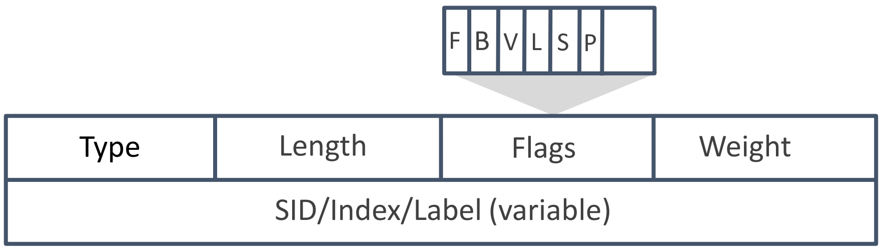
The Prefix-SID sub-TLV is always included when the associated Prefix Reachability TLV is propagated across level boundaries. This is important to allow shortest path SR-MPLS to work in networks containing multiple levels.

Adjacency-SID advertisement

Adjacency-SID TLVs are automatically advertised for all valid and formed adjacencies. Adjacency-SID labels are automatically assigned from the dynamic label range and are advertised as an explicit label value (not an index). The ILM is programmed with a pop operation for each advertised Adjacency-SID in the data-path; in effect it is a swap to implicit-null operation.

The IS-IS Adjacency-SID is carried in the Extended IS Reachability TLV and has the format shown in *Figure 3-3*.

Figure 3-3: IS-IS Adjacency-SID sub-TLV



The weight field is used for load-balancing in the presence of multiple parallel adjacencies. Weighted load-balancing across multiple parallel adjacencies is not currently supported by SR-OS (although other forms of weighted load-balancing are supported and discussed in later chapters). The flags field contains a number of flags:

F-Flag	Address-Family Flag. If unset the Adj-SID is used when forwarding IPv4-encapsulated traffic to the neighbour. If set, the Adj-SID is used when forwarding IPv6-encapsulated traffic to the neighbour.
B-Flag	Backup Flag. If set the Adj-SID is eligible for protection. By default, SR-OS sets the B-flag to zero. If LFA is enabled, the B-flag is set to one.
V-Flag	Value Flag. If set the Adj-SID carries an explicit value. SR-OS always sets the V-flag to one.
L-Flag	Local Flag. If set the value/index carried by the Adj-SID has local significance. SR-OS always sets the L-flag to one.
S-Flag	Set Flag. When set the S-Flag indicates that the Adj-SID refers to a set of adjacencies. Only set by SR-OS when Adjacency-Sets are explicitly configured.

P-Flag	Persistent Flag. When set the P-flag indicates that the Adj-SID is persistently allocated (it is consistent across a router restart). Dynamically allocated Adj-SIDs are non-persistent. Statically assigned Adj-SIDs are persistent.

SID/Label Binding TLV

In addition to Prefix-SIDs and Adjacency-SIDs, SR-OS also provides support for the SID/Label Binding TLV. There are multiple potential uses of the SID/Label Binding TLV, one of which is that it may be used to advertise prefix to SID/Label mappings. This is the function of an SR Mapping Server (SRMS), and the use of an SRMS and the SID/Label Binding TLV is discussed in chapter 5.

Maximum Segment Depth

In an SR domain the Maximum Segment Depth (MSD) refers to the number of segments/labels a router is capable of imposing on a given packet. This information is essential to a centralised controller computing SR paths to ensure that the SID stack depth computed does not exceed the number of SIDs the head end is capable of actually imposing.

IS-IS is extended to advertise a Node MSD Advertisement sub-TLV [6] that acts as a container to signal multiple MSD types as concatenated sub-TLVs, each consisting of MSD-Type and MSD-Value. The IS-IS Node MSD Advertisement is carried as a sub-TLV of the Router Capabilities TLV and defines the Base MPLS Imposition (BMI-MSD) as a sub-TLV of the Node MSD Advertisement. The BMI-MSD is used to signal the total number of MPLS labels that can be imposed, including all service, transport, and OAM/entropy labels. It represents the smallest MSD supported by the node on the set of interfaces configured for use by the IGP instance. The Node MSD containing the BMI-MSD is advertised as soon as segment-routing is enabled.

Entropy Label Support

The insertion of Entropy labels at an ingress LSR provides a mechanism to load-balance traffic flows at transit LSRs that only parse the MPLS header. It consists of the insertion of an Entropy Label (EL) that is not signalled, and whose only purpose in the label stack is to provide entropy to improve load-balancing. However, to ensure that the egress LSR can unambiguously distinguish between entropy labels and application labels, an Entropy Label Indicator (ELI) is inserted as the label immediately preceding an EL (further towards the top of the stack) that uses the special-purpose MPLS label value of 7. The EL is inserted after the relevant MPLS transport label(s) and before any service labels.

An ingress LSR cannot insert EL's for packets going into a given label switched path (LSP) unless an egress LSR has indicated that it is capable of processing ELs on that LSP. This is referred to as the Entropy Label Capability (ELC). In addition to the ELC, it may be useful for an ingress LSR to know how many labels each transit LSR is capable of parsing for the purpose of EL-based load-balancing. After all, there is little point in inserting the ELI/EL as the fifth and sixth label in the label stack if a given LSR is only capable of parsing four labels deep. IS-IS has extensions [8] that allows an SR router to indicate its ELC and its Entropy Readable Label Depth (ERLD).

Although the ELC is considered a nodal attribute it is advertised with a prefix. The rationale for this is to allow the ELC information to be propagated across multiple levels, or indeed multiple domains. Hence, when a prefix is propagated between IS-IS levels the ELC signalling is preserved. Similarly, when redistributing between protocol instances it is desirable (although not enforced) to preserve the ELC information. The ELC is advertised as an Extended IPv4/IPv6 Prefix Attribute [10]. The Prefix Attribute Flags field has an ELC flag (E-flag) used to signal that a node is capable of processing Entropy Labels. The ERLD is advertised as a sub-TLV of the Node MSD Advertisement TLV previously discussed, using an MSD type of ERLD-MSD.

Note: Signalling Entropy Label Capability simply means that the router is capable of processing Entropy Labels on received packets. It does not mean that the router will impose Entropy Labels as this action requires explicit configuration that is discussed in chapter 14.

Configuration

This section details the configuration requirements for enabling IS-IS shortest path SR. The prerequisite before proceeding with the configuration outlined in this section is that a label range has been allocated to SR, and the following example uses the SRGB from *Output 2-2* which for completeness is 12000-19999.

Output 3-1 contains the entire IS-IS configuration for completeness. A description of the pertinent configuration commands required to enable shortest path SR follows the output.

Output 3-1: IS-IS Basic Configuration for SR

```
        isis 0 {
            admin-state enable
            authentication-key <password>
            authentication-type message-digest
            advertise-router-capability as
            prefix-attributes-tlv true
            traffic-engineering true
            area-address [49.0001]
            segment-routing {
                admin-state enable
                entropy-label true
                prefix-sid-range {
                    global
                }
            }
            interface "link-to-R2" {
                hello-authentication-key <password>
                hello-authentication-type message-digest
                interface-type point-to-point
                level-capability 2
                level 2 {
                    metric 100
                }
            }
            interface "link-to-R4" {
                hello-authentication-key <password>
                hello-authentication-type message-digest
                interface-type point-to-point
                level-capability 2
                level 2 {
                    metric 100
                }
            }
            interface "system" {
                passive true
                level-capability 2
                ipv4-node-sid {
                    label 14001
                }
            }
            level 2 {
                wide-metrics-only true
            }
        }
```

The **advertise-router-capability** command instructs the router to advertise the Capabilities TLV which as discussed previously contains a number of sub-TLVs for SR, including the SR-Capabilities sub-TLV and the SR-Algorithm sub-TLV. The command has an optional argument of **area** or **as** and in this example is set to the latter. The IS-IS Router Capability TLV allows for area or domain-wide propagation, and this is controlled by an S-flag and D-flag. When the S-flag is set, the Router Capability TLV must be flooded across the entire routing domain. If the S-bit is not set, the TLV must not be leaked between levels. The D-flag is set if the Router Capability TLV is leaked from Level 2 to Level 1 to ensure that it is not re-advertised back into Level 2. However, the IS-IS extensions for SR [2] specify that the SR-Capabilities sub-TLV and SR-Algorithm sub-TLV must not be advertised across level boundaries. To enforce this, it specifies that the Router Capability TLV S-flag must remain unset. In short, the S-flag remains unset regardless of the area/as setting.

Note: When the scope is set to **area**, SR-OS originates a Router Capability TLV containing only area-wide capabilities. When the scope is set to **as**, SR-OS will originate a Router Capability TLV with area-wide capabilities and a second Router Capability TLV with domain-wide capabilities. In the current release however, SR-OS does not support any domain-wide capabilities, hence the S-bit is always unset regardless of whether **area** or **as** is configured. In effect, the **area** argument can be considered as having no effect in the current release but is available as an option in readiness for future support of domain-wide capabilities.

Recall that ELC signalling is carried as a prefix attribute, and the **prefix-attributes-tlv** command allows this TLV to be advertised. The ELC, BMI-MSD, and ERLD-MSD are all advertised as follows:

i. The ELC is automatically advertised when **segment-routing**, **segment-routing entropy-label**, and **prefix-attributes-tlv** are all enabled, noting that **segment-routing entropy-label** is set to **true** by default and would therefore not be shown in a configuration output but is shown here for clarity.

ii. The BMI-MSD and the ERLD-MSD are advertised when **segment-routing**, **segment-routing entropy-label**, and **advertise-router-capability** are all enabled.

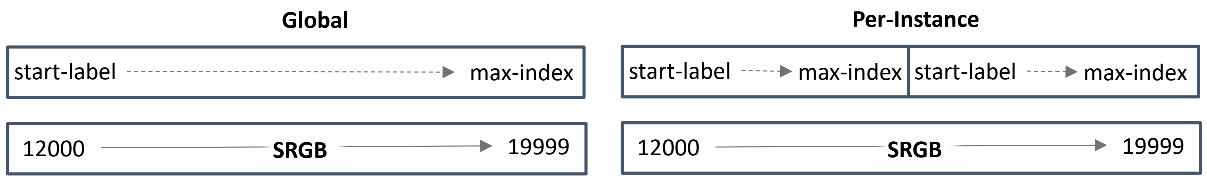
The **traffic-engineering** command enables legacy/conventional traffic engineering extensions in the IS-IS instance allowing for additional link attributes such as admin-group and TE metric to be flooded throughout the domain. In a network that runs purely shortest path SR, the **traffic-engineering** command is not strictly required because no CSPF is computed, but it is considered good practice and also allows for easier adoption of SR-TE should an operator need it. Moreover, many networks already have traffic engineering deployed so there is no reason to remove it.

Within the **segment-routing** context, the **prefix-sid-range** command allows for the configuration of the Prefix-SID *start-label* and *max-index* values, and two mutually exclusive modes of operation exist known as the *global* option and the *per-instance* option:

i. The **global** command takes the lowest value in the SRGB as the **start-label** value and the range of the SRGB as the **max-index**. When the **global** option is chosen any other instances of IS-IS configured on the same router are subsequently restricted to the same option.
ii. In the per-instance mode of operation the SRGB can be sub-divided into non-overlapping sub-ranges which can be individually allocated to different IS-IS instances. The per-instance mode is enabled by explicitly using the commands **start-label** and **max-index** and each sub-range may be independently used by different IGP instances provided each sub-range falls within the range of the SRGB.

In the example configuration of *Output 3-1* the **global** option is selected, but the concept of both options is shown in *Figure 3-4*.

Figure 3-4: Prefix-SID-Range Options



Under the system interface of the IS-IS instance a unique Node-SID is assigned to the primary IP address using the **ipv4-node-sid** command. The Node-SID can be expressed as an absolute value using the **label** argument followed by an explicit value, or it can be expressed as a label offset using the **index** argument, in which case the label is derived from the formula {start-label + SID index}. Regardless of whether the **label** or **index** option is configured, SR-OS always advertises the Node-SID as an index value. Node-SIDs can be assigned separately to IPv4 and IPv6 in a similar manner. An IPv6 Node-SID would only be needed if the intention was to use SR-MPLS with an IPv6 control plane. This topic is covered in detail in chapter 11 and is therefore not covered in any detail here.

Note: By default, SR-OS enforces the uniqueness of a Node-SID across multiple instances of IS-IS. That is, if a SID is assigned to an interface, that interface and SID cannot be assigned to multiple IS-IS instances. If there is a requirement to use the same Node-SID across multiple IS-IS instances a shared SID must be used, which is described in Chapter 14.

Finally, the segment-routing context is given an **admin-state** of **enable**. Once SR is enabled, the SR Capability sub-TLV and SR Algorithm sub-TLV are advertised. IPv4 Adjacency-SIDs sub-TLVs are advertised for every valid and formed adjacency, and if **ipv6-routing** is enabled under IS-IS, distinct IPv6 Adjacency-SID sub-TLVs are also advertised. Finally, Prefix-SID sub-TLVs are advertised for every configured IPv4 and/or IPv6 Node-SID. With the above configuration applied, *Output 3-2* shows the IS-IS database entry Link State PDU (LSP) flooded by SR-OS, truncated to show only the pertinent TLVs; notably the Capability TLV, the Extended IS Reachability TLV, and the Extended IP Reachability TLV. The contents of each of these top-level TLVs are analysed below the output using an on-the-wire packet capture of the advertised LSP.

Output 3-2: IS-IS Database Entry

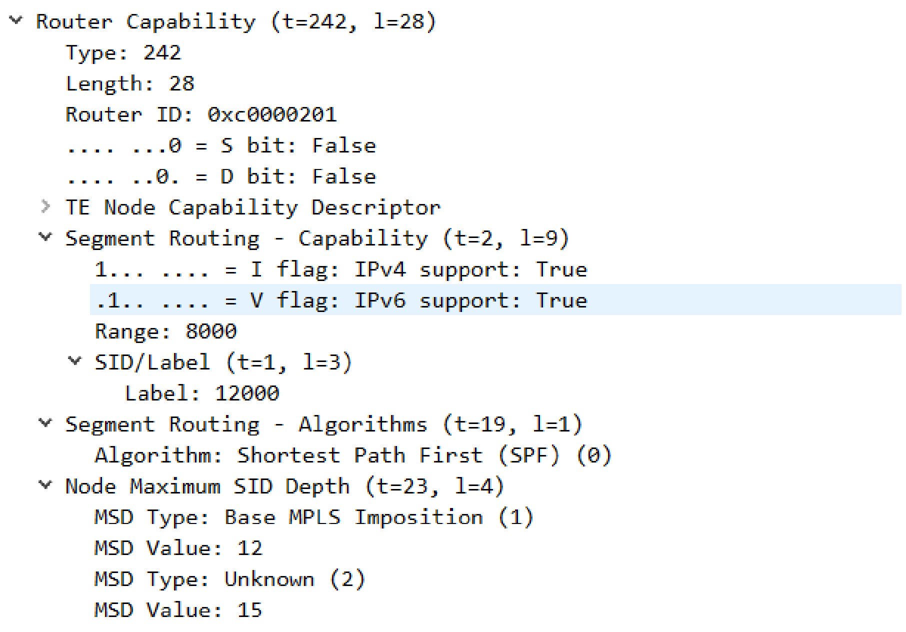
```
A:admin@R1# show router isis database R1.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================
... [snip]
TLVs :... [snip]
  Router Cap : 192.0.2.1, D:0, S:0
    TE Node Cap : B E M  P
    SR Cap: IPv4 MPLS-IPv6
       SRGB Base:12000, Range:8000
    SR Alg: metric based SPF
    Node MSD Cap: BMI : 12 ERLD : 15
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 76
    IF Addr   : 192.168.0.1
    Nbr IP    : 192.168.0.2
    MaxLink BW: 10000000 kbps
    Resvble BW: 10000000 kbps
    Unresvd BW:
        BW[0] : 10000000 kbps
        BW[1] : 10000000 kbps
        BW[2] : 10000000 kbps
        BW[3] : 10000000 kbps
        BW[4] : 10000000 kbps
        BW[5] : 10000000 kbps
        BW[6] : 10000000 kbps
        BW[7] : 10000000 kbps
    Admin Grp : 0x0
    TE Metric : 100
    Adj-SID: Flags:v4BVL Weight:0 Label:524286
  TE IS Nbrs   :
... [snip]
  TE IP Reach   :
    Default Metric  : 0
    Control Info:   S, prefLen 32
    Prefix   : 192.0.2.1
    Sub TLV   :
      AttrFlags: NE
      Prefix-SID Index:2001, Algo:0, Flags:NnP
```

Figure 3-5 shows the Router Capability TLV (TLV 242) from the advertised IS-IS LSP. The SR Capability sub-TLV has the I and V flags set to indicate support for both IPv4 and IPv6 SR-MPLS, and the range field indicates a label range of 8000. The SID/Label indicates a start-label of 12000, hence the label range is 12000-19999. The SR Algorithms sub-TLV indicates that only the base shortest path first algorithm is supported (algorithm 0).

The Capability TLV also contains the Node MSD Advertisement sub-TLV, which contains two sub-TLVs. The first is the BMI-MSD which shows a value of 12 labels. The second (which wireshark was unfortunately unable to decode) has an MSD-Type of 2 indicating that it is the ERLD-MSD, which has a value of 15 labels. The BMI of 12 labels and ERLD of 15 labels is standard for Nokia FP-based platforms. The BMI and ERLD for other platforms that use third-party silicon may vary, and indeed for the stated BMI packets may be circulated and processed more than once through the datapath to achieve higher numbers. Supported label stack depths for different platforms are discussed in detail in chapter 6.

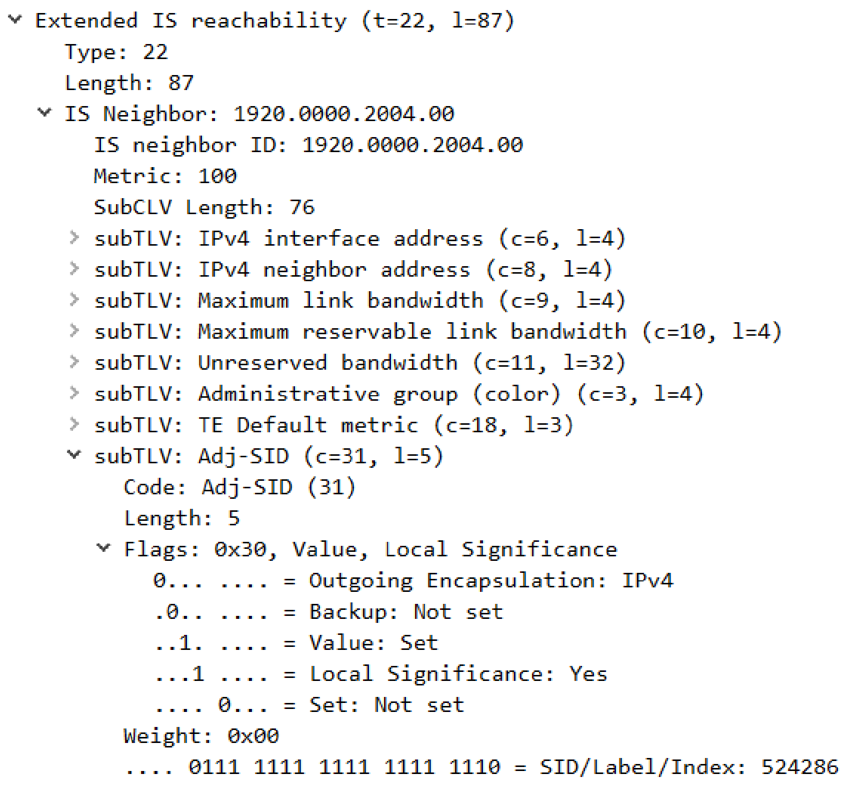
Figure 3-5: Router Capability TLV



```
Router Capability (t=242, l=28)
  Type: 242
  Length: 28
  Router ID: 0xc0000201
  .... ...0 = S bit: False
  .... ..0. = D bit: False
  TE Node Capability Descriptor
  Segment Routing - Capability (t=2, l=9)
    1... .... = I flag: IPv4 support: True
    .1.. .... = V flag: IPv6 support: True
    Range: 8000
    SID/Label (t=1, l=3)
      Label: 12000
  Segment Routing - Algorithms (t=19, l=1)
    Algorithm: Shortest Path First (SPF) (0)
  Node Maximum SID Depth (t=23, l=4)
    MSD Type: Base MPLS Imposition (1)
    MSD Value: 12
    MSD Type: Unknown (2)
    MSD Value: 15
```

The next TLV in the IS-IS LSP of interest is the Extended IS Reachability TLV shown in *Figure 3-6.* In this figure the conventional traffic engineering parameters such as admin-group, link bandwidth, and TE-metric are listed, followed by the Adjacency-SID sub-TLV. The F-flag (Family) is set to zero indicating that this interface is forwarding IPv4-encapsulated traffic. The B-Flag (Backup) is set to zero, so the Adj-SID is not eligible for protection. Note that if **loopfree-alternates** is enabled under IS-IS, the B-Flag is set to one. The V-flag (Value) is set to one to indicate that the advertised label is an explicit value (not an index). The L-flag (Local) is set to one indicating that the advertised label has local significance and is always set to one regardless of how the Adj-SID label is assigned (statically or dynamically). Finally, the S-flag (Set) is unset indicating that this adjacency is not part of a set. The weight field is set to zero and is currently not supported in SR-OS.

Figure 3-6: Extended IS Reachability TLV



```
⌄ Extended IS reachability (t=22, l=87)
    Type: 22
    Length: 87
  ⌄ IS Neighbor: 1920.0000.2004.00
      IS neighbor ID: 1920.0000.2004.00
      Metric: 100
      SubCLV Length: 76
    › subTLV: IPv4 interface address (c=6, l=4)
    › subTLV: IPv4 neighbor address (c=8, l=4)
    › subTLV: Maximum link bandwidth (c=9, l=4)
    › subTLV: Maximum reservable link bandwidth (c=10, l=4)
    › subTLV: Unreserved bandwidth (c=11, l=32)
    › subTLV: Administrative group (color) (c=3, l=4)
    › subTLV: TE Default metric (c=18, l=3)
    ⌄ subTLV: Adj-SID (c=31, l=5)
        Code: Adj-SID (31)
        Length: 5
      ⌄ Flags: 0x30, Value, Local Significance
          0... .... = Outgoing Encapsulation: IPv4
          .0.. .... = Backup: Not set
          ..1. .... = Value: Set
          ...1 .... = Local Significance: Yes
          .... 0... = Set: Not set
        Weight: 0x00
        .... 0111 1111 1111 1111 1110 = SID/Label/Index: 524286
```

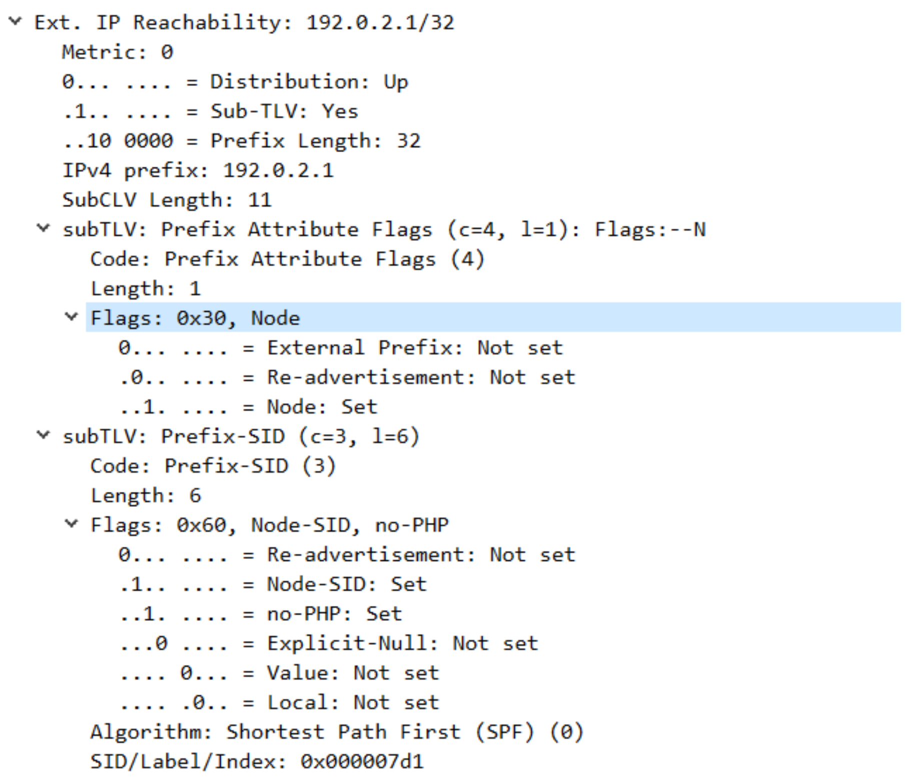
The final TLV containing SR information is the Extended IP Reachability TLV shown in *Figure 3-7*. The prefix is 192.0.2.1/32, and it contains both a prefix attribute sub-TLV and a prefix-SID sub-TLV. The prefix attribute sub-TLV has a flags field that in the output shows only the Node flag as set. Unfortunately, the packet capture has not decoded the ELC flag but does acknowledge that it is set because the value of the flags field is hex 0x30. This represents decimal 48, meaning that bits 2 (Node) and 3 (ELC) are set.

In the prefix-SID sub-TLV flags field the N-flag (Node-SID) and the P-flag (no-PHP) are set to one to indicate that this is a Node-SID, and that penultimate hop label popping should not be carried out. SR-OS always sets the P-flag to one for Node-SIDs, and currently this is non-configurable. The V-flag (Value) is set to zero indicating that the advertised label is an index, while the L-flag (Local) is set to zero to indicate that the label is not locally significant (it is globally unique). The R-flag (Readvertisement) is only set if the prefix is redistributed between levels and is therefore unset in this example as it is a pure Level-2 network. A Prefix-SID sub-TLV contained in an IP Reachability TLV remains intact during redistribution. This redistribution is default behaviour when redistributing from L1 to L2 but will only occur in the L2 to L1 direction if instructed by policy. In both instances the R-Flag is set.

Note: SR-OS provides support for redistributing Prefix-SID information between IS-IS instances. However, SR-OS will not propagate the Prefix-SID sub-TLV if the prefix is redistributed from a different protocol.

The algorithm associated with the prefix is the base shortest path first algorithm (algorithm 0) which is the default behaviour. Finally, the advertised label is 0x000007d1. This is an index value of 2001, which combined with the start-label of 12000 yields an absolute value of 14001.

Figure 3-7: Extended IP Reachability TLV



```
⌄ Ext. IP Reachability: 192.0.2.1/32
    Metric: 0
    0... .... = Distribution: Up
    .1.. .... = Sub-TLV: Yes
    ..10 0000 = Prefix Length: 32
    IPv4 prefix: 192.0.2.1
    SubCLV Length: 11
  ⌄ subTLV: Prefix Attribute Flags (c=4, l=1): Flags:--N
      Code: Prefix Attribute Flags (4)
      Length: 1
    ⌄ Flags: 0x30, Node
        0... .... = External Prefix: Not set
        .0.. .... = Re-advertisement: Not set
        ..1. .... = Node: Set
  ⌄ subTLV: Prefix-SID (c=3, l=6)
      Code: Prefix-SID (3)
      Length: 6
    ⌄ Flags: 0x60, Node-SID, no-PHP
        0... .... = Re-advertisement: Not set
        .1.. .... = Node-SID: Set
        ..1. .... = no-PHP: Set
        ...0 .... = Explicit-Null: Not set
        .... 0... = Value: Not set
        .... .0.. = Local: Not set
      Algorithm: Shortest Path First (SPF) (0)
      SID/Label/Index: 0x000007d1
```

A number of operational commands are available in SR-OS to validate the integrity of locally assigned and learned SIDs. The Adj-SID assigned to a given interface can be verified using the command shown in *Output 3-3*. In this example the router has multiple interfaces, so I have used pipe/match commands to reduce the size of the output. As can be observed, the adjacency to hostname R2 formed over the interface “link-to-R2” has an Adj-SID of 524285 while the adjacency to hostname R4 formed over the interface “link-to-R4” has an Adj-SID of 524286.

Output 3-3: IS-IS Adjacency with Adj-SID

```
*A:R1# show router isis adjacency detail | match expression "Hostname|Interface|IPv4 Adj
SID"
Hostname      : R2
Interface     : link-to-R2                         Up Time      : 0d 07:15:50
IPv4 Adj SID        : Label 524285
Hostname      : R4
Interface     : link-to-R4                         Up Time      : 0d 07:16:41
IPv4 Adj SID        : Label 524286
```

Prefix-SIDs learned locally and from other SR routers in the domain can be displayed using the command illustrated in *Output 3-4*. The topology of the SR domain is largely irrelevant at this point, but as can be observed there are six routers in the domain, R1 through R6, with system addresses of 192.0.2.1/32 through 192.0.2.6/32 respectively. The SID values are displayed as an index, and since all routers have the same SRGB (12000-19999) the Node-SID labels can be derived as 14001 through 14006 respectively. As previously described, the

N flag indicates a Node-SID, and the nP flag indicates that PHP should not be carried out by the penultimate hop router for this SID.

Output 3-4: IS-IS Prefix-SIDs

```
A:admin@R1# show router isis prefix-sids

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                            SID        Lvl/Typ     SRMS   AdvRtr
                                                          MT     Flags
-------------------------------------------------------------------------------
192.0.2.1/32                      2001       2/Int.       N     R1
                                                             0      NnP
192.0.2.2/32                      2002       2/Int.       N     R2
                                                             0      NnP
192.0.2.3/32                      2003       2/Int.       N     R3
                                                             0      NnP
192.0.2.4/32                      2004       2/Int.       N     R4
                                                             0      NnP
192.0.2.5/32                      2005       2/Int.       N     R5
                                                             0      NnP
192.0.2.6/32                      2006       2/Int.       N     R6
                                                             0      NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 6 (6 unique)
```

Another useful command for validating the status of SR tunnels is shown in *Output 3-5*. Unfortunately, as this is a *tools* command it is not strictly maintained or documented and does not provide any specific filtering arguments to search for a particular tunnel other than in-label (pipe and match is still possible of course). Regardless, the output provides a good overview of the router's behaviour for each tunnel including the label stack and outgoing interface (or interfaces if **loopfree-alternates** is enabled and a backup path has been computed). For a local Node-SID the Incoming Label Map (ILM) table is programmed with a pop operation and no out-label is shown. For local Adj-SIDs the ILM is also programmed with a pop operation but it is treated as a label-switch to implicit-null, which is why the Adj-SIDs in the output have an out-label of 3 (implicit-null).

Output 3-5: SR Tunnels Dump Output

```
A:admin@R1# tools dump router segment-routing tunnel
========================================================================================
Legend: (B) - Backup Next-hop for Fast Re-Route
        (D) - Duplicate
label stack is ordered from top-most to bottom-most
========================================================================================
---------------------------------------------------------------------------------------+
 Prefix                                                                                |
 Sid-Type        Fwd-Type       In-Label  Prot-Inst(algoId)                            |
                 Next Hop(s)                                   Out-Label(s) Interface  |
---------------------------------------------------------------------------------------+
 192.0.2.1
 Node            Terminating    14001     ISIS-0
 192.0.2.2
 Node            Orig/Transit   14002     ISIS-0
                 192.168.0.2                                   14002       link-to-R2
 192.0.2.3
 Node            Orig/Transit   14003     ISIS-0
                 192.168.0.2                                   14003       link-to-R2
 192.0.2.4
 Node            Orig/Transit   14004     ISIS-0
                 192.168.0.10                                 14004      link-to-R4
 192.0.2.5
 Node            Orig/Transit   14005     ISIS-0
                 192.168.0.2                                  14005      link-to-R2
                 192.168.0.10                                 14005      link-to-R4
 192.0.2.6
 Node            Orig/Transit   14006     ISIS-0
                 192.168.0.2                                  14006      link-to-R2
                 192.168.0.10                                 14006      link-to-R4
 192.168.0.2
 Adjacency       Transit        524285    ISIS-0
                 192.168.0.2                                  3          link-to-R2
 192.168.0.10
 Adjacency       Transit        524286    ISIS-0
                 192.168.0.10                                 3          link-to-R4
-------------------------------------------------------------------------------+
No. of Entries: 8
-------------------------------------------------------------------------------+
```

When a Node-SID is learned through IS-IS and the associated prefix is resolved as reachable it is programmed in the tunnel table. In *Output 3-6* there are single entries for Node-SIDs 192.0.2.2, 192.0.2.3, and 192.0.2.4, and two entries for Node-SIDs 192.0.2.5 and 192.0.2.6 with different next-hops as there are Equal Cost Multipath (ECMP) routes to these Node-SIDs.

There are also entries for 192.168.0.2/32 and 192.168.0.10/32 which are the next-hops for each of the router's adjacent neighbours, and these require some explanation. When SR-TE is deployed an ingress LER may push on a label stack where each label represents a link or a node (or possibly some other form of segment). SR-TE LSPs consisting of multiple labels (segments) are internally modelled by SR-OS as hierarchical constructs. The SR-TE LSP may have a number of labels imposed, but the ingress LER only needs to track the reachability of the first strict or loose hop as this hop is all that needs to be resolved for the LSP to be considered valid. This first hop represents the Next-Hop Label Forwarding Entry (NHLFE) of the SR shortest path tunnel, and this is maintained in the tunnel-table as a Node-SID or Adj-SID (the remainder of the label stack if not meaningful to the forwarding decision made by the ingress LER). This hierarchical construct is referred to as a *super NHLFE* and is tunnelled over the NHLFE of the first hop. SR-OS keeps an entry for the SR shortest path tunnel to a downstream Node-SID or Adj-SID in the tunnel table so that its NHLFE is readily available for use by super NHLFEs, hence the two entries exist for R1's adjacent neighbours (192.168.0.2/32 and 192.168.0.10/32) as well as all learned Node-SIDs. The purpose of this super NHLFE concept is to optimise NHLFE resource utilisation, particularly when many SR-TE LSPs have the same first next-hop. Each SR-TE LSP becomes its own super NHLFE but they all use a common NHLFE (or set of NHLFEs if ECMP is present). This concept of super NHLFEs will be discussed and illustrated numerous times throughout this document.

The last point of note is the preference of 11 assigned to SR-ISIS. This is higher than RSVP (7), SR-TE (8), and LDP (9), but can be changed in the **configure router isis segment-routing** context using the **tunnel-table-pref** command. This is discussed further in chapter 8 where the use of SR for BGP services is covered, together with some potential migration approaches.

Output 3-6: SR-ISIS Tunnel-Table

```
A:admin@R1# show router tunnel-table

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.2/32          isis (0)  MPLS  524299    11     192.168.0.2    100
192.0.2.3/32          isis (0)  MPLS  524300    11     192.168.0.2    200
192.0.2.4/32          isis (0)  MPLS  524301    11     192.168.0.10   100
192.0.2.5/32          isis (0)  MPLS  524302    11     192.168.0.2    200
192.0.2.5/32          isis (0)  MPLS  524302    11     192.168.0.10   200
192.0.2.6/32          isis (0)  MPLS  524303    11     192.168.0.2    300
192.0.2.6/32          isis (0)  MPLS  524303    11     192.168.0.10   300
192.168.0.2/32        isis (0)  MPLS  524305    11     192.168.0.2    0
192.168.0.10/32       isis (0)  MPLS  524298    11     192.168.0.10   0
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

4 Shortest Path SR with OSPF

Shortest Path SR refers to the use of a single Prefix-SID (Node-SID) in the SR-MPLS header representing the destination node. The Prefix-SID/Node-SID represents the shortest-path ECMP-aware path to the destination. Throughout this chapter when a reference is made to OSPF it refers explicitly to OSPFv2.

OSPF Base SR Extensions

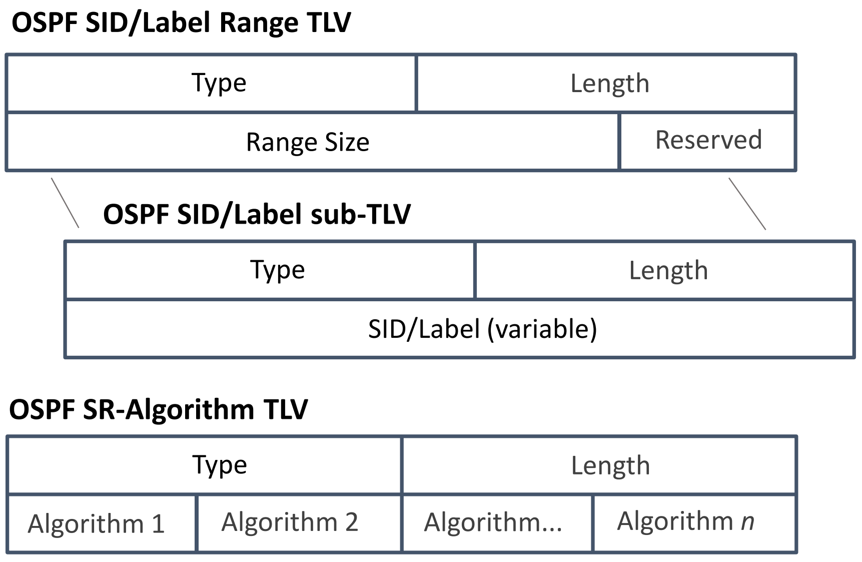
SR has no requirement for a dedicated MPLS control plane in the same way that RSVP or LDP does, and simply piggybacks SR segment information into the IGP for dissemination across the SR domain. This segment information includes Prefix-SIDs and Adjacency-SIDs that a given node owns as well as its SR data plane capabilities and label range used for SR. This section describes the main extensions to OSPF for SR-MPLS and then illustrates how they are enabled in SR-OS.

Router Capabilities

OSPF has extensions defined for advertising router capability information using the Router Information (RI) opaque LSA [4]. It is intended to indicate capabilities such as graceful restart, traffic engineering parameters, and OSPF stub. It is extended to indicate a routers capability to support SR.

The RI Opaque LSA in OSPFv2 and OSPFv3 adds support for the OSPF SID/Label Range TLV to advertise an SR router’s supported SID/Label space. It is a top-level TLV of the RI Opaque LSA and supports a single sub-TLV known as the SID/Label sub-TLV which is used to advertise the first SID/Label in the supported range. Hence, the combination of the SID/Label Range TLV and the SID/Label sub-TLV contain the start label and label range. The SR-Algorithm TLV is again a top level TLV of the RI Opaque LSA and is used to indicate the algorithms that a given SR router supports when calculating reachability to other nodes or prefixes. In the base SR extensions only two algorithms are defined. Algorithm 0 uses the well-known shortest path algorithm based on link metric. Algorithm 1 uses a strict SPF algorithm based on link metric. It is identical to algorithm 0, but algorithm 1 requires that all nodes along the path adhere to the SPF routing decision and do not use any form of local policy to alter that decision. SR-OS does not currently support algorithm 1.

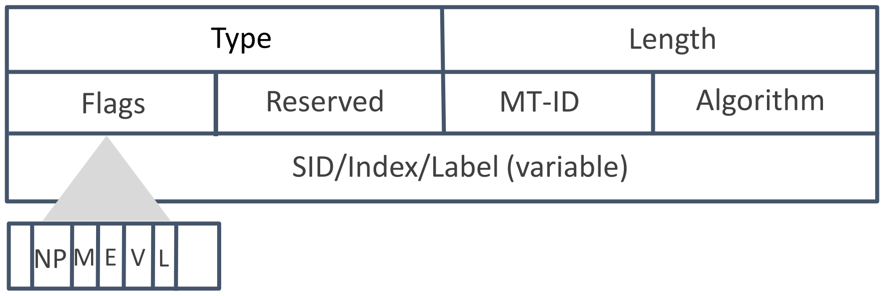
Figure 4-1: OSPF Base SR Extensions to RI Opaque LSA



Prefix-SID advertisement

The Prefix-Sid sub-TLV is a sub-TLV of the OSPF Extended Prefix TLV and can also be a sub-TLV of the OSPF Extended Prefix Range TLV when originated by a Mapping Server.

Figure 4-2: OSPF Prefix-SID sub-TLV



The algorithm field contains the identifier of the algorithm the router uses to compute the reachability of the prefix to which the Prefix-SID is associated. As described above, the base algorithm described only two possible algorithms, but the introduction of Flexible-Algorithm [1] provides a significant increase in flexibility and is discussed in detail in chapter 10. The MT-ID field allows for a Multi-Topology ID to be advertised with the prefix. A number of flags are defined in the flags field:

NP-Flag	No-PHP (Penultimate Hop-Popping) flag. If set the penultimate hop must not pop the Prefix-SID before forwarding the packet to the advertising router. This is always set to one in SR-OS.
M-flag	Mapping Server flag. If set the Prefix-SID was originated by an SR Mapping Server (SRMS).
E-Flag	Explicit Null Flag. If set, any upstream neighbor of the Prefix-SID originator must replace the Prefix-SID with a Prefix-SID that has an Explicit Null value.

V-Flag	Value Flag. If set the Prefix-SID carries a value instead of an index. SR-OS always advertises Prefix-SIDs as an index.
L-Flag	Local flag. If set the value/index carried by the Prefix-SID has local significance.

It's worth noting that (unlike IS-IS) the OSPF Prefix SID sub-TLV flags field has no flag to indicate that this is a Node-SID. This is because the OSPF Extended Prefix TLV (which carries the Prefix-SID sub-TLV) already contains a flags field containing a Node Flag (N-flag). The N-flag is set when the prefix identifies the advertising router with a globally reachable address.

The Prefix-SID sub-TLV is always included when the associated Extended Prefix TLV is propagated across areas. The Extended Prefix Range TLV is not propagated across areas and therefore neither is a Prefix-Sid sub-TLV attached to it.

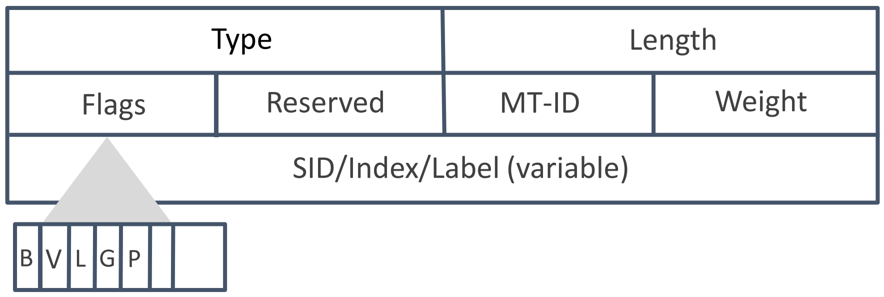
Note: SR-OS does not currently support the redistribution of Prefix-SID information between OSPF instances.

Adjacency-SID advertisement

Adjacency-SID TLVs are automatically advertised for all valid and formed adjacencies. Adjacency-SID labels are automatically assigned from the dynamic label range and are advertised as an explicit label value (not an index). The ILM is programmed with a pop operation for each advertised Adjacency-SID in the data-path; in effect it is a swap to an implicit-null operation.

The OSPF Adj-SID is a sub-TLV of the Extended Link TLV and has the format shown in *Figure 4-3*.

Figure 4-3: OSPF Adjacency-SID sub-TLV



Like the Prefix-SID sub-TLV, the Adjacency-SID sub-TLV provides the ability to advertise a Multi-Topology ID. The weight field is used for load-balancing in the presence of multiple parallel adjacencies. Weighted load-balancing across multiple parallel adjacencies is not currently supported by SR-OS (although other forms of weighted load-balancing are supported and discussed in later chapters). The flags field contains a number of flags:

B-Flag	Backup Flag. If set the Adj-SID is eligible for protection. By default, SR-OS sets the B-flag to zero. If LFA is enabled, the B-flag is set to one.

V-Flag	Value Flag. If set the Adj-SID carries an explicit value. SR-OS always sets the V-flag to one.
L-Flag	Local Flag. If set the value/index carried by the Adj-SID has local significance. SR-OS always sets the L-flag to one.
G-Flag	Group-Flag. When set the G-Flag indicates that the Adj-SID refers to a group of adjacencies.
P-Flag	Persistent Flag. When set the P-flag indicates that the Adj-SID is persistently allocated (i.e. is consistent across a router restart). Dynamically allocated Adj-SIDs are non-persistent. Statically assigned Adj-SIDs are persistent.

Maximum Segment Depth

In an SR domain the Maximum Segment Depth (MSD) refers to the number of segments/labels a router is capable of imposing on a given packet. This information is essential to a centralised controller computing SR paths to ensure that the SID stack depth of a computed does not exceed the number of SIDs the head end is capable of actually imposing.

OSPF is extended to advertise a Node MSD Advertisement TLV [7] that acts as a container to signal multiple MSD types as concatenated sub-TLVs, each consisting of MSD-Type and MSD-Value. The OSPF Node Advertisement is carried within the RI Opaque LSA and defines the Base MPLS Imposition (BMI-MSD) as a sub-TLV of the Node MSD Advertisement. The BMI-MSD is used to signal the total number of MPLS labels that can be imposed, including all service, transport, and OAM/entropy labels. It represents the smallest MSD supported by the node on the set of interfaces configured for use by the IGP instance. The Node MSD containing the BMI-MSD is advertised as soon as segment-routing is enabled.

Entropy Label Support

The insertion of Entropy labels at an ingress LSR provides a mechanism to load-balance traffic flows at transit LSRs that only parse the MPLS header. It consists of the insertion of an Entropy Label (EL) that is not signalled, and whose only purpose in the label stack is to provide entropy to improve load-balancing. However, to ensure that the egress LSR can unambiguously distinguish between entropy labels and application labels, an Entropy Label Indicator (ELI) is inserted as the label immediately preceding an EL (further towards the top of the stack) that uses the special-purpose MPLS label value of 7. The EL is inserted after the relevant MPLS transport label(s) and before any service labels.

An ingress LSR cannot insert EL's for packets going into a given label switched path (LSP) unless an egress LSR has indicated that it is capable of processing ELs on that LSP. This is referred to as the Entropy Label Capability (ELC). In addition to the ELC, it may be useful for an ingress LSR to know how many labels each transit LSR is capable of parsing for the purpose of EL-based load-balancing. After all, there is little point in inserting the ELI/EL as the fifth and sixth label in the label stack if a given LSR is only capable of parsing four labels deep. OSPF has extensions [9] that allows an SR router to indicate its ELC and its ERLD.

Although the ELC is considered a nodal attribute it is advertised with a prefix. The rationale for this is to allow the ELC information to be propagated across multiple areas, or indeed multiple domains. Hence, when a prefix is propagated between areas by an OSPF Area Border Router (ABR), the ELC signalling is preserved. Similarly, when redistributing between protocol instances it is desirable (although not enforced) to preserve the ELC information. The ELC is advertised as an Extended Prefix attribute (OSPFv2) or the Prefix Options field (OSPFv3). Both contain a flags field that has an ELC flag (E-Flag) to indicate the ELC setting. The ERLD is advertised as a sub-TLV of the Node MSD Advertisement TLV previously discussed, using an MSD type of ERLD-MSD.

Note: Signalling Entropy Label Capability simply means that the router is capable of processing Entropy Labels on received packets. It does not mean that the router will impose Entropy Labels on any egress packets as this action requires explicit configuration that is covered in chapter 14.

Configuration

This section details the configuration requirements for enabling OSPF shortest path SR. The prerequisite before proceeding with the configuration outlined in this section is that a label range has been allocated to SR, and the following example uses the SRGB from *Output 2-2* which for completeness is 12000-19999.

Output 4-1 contains the entire OSPF configuration for completeness. A description of the pertinent configuration commands required to enable shortest path SR follows the output.

Output 4-1: OSPF Basic Configuration for SR

```
        ospf 0 {
            admin-state enable
            advertise-router-capability area
            traffic-engineering true
            segment-routing {
                admin-state enable
                entropy-label true
                prefix-sid-range {
                    global
                }
            }
            area 0.0.0.0 {
                interface "link-to-R2" {
                    interface-type point-to-point
                    metric 100
                    authentication-type message-digest
                    message-digest-key 1 {
                        md5 <password> hash2
                    }
                }
                interface "link-to-R4" {
                    interface-type point-to-point
                    metric 100
                    authentication-type message-digest
                    message-digest-key 1 {
                        md5 <password> hash2
                    }
                }
                interface "system" {
                    node-sid {
                        label 14001
                    }
                }
            }
        }
```

The **advertise-router-capability** command instructs the router to advertise the RI Opaque LSA which as discussed previously contains a number of TLVs/sub-TLVs for SR, including the SR-Algorithm TLV, the SID/Label Range TLV and the SID/Label sub-TLV. The command has an optional argument of **area** or **as** and in OSPF it must be set to area. The RI LSA may be link-scoped, area-scoped, or AS-scoped. However, SR capabilities extensions for OSPF [3], notably the SR Algorithm TLV and SID/Label Range TLV, are specified as having only area-scope flooding.

The **traffic-engineering** command enables legacy/conventional traffic engineering extensions in the OSPF instance allowing for additional link attributes such as admin-group and TE metric to be flooded throughout the area. In a network that runs purely shortest path SR, the **traffic-engineering** command is not strictly required because no CSPF is computed, but it is considered good practice and also allows for easier adoption of SR-TE should an operator need it. Moreover, many networks already have traffic engineering deployed so there is no reason to remove it.

Recall that ELC signalling is carried as a prefix attribute, and the **prefix-attributes-tlv** command allows this TLV to be advertised. The ELC, BMI-MSD, and ERLD-MSD are all advertised as follows:

i. The router BMI-MSD is advertised when **segment-routing advertise-router-capability** is enabled.
ii. The ELC is automatically advertised when **segment-routing**, **segment-routing entropy-label** are enabled, noting that **segment-routing entropy-label** is set to **true** by default and would therefore not be shown in a configuration output but is shown here for clarity.
iii. The BMI-MSD and the ERLD-MSD are advertised when **segment-routing**, **segment-routing entropy-label**, and **advertise-router-capability** are all enabled.

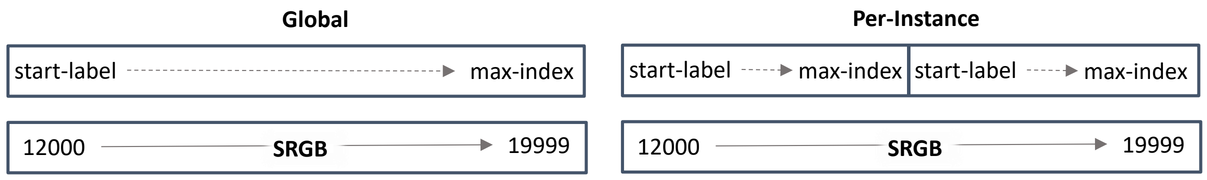
Within the **segment-routing** context, the **prefix-sid-range** command allows for the configuration of the Prefix-SID *start-label* and *max-index* values, and two mutually exclusive modes of operation exist known as the *global* option and the *per-instance* option:

i. The **global** command takes the lowest value in the SRGB as the **start-label** value and the range of the SRGB as the **max-index**. When the **global** option is chosen any other instances of OSPF configured on the same router are subsequently restricted to the same option.
ii. In the per-instance mode of operation the SRGB can be divided into non-overlapping sub-ranges which can be individually allocated to different OSPF instances. The per-instance mode is enabled by explicitly using the commands **start-label** and **max-**

index and each sub-range may be independently used by different IGP instances provided each sub-range falls within the range of the SRGB.

In the example configuration of *Output 4-1* the **global** option is selected, but the concept of both options is shown in *Figure 4-4*.

Figure 4-4: Prefix-SID Range Options



Under the system interface of the OSPF instance a unique Node-SID is assigned to the primary IP address using the **node-sid** command. The Node-SID can be expressed as an absolute value using the **label** argument followed by an explicit value, or it can be expressed as a label offset using the **index** argument, in which case the label is derived from the formula {start-label + SID index}. Regardless of whether the **label** or **index** option is configured, SR-OS always advertises the Node-SID as an index value.

Note: By default, SR-OS enforces the uniqueness of a Node-SID across multiple instances of OSPF. That is, if a SID is assigned to an interface, that interface and SID cannot be assigned to multiple OSPF instances. If there is a requirement to use the same Node-SID across multiple OSPF instances a shared SID must be used, which is described in Chapter 14.

Finally, the **segment-routing** context is put into an **admin-state** of **enable**. Once SR is enabled, the router informational SR capabilities are advertised including the SR-Algorithm TLV and SID/Label Range TLV. Adjacency-SIDs sub-TLVs are advertised for every valid and formed adjacency, and Prefix-SID sub-TLVs are advertised for every configured Node-SID. With the above configuration applied, *Output 4-2* shows the OSPF database entry for the advertised Router Information Opaque LSA containing the SR capabilities. Immediately following the output, the RI Opaque is analysed using an on-the-wire packet capture of the LSA.

Output 4-2: Opaque-Database RI LSA

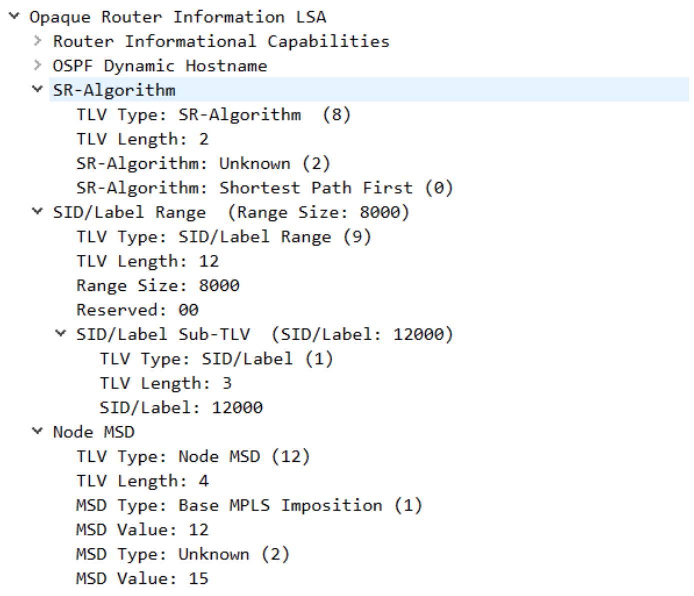
```
A:admin@R1# show router ospf opaque-database adv-router 192.0.2.1 ls-id 4.0.0.0 detail

===============================================================================
Rtr Base OSPFv2 Instance 0 Opaque Link State Database (type: All) (detail)
===============================================================================
-------------------------------------------------------------------------------
Opaque LSA
-------------------------------------------------------------------------------
Area Id           : 0.0.0.0               Adv Router Id     : 192.0.2.1
Link State Id     : 4.0.0.0               LSA Type          : Area Opaque
Sequence No       : 0x80000026            Checksum          : 0xeaca
Age               : 1382                  Length            : 68
Options           :  E
Advertisement     : Router Info
    Capabilities (1) Len 4 :
        0x38000000
    Hostname:
              shown as 0 chars, but was 2 bytes in TLV
    SR algorithm (8) Len 2 :
        0x2         0x0
    SR label range (9) Len 12 :
        Range-size=8000
        Sub-TLV SID/label(1) len 3 :
            label=12000
    Maximum Node SID Depth (12) Len 4 :
        Base MPLS Imp (1) : 12
        ERLD (2) : 15
```

Note: When searching the OSPF opaque-database for LSAs advertised by a given OSPF speaker there are likely to be multiple entries with different Link State Id's, and it may not be immediately obvious which Link State Id represents which opaque LSA type. SR-OS takes the value of the one-byte Opaque Type field and the three-byte Opaque ID from the Opaque LSA header, concatenates them, and displays them as the Link State ID. Of the more commonly used opaque types, type 1 is the Traffic Engineering LSA, type 4 is the Router Information LSA, Type 7 is the Extended Prefix LSA, and type 8 is the Extended Link LSA. For example, a Link State Id of 8.0.0.3 might be the Extended Link LSA for one link and a Link State Id of 8.0.0.4 might be the Extended Link LSA for a second link.

Figure 4-5 shows the RI Opaque LSA containing the SR-Algorithm TLV, the SID/Label Range TLV, and the Node MSD Advertisement TLV. The SR-Algorithm TLV indicates support for the base shortest path first algorithm (algorithm 0), and also for algorithm 2. This is a proprietary SR-OS implementation of a Node-SID backup that is not relevant and therefore not discussed further here. The SID/Label Range indicates a range size of 8000 with the SID/Label sub-TLV showing a start label of 12000 for a total range of 12000-19999. The Node MSD Advertisement TLV contains two sub-TLVs. The first is the BMI-MSD which shows a value of 12 labels. The second (which wireshark was unfortunately unable to decode) has an MSD-Type of 2 indicating that it is the ERLD-MSD, which has a value of 15 labels. The BMI of 12 labels and ERLD of 15 labels is standard for Nokia FP-based platforms. The BMI and ERLD for other platforms that use third-party silicon may vary, and indeed for BMI may circulate more than once through the datapath to achieve higher numbers. Supported label stack depths for different platforms are discussed in detail in chapter 6.

Figure 4-5: RI Opaque LSA



```
˅ Opaque Router Information LSA
  › Router Informational Capabilities
  › OSPF Dynamic Hostname
  ˅ SR-Algorithm
      TLV Type: SR-Algorithm  (8)
      TLV Length: 2
      SR-Algorithm: Unknown (2)
      SR-Algorithm: Shortest Path First (0)
  ˅ SID/Label Range  (Range Size: 8000)
      TLV Type: SID/Label Range (9)
      TLV Length: 12
      Range Size: 8000
      Reserved: 00
    ˅ SID/Label Sub-TLV  (SID/Label: 12000)
        TLV Type: SID/Label (1)
        TLV Length: 3
        SID/Label: 12000
  ˅ Node MSD
      TLV Type: Node MSD (12)
      TLV Length: 4
      MSD Type: Base MPLS Imposition (1)
      MSD Value: 12
      MSD Type: Unknown (2)
      MSD Value: 15
```

Output 4-3 shows the Extended Link Opaque LSA OSPF database entry for one of the routers uplinks. It is again followed by a packet capture of the same LSA with some explanatory text.

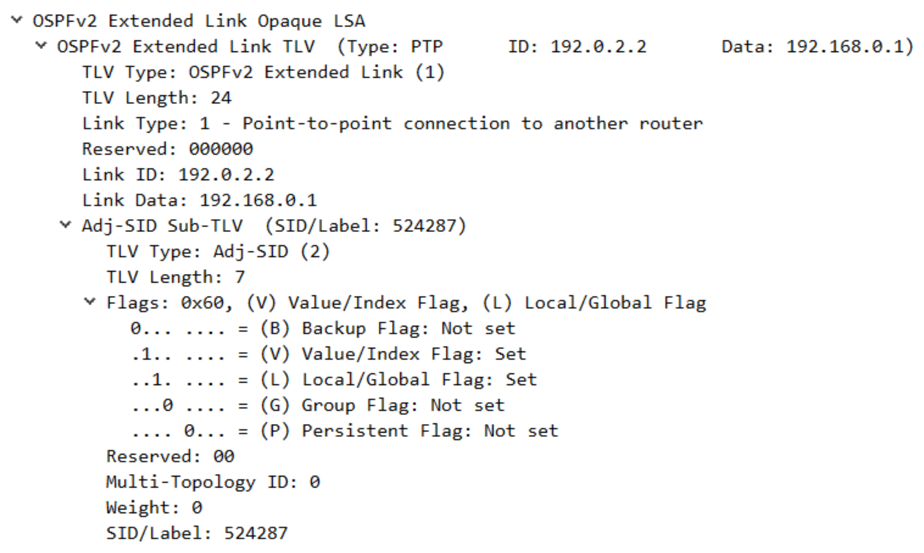
Output 4-3: Opaque-Database Extended Link LSA

```
A:admin@R1# show router ospf opaque-database adv-router 192.0.2.1 ls-id 8.0.0.3 detail

===============================================================================
Rtr Base OSPFv2 Instance 0 Opaque Link State Database (type: All) (detail)
===============================================================================
-------------------------------------------------------------------------------
Opaque LSA
-------------------------------------------------------------------------------
Area Id          : 0.0.0.0              Adv Router Id    : 192.0.2.1
Link State Id    : 8.0.0.3              LSA Type         : Area Opaque
Sequence No      : 0x80000002           Checksum         : 0xf8bf
Age              : 871                  Length           : 48
Options          :  E
Advertisement    : Extended Link
    TLV Extended link (1) Len 24  :
        link Type=P2P (1)  Id=192.0.2.2 Data=192.168.0.1
        Sub-TLV Adj-SID (2) len 7 :
            Flags=Value Local (0x60)
            MT-ID=0 Weight=0 SID/Index/Label=524287
```

As illustrated in *Figure 4-6* the Adj-Sid sub-TLV is carried within the Extended Link TLV (and Extended Link Opaque LSA). The V-flag (Value) is set to one to indicate that the advertised label is an explicit value (not an index). The L-flag (Local) is set to one indicating that the advertised label has local significance and is always set to one in SR-OS regardless of how the Adj-SID label is assigned (statically or dynamically). The B-Flag (Backup) is set to zero, indicating that the Adj-SID is not eligible for protection. If **loopfree-alternates** is enabled under OSPF, then the B-Flag will be set to one. The G-Flag (Group) is unset, as is the P-Flag (Persistence), hence upon a router restart a different Adj-SID label may be allocated. The weight field is set to zero and is currently not supported in SR-OS. Finally, the SID/Label field indicates a value of 524287.

Figure 4-6: Extended Link Opaque LSA



```
˅ OSPFv2 Extended Link Opaque LSA
  ˅ OSPFv2 Extended Link TLV  (Type: PTP      ID: 192.0.2.2       Data: 192.168.0.1)
      TLV Type: OSPFv2 Extended Link (1)
      TLV Length: 24
      Link Type: 1 - Point-to-point connection to another router
      Reserved: 000000
      Link ID: 192.0.2.2
      Link Data: 192.168.0.1
    ˅ Adj-SID Sub-TLV  (SID/Label: 524287)
        TLV Type: Adj-SID (2)
        TLV Length: 7
      ˅ Flags: 0x60, (V) Value/Index Flag, (L) Local/Global Flag
          0... .... = (B) Backup Flag: Not set
          .1.. .... = (V) Value/Index Flag: Set
          ..1. .... = (L) Local/Global Flag: Set
          ...0 .... = (G) Group Flag: Not set
          .... 0... = (P) Persistent Flag: Not set
        Reserved: 00
        Multi-Topology ID: 0
        Weight: 0
        SID/Label: 524287
```

The final TLV containing SR information is the Extended Prefix LSA containing the Prefix-SID sub-TLV, which is shown in *Output 4-4* and the corresponding packet capture in *Figure 4-7*.

Output 4-4: Opaque-Database Extended Prefix LSA

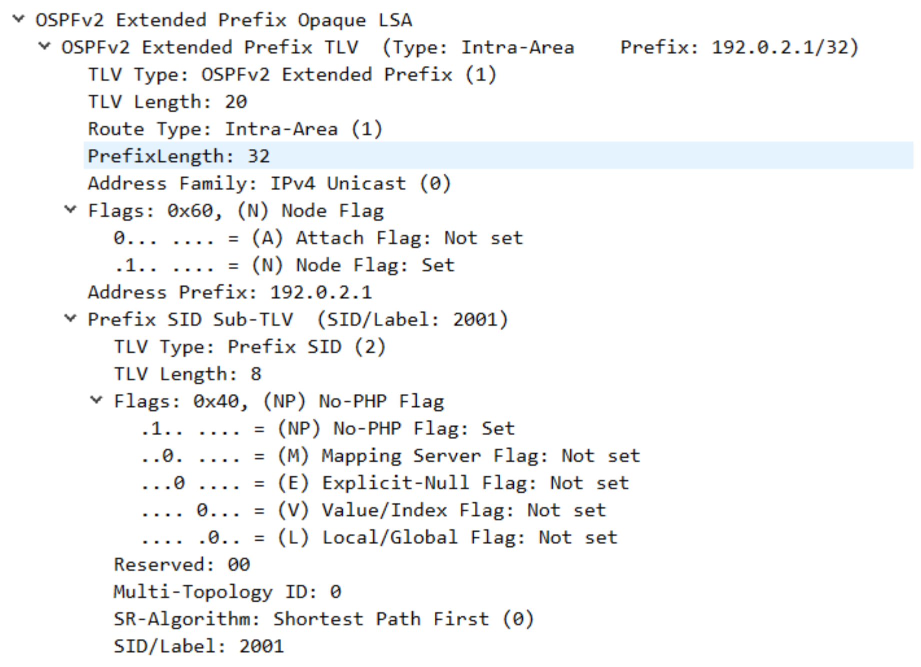
```
A:admin@R1# show router ospf opaque-database adv-router 192.0.2.1 ls-id 7.0.0.2 detail

===============================================================================
Rtr Base OSPFv2 Instance 0 Opaque Link State Database (type: All) (detail)
===============================================================================
-------------------------------------------------------------------------------
Opaque LSA
-------------------------------------------------------------------------------
Area Id          : 0.0.0.0                Adv Router Id    : 192.0.2.1
Link State Id    : 7.0.0.2                LSA Type         : Area Opaque
Sequence No      : 0x80000003             Checksum         : 0x8f1
Age              : 1119                   Length           : 44
Options          :  E
Advertisement    : Extended Prefix
    TLV Extended prefix (1) Len 20 :
        rtType=1 pfxLen=32 AF=0 pfx=192.0.2.1
            Flags=Node Elc (0x60)
        Sub-TLV Prefix SID (2) len 8 :
            Flags=noPHP (0x40)
            MT-ID=0 Algorithm=0 SID/Index/Label=2001
```

In the Extended Prefix TLV itself, the prefix is the system interface of 192.0.2.1 and it has an Address-Family of IPv4 Unicast indicated by the AF value of 0 (the only value currently defined). The route type is 1 meaning that the route is intra-area. Lastly, the flags field of the Extended Prefix TLV has the N-Flag set because the prefix identifies the advertising router, and the E-Flag set to indicate that the router is capable of processing Entropy Labels.

In the Prefix-SID sub-TLV, The NP-Flag is set to indicate no penultimate hop popping is to be performed for this SID. SR-OS always sets the NP-flag to one for Node-SIDs and this is currently non-configurable. The remaining flags are set to zero, notably the V-flag (Value) is set to zero indicating that the advertised label is an index, while the L-flag (Local) is set to zero to indicate that the label is not locally significant (it is globally unique). The algorithm associated with the prefix the base shortest path first algorithm (algorithm 0) which is the default behaviour. Finally, the advertised label is 2001, and since it is an index value it is combined with the start-label of 12000 to yield an absolute value of 14001.

Figure 4-7: Extended Prefix Opaque LSA



```
v OSPFv2 Extended Prefix Opaque LSA
  v OSPFv2 Extended Prefix TLV  (Type: Intra-Area    Prefix: 192.0.2.1/32)
      TLV Type: OSPFv2 Extended Prefix (1)
      TLV Length: 20
      Route Type: Intra-Area (1)
      PrefixLength: 32
      Address Family: IPv4 Unicast (0)
    v Flags: 0x60, (N) Node Flag
        0... .... = (A) Attach Flag: Not set
        .1.. .... = (N) Node Flag: Set
      Address Prefix: 192.0.2.1
    v Prefix SID Sub-TLV  (SID/Label: 2001)
        TLV Type: Prefix SID (2)
        TLV Length: 8
      v Flags: 0x40, (NP) No-PHP Flag
          .1.. .... = (NP) No-PHP Flag: Set
          ..0. .... = (M) Mapping Server Flag: Not set
          ...0 .... = (E) Explicit-Null Flag: Not set
          .... 0... = (V) Value/Index Flag: Not set
          .... .0.. = (L) Local/Global Flag: Not set
        Reserved: 00
        Multi-Topology ID: 0
        SR-Algorithm: Shortest Path First (0)
        SID/Label: 2001
```

A number of operational commands are available in SR-OS to validate the integrity of locally assigned and learned SIDs. The Adj-SID assigned to a given interface can be verified using the command shown in *Output 4-5*. In this example the router has multiple interfaces, so I have used pipe/match commands to reduce the size of the output. As shown in the output, the adjacency to neighbour router 192.0.2.2 over link-to-R2 has an Adj-SID of 524287 while the adjacency to neighbor router 192.0.2.4 over link-to-R4 has an Adj-SID of 524286.

Output 4-5: OSPF Adjacency with Adj-SID

```
*A:R1# show router ospf neighbor detail | match expression "Neighbor Rtr Id|Adj SR SID"
Neighbor Rtr Id : 192.0.2.2  Interface: link-to-R2
Area Id         : 0.0.0.0               Adj SR SID        : Label 524287
Neighbor Rtr Id : 192.0.2.4  Interface: link-to-R4
Area Id         : 0.0.0.0               Adj SR SID        : Label 524286
```

Prefix-SIDs learned locally and from other SR routers in the domain can be displayed using the command illustrated in *Output 4-6*. The topology of the SR domain is largely irrelevant at this point, but as can be observed there are six routers in the domain, R1 through R6, with system addresses of 192.0.2.1/32 through 192.0.2.6/32 respectively. The SID values are displayed as an index, and since all routers have the same SRGB (12000-19999) the Node-SID labels can be derived as 14001 through 14006 respectively. As previously described, the N flag indicates a Node-SID, and the nP flag indicates that PHP should not be carried out by the penultimate hop router for this SID.

Output 4-6: OSPF Prefix-SIDs

```
A:admin@R1# show router ospf prefix-sids
```

```
===============================================================================
Rtr Base OSPFv2 Instance 0 Prefix-Sids
===============================================================================
Prefix                             Area            RtType     SID
                                   Adv-Rtr         SRMS       Flags
-------------------------------------------------------------------------------
192.0.2.1/32                       0.0.0.0         INTRA-AREA 2001
                                   192.0.2.1       N          NnP
192.0.2.2/32                       0.0.0.0         INTRA-AREA 2002
                                   192.0.2.2       N          NnP
192.0.2.3/32                       0.0.0.0         INTRA-AREA 2003
                                   192.0.2.3       N          NnP
192.0.2.4/32                       0.0.0.0         INTRA-AREA 2004
                                   192.0.2.4       N          NnP
192.0.2.5/32                       0.0.0.0         INTRA-AREA 2005
                                   192.0.2.5       N          NnP
192.0.2.6/32                       0.0.0.0         INTRA-AREA 2006
                                   192.0.2.6       N          NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 6
```

Another useful command for validating the status of SR tunnels is shown in *Output 4-7*. Unfortunately, as this is a *tools* command it is not strictly maintained and does not provide any specific filtering arguments to search for a particular tunnel other than in-label (pipe and match is still possible of course). Regardless, the output provides a good overview of the router's behaviour for each tunnel together with the label stack and outgoing interface (or interfaces if **loopfree-alternates** is enabled and a backup path has been computed). For a local Node-SID the Incoming Label Map (ILM) table is programmed with a pop operation and no out-label is shown. For local Adj-SIDs the ILM is also programmed with a pop operation but it is treated as a label-switch to implicit-null, which is why the Adj-SIDs in the output have an out-label of 3 (implicit-null).

Output 4-7: SR Tunnels Dump Output

```
A:admin@R1# tools dump router segment-routing tunnel
==========================================================================================
Legend: (B) - Backup Next-hop for Fast Re-Route
        (D) - Duplicate
label stack is ordered from top-most to bottom-most
==========================================================================================
------------------------------------------------------------------------------------------+
 Prefix                                                                                   |
 Sid-Type        Fwd-Type       In-Label  Prot-Inst(algoId)                               |
                 Next Hop(s)                                  Out-Label(s) Interface      |
------------------------------------------------------------------------------------------+
 192.0.2.1
 Node            Terminating    14001     OSPF-0
 192.0.2.2
 Node            Orig/Transit   14002     OSPF-0
                 192.168.0.2                                   14002       link-to-R2
 192.0.2.3
 Node            Orig/Transit   14003     OSPF-0
                 192.168.0.2                                   14003       link-to-R2
 192.0.2.4
 Node            Orig/Transit   14004     OSPF-0
                 192.168.0.10                                  14004       link-to-R4
 192.0.2.5
 Node            Orig/Transit   14005     OSPF-0
                 192.168.0.2                                   14005       link-to-R2
                 192.168.0.10                                  14005       link-to-R4
 192.0.2.6
 Node            Orig/Transit   14006     OSPF-0
                 192.168.0.2                                 14006      link-to-R2
                 192.168.0.10                                14006      link-to-R4
 192.168.0.10
 Adjacency       Transit        524286    OSPF-0
                 192.168.0.10                                3          link-to-R4
 192.168.0.2
 Adjacency       Transit        524287    OSPF-0
                 192.168.0.2                                 3          link-to-R2
-------------------------------------------------------------------------------+
No. of Entries: 8
-------------------------------------------------------------------------------+
```

When a Node-SID is learned through OSPF and the associated prefix is resolved as reachable it is programmed in the tunnel table. In *Output 4-8* there are single entries for Node-SIDs 192.0.2.2, 192.0.2.3, and 192.0.2.4, and two entries for Node-SIDs 192.0.2.5 and 192.0.2.6 with different next-hops as there are Equal Cost Multipath (ECMP) routes to these Node-SIDs.

There are also entries for 192.168.0.2/32 and 192.168.0.10/32 which are the next-hops for each of the router's adjacent neighbours, and these require some explanation. When SR-TE is deployed an ingress LER may push on a label stack where each label represents a link or a node (or possibly some other form of segment). SR-TE LSPs consisting of multiple labels (segments) are internally modelled by SR-OS as hierarchical constructs. The SR-TE LSP may have a number of labels imposed, but the ingress LER only needs to track the reachability of the first strict or loose hop as this hop is all that needs to be resolved for the LSP to be considered valid. This first hop represents the Next-Hop Label Forwarding Entry (NHLFE) of the SR shortest path tunnel, and this is maintained in the tunnel-table as a Node-SID or Adj-SID (the remainder of the label stack if not meaningful to the forwarding decision made by the ingress LER). This hierarchical construct is referred to as a *super NHLFE* and is tunnelled over the NHLFE of the first hop. SR-OS keeps an entry for the SR shortest path tunnel to a downstream Node-SID or Adj-SID in the tunnel table so that its NHLFE is readily available for use by super NHLFEs, hence the two entries exist for R1's adjacent neighbours (192.168.0.2/32 and 192.168.0.10/32) as well as all learned Node-SIDs. The purpose of this super NHLFE concept is to optimise NHLFE resource utilisation, particularly when many SR-TE LSPs have the same first next-hop. Each SR-TE LSP becomes its own super NHLFE but they all use a common NHLFE (or set of NHLFEs if ECMP is present). This concept of super NHLFEs will be discussed and illustrated numerous times throughout this document.

The last point of note is the preference of 10 assigned to SR-OSPF. This is higher than RSVP (7), SR-TE (8), and LDP (9), but can be changed in the **configure router ospf segment-routing** context using the **tunnel-table-pref** command. This is discussed further in chapter 8 where the use of SR for BGP services is covered, together with some potential migration approaches.

Output 4-8: SR-OSPF Tunnel Table

```
A:admin@R1# show router tunnel-table

===============================================================================
IPv4 Tunnel Table (Router: Base)
```

```
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.2/32          ospf (0)  MPLS  524320    10     192.168.0.2    100
192.0.2.3/32          ospf (0)  MPLS  524319    10     192.168.0.2    200
192.0.2.4/32          ospf (0)  MPLS  524318    10     192.168.0.10   100
192.0.2.5/32          ospf (0)  MPLS  524317    10     192.168.0.2    200
192.0.2.5/32          ospf (0)  MPLS  524317    10     192.168.0.10   200
192.0.2.6/32          ospf (0)  MPLS  524316    10     192.168.0.2    300
192.0.2.6/32          ospf (0)  MPLS  524316    10     192.168.0.10   300
192.168.0.2/32        ospf (0)  MPLS  524314    10     192.168.0.2    0
192.168.0.10/32       ospf (0)  MPLS  524315    10     192.168.0.10   0
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
===============================================================================
```

5 LDP and SR Interworking

An MPLS Control-Plane Client (MCC) refers to any control-plane protocol that installs forwarding entries into the MPLS data plane such as SR, LDP, RSVP-TE, and BGP. When these MCCs install entries into the data plane they need to ensure that any incoming labels that they allocate are unique. Typically, LDP, RSVP-TE and BGP share a common label space on a given router and use the concept of a label manager to ensure that labels are uniquely allocated. SR makes use of the SRGB for label allocation, which therefore allows SR to co-exist with any other MCC and function as ships-in-the-night.

Where these MCCs co-exist across the entirety of a network, services may independently select an MPLS transport protocol, with methods for selecting that protocol being implementation specific. This level of co-existence is relatively simple and clear-cut. However, as SR is deployed over time there may be cases where SR is present in one part of the network and LDP is present in another part. In this scenario it might be useful to have an interworking function to allow VPN services in the SR part of the network to migrate to SR whilst maintaining connectivity with the VPN sites that remain in the LDP part of the network. This is entirely possible using SR to LDP interworking [11].

SR Mapping Server

SR to LDP interworking requires the use of an SR Mapping Server (SRMS). The SRMS functionality consists of two functional blocks, the Mapping Server (MS) and the Mapping Client (MC).

The MS is a control-plane-only function that can be located anywhere in the SR domain and whose function is to advertise Prefix-SIDs on behalf of non-SR routers. Multiple Mapping Servers can exist, and of course this would be recommended for the purpose of redundancy. The Prefix-SIDs that each of the redundant Mapping Servers advertises should be consistent with other Mapping Servers, but in the event that they are not rules exist as to how these Prefix-SID should be resolved.

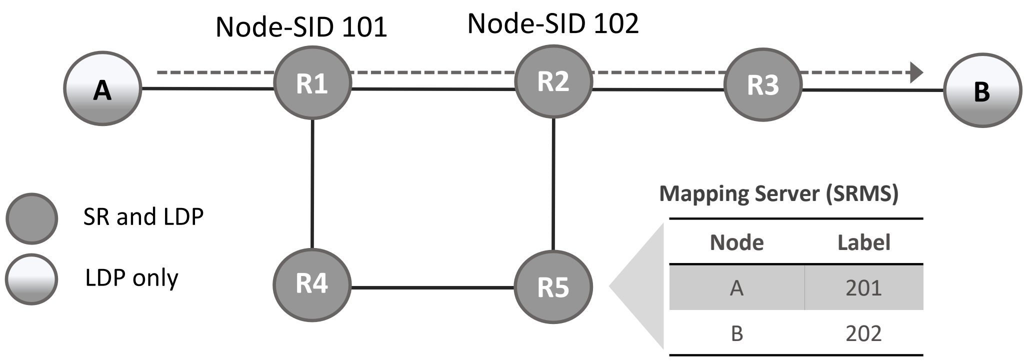
An MC is a data-plane function that receives the MS mapping advertisements and interprets each SR-mapping advertisement as an assignment of a Prefix-SID to a prefix belonging to a non-SR router. The MC can then create stitching points between LDP FECs and SR Prefix-SIDs (Node-SIDs) for the same prefix. This provides two benefits:

i. It allows for the use of a heterogenous end-to-end MPLS transport protocol consisting of SR and LDP.
ii. It allows LDP tunnels to be stitched to fast reroute backup SR tunnels that are computed using Topology Independent LFA (TI-LFA), thus extending fast reroute coverage for LDP. This is discussed further in chapter 9.

Like the Mapping Server, Mapping Clients can and should be redundant. Indeed, every SR router in an SR domain may operate as a Mapping Client if required.

An example how SR and LDP can interwork to provide end-to-end services is illustrated in *Figure 5-1*. Here routers R1 through R5 run SR and LDP, whilst routers A and B are non-SR routers running LDP only. Router R5 is the SRMS and advertises Node-SIDs 201 and 202 for the LDP-only routers A and B respectively. Router A forwards a packet to R1 using the LDP FEC B. As R1 runs both SR and LDP, it will have an LDP label binding for its next-hop R2 for FEC B, but it will also have Node-SID 202. In this case the selection of SR over LDP for the next-hop is a local implementation matter, but for this example R1 selects SR and therefore swaps its local LDP label for FEC B to Node-SID 202 and forwards the packet to R2. R2 swaps Node-SID 202 for Node-SID 202 and forwards to R3. R3 knows that B is not SR-capable (as B did not advertise an SR capability in IS-IS/OSPF), so R3 swaps Node-SID 202 for LDP FEC B.

Figure 5-1: SR-LDP Interworking

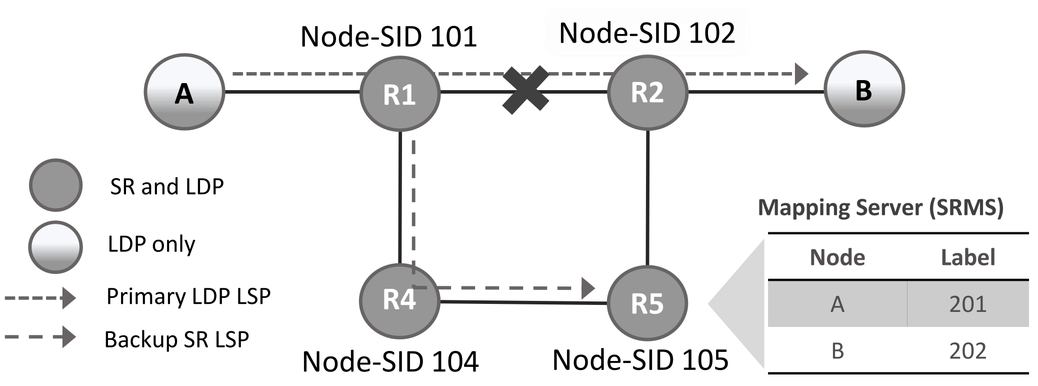


Node	Label
A	201
B	202

A key point to note from the above example is that the LDP LSP is not tunnelled inside SR. The LDP label is removed before forwarding by SR, hence the LDP FEC and the SR Node-SID are stitched by the MC router R1. Similarly, at R3 the SR Node-SID is stitched to an LDP FEC.

An example of using SR to provide LDP fast reroute (and potentially extend coverage) is shown in *Figure 5-2*. Here Router A has an LSP to Router B and the intention is to provide this LSP with protection against a failure of the link R1-R2. In steady state LDP is used as the preferred transport tunnel for the LSP from router A to router B and the path is A-R1-R2-B. Upon failure of the link R1-R2, R1 swaps the incoming LDP label from A with the Node-SID for B (202). R1's backup is an RLFA repair tunnel to R5, so R1 pushes on the label stack {105, 202} and forwards the packet to R4. R4 swaps the active label 105 for label 105 and forwards the packet towards R5 (there is no PHP in this example). R5 pops the active label 105 and swaps label 202 for label 202 before forwarding to R2. R2 knows that its next-hop B is not SR-capable, so R2 swaps label 202 for the LDP label advertised by B.

Figure 5-2: Using SR for LDP Fast Reroute



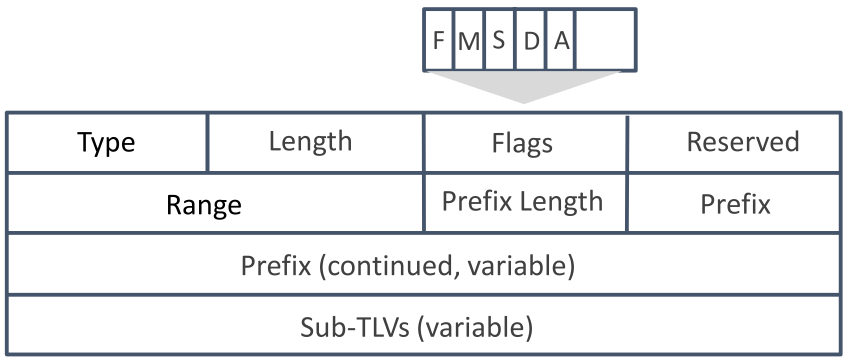
Node	Label
A	201
B	202

SID/Label Binding TLV

To originate Prefix-SIDs from a Mapping Server IS-IS uses a SID/Label Binding TLV containing a Prefix-SID sub-TLV, and OSPF uses the Prefix-SID sub-TLV covered in chapter 4 with the Mapping Server flag (M-flag) set. Both allow for advertisement of one or more SID index/labels and one or more prefixes.

The format of the IS-IS SID/Label Binding TLV is shown in *Figure 5-3*. The range field allows the SRMS to advertise a range of addresses and their associated Prefix-SIDs. This allows for compression of Prefix-SIDs advertised by an SRMS but does impose the restriction that the prefix and corresponding SID/Label Block are both contiguous. If a single SID is advertised the range field is simply set to one. The prefix and prefix length represent the prefix or prefixes being advertised. If range is set to 1, it is a single prefix. If range is set to greater than one the prefix represents the first address to which a SID is to be assigned.

Figure 5-3: IS-IS SID/Label Binding TLV



The flags field contains the following flags:

F-Flag	Address Family flag. If unset the prefix carries an IPv4 prefix. If set the prefix carries an IPv6 prefix.
M Flag	Mirror Context flag. Set if the advertised SID corresponds to a mirrored context used for Node backup purposes [12].

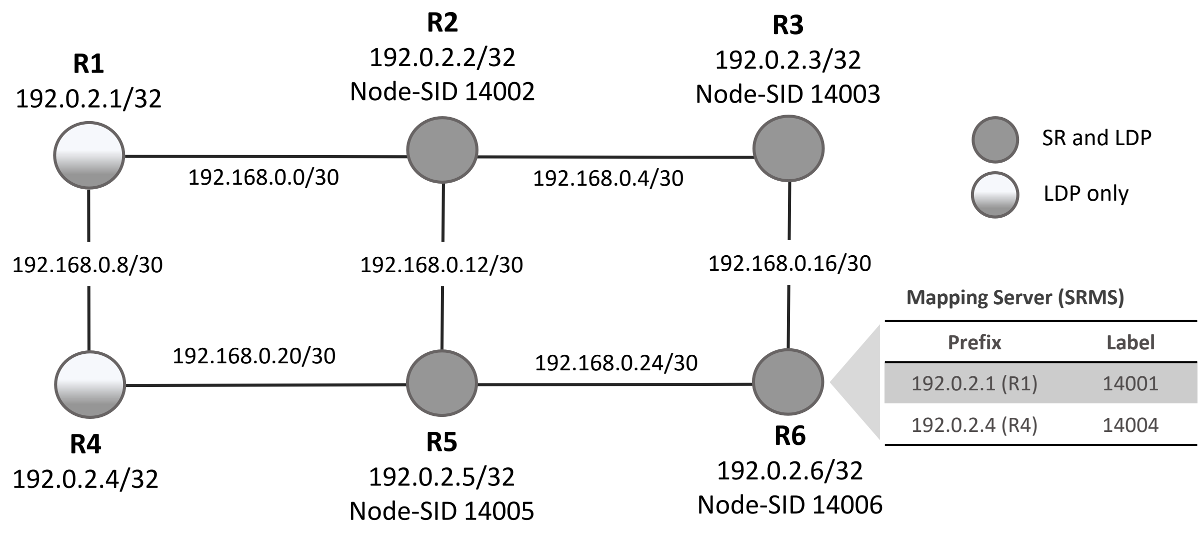
S-Flag	If set the SID/Label Binding TLV should be flooded across the entire routing domain. If not set the SID/Label Binding TLV must not be leaked between levels.
D-Flag	When the SID/Label Binding TLV is leaked from Level-2 to Level-1 the D-flag must be set. Note that the D-Flag is not set when leaking from Level-1 to Level-2.
A-Flag	Attached Flag. Allows the advertising router to indicate whether or not the advertised prefix(es) is/are directly connected.

The Prefix-SID sub-TLV described in chapter 3 is carried as a sub-TLV of the SID/Label Binding TLV and contains the SID/Index/Label value associated with the prefix and range. If the prefix range is greater than one the Prefix-SID sub-TLV will contain the first SID/Index/Label value of the contiguous advertised range.

Configuration

The topology in *Figure 5-4* is used to demonstrate SR to LDP interworking. Routers R1 through R6 all reside in a single IS-IS Level-2 domain and their associated system addresses together with Node-SIDs where appropriate are shown. Routers R2, R3, R5, and R6 run both SR and link-layer LDP. Routers R1 and R4 are non-SR routers running only link-layer LDP. Router R6 therefore functions as an SRMS and advertises Prefix-SIDs on behalf of R1 and R4. The topology represents a network that has deployed LDP end-to-end but only partially deployed SR.

Figure 5-4: SRMS Test Topology



Output 5-1 shows the configuration at R6 to enable SRMS functionality and advertise Prefix-SIDs for routers R1 and R4. Within the mapping-server context two individual prefixes are specified together with their associated SID index values. An optional **set-flags** argument exists for each prefix to allow for the setting of the S-flag to indicate to other IS-IS routers in the domain that the SID/Label Binding TLV should be flooded through the entire (Level-

1/Level-2) domain. Because the example topology is entirely Level-2, flooding is implicit and therefore the S-Flag is not used.

Output 5-1: R6 SRMS Configuration

```
        isis 0 {
            segment-routing {
                mapping-server {
                    admin-state enable
                    node-sid-map 2001 {
                        ip-prefix 192.0.2.1/32
                    }
                    node-sid-map 2004 {
                        ip-prefix 192.0.2.4/32
                    }
                }
            }
        }
```

In this example it is not possible to use a range to advertise the prefixes as they are non-contiguous. Hence, each of the two prefixes will be advertised by R6 as an individual SID/Label Binding TLV. This is illustrated in *Output 5-2* which shows two separate SID/Label Binding TLV entries, each with a range of 1 to indicate a single advertised prefix. The `bFlgs` field indicates Binding SID flags, and this has the F-Flag (Address Family flag) unset to show that the advertised prefixes are IPv4. Both TLVs contain a Prefix-SID sub-TLV advertising the SID index and algorithm associated with each prefix. The `pFlgs` field indicates Prefix-SID sub-TLV flags and shows that the N-Flag (Node-SID) flag is set.

Output 5-2: IS-IS SID/Label Binding TLVs

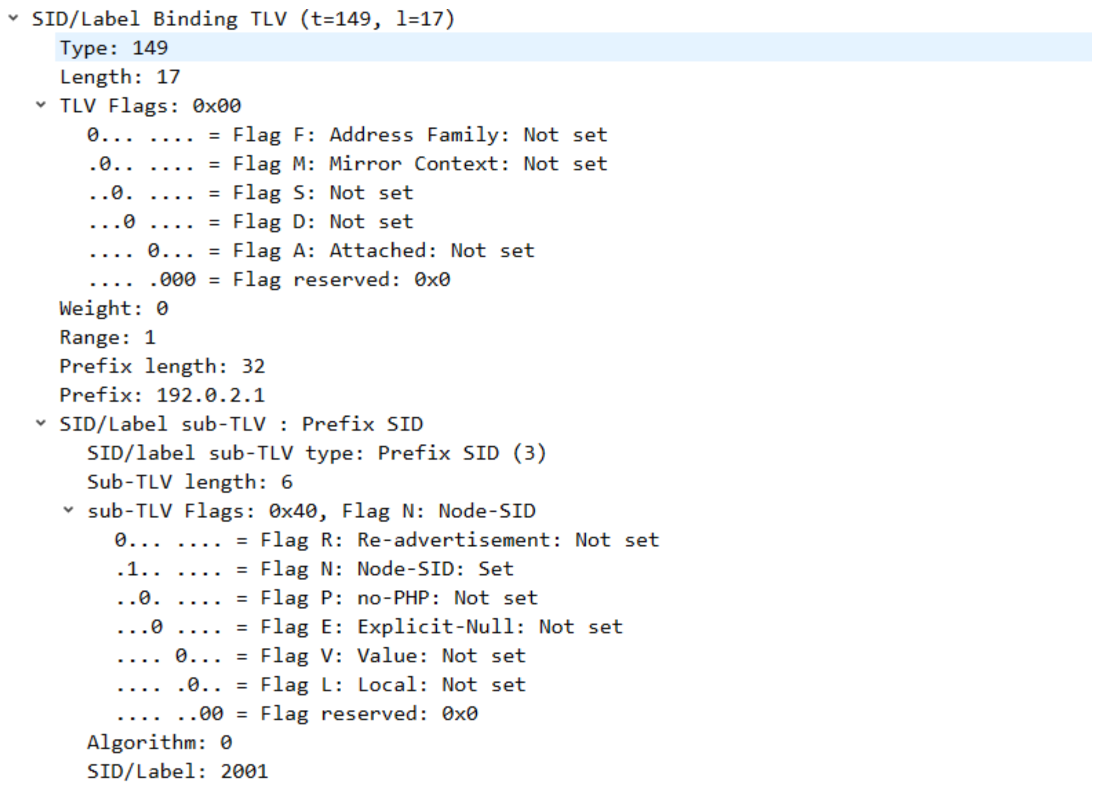
```
A:admin@R6# show router isis database R6.00-00 detail level 2

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================
... [snip]
TLVs :
  SID Label Binding:
    Prefix: 192.0.2.1/32 Range:1 Weight:0 bFlgs:v4 SID:2001 Algo:0 pFlgs:N
  SID Label Binding:
    Prefix: 192.0.2.4/32 Range:1 Weight:0 bFlgs:v4 SID:2004 Algo:0 pFlgs:N

... [snip]
```

Figure 5-5 shows a packet capture of the SID/Label Binding TLV for prefix 192.0.2.1/32. For conciseness, only a single SID/Label Binding TLV (of the two advertised) is shown.

Figure 5-5: SID/Label Binding TLV



```
˅ SID/Label Binding TLV (t=149, l=17)
    Type: 149
    Length: 17
  ˅ TLV Flags: 0x00
      0... .... = Flag F: Address Family: Not set
      .0.. .... = Flag M: Mirror Context: Not set
      ..0. .... = Flag S: Not set
      ...0 .... = Flag D: Not set
      .... 0... = Flag A: Attached: Not set
      .... .000 = Flag reserved: 0x0
    Weight: 0
    Range: 1
    Prefix length: 32
    Prefix: 192.0.2.1
  ˅ SID/Label sub-TLV : Prefix SID
      SID/label sub-TLV type: Prefix SID (3)
      Sub-TLV length: 6
    ˅ sub-TLV Flags: 0x40, Flag N: Node-SID
        0... .... = Flag R: Re-advertisement: Not set
        .1.. .... = Flag N: Node-SID: Set
        ..0. .... = Flag P: no-PHP: Not set
        ...0 .... = Flag E: Explicit-Null: Not set
        .... 0... = Flag V: Value: Not set
        .... .0.. = Flag L: Local: Not set
        .... ..00 = Flag reserved: 0x0
      Algorithm: 0
      SID/Label: 2001
```

Once the mapping server has advertised prefixes for the non-SR routers in the topology it is the job of the mapping clients (MC's) to interpret those advertisements and create the stitching points between LDP and SR. In the simple test topology shown in *Figure 5-4* it should be reasonably clear that only routers R2 and R5 need to function as MC's for the non-SR routers R1 and R4, but in reality, every SR router in an SR domain can function as an MC in the data-plane. Indeed, one of the benefits of LDP to SR interworking is that it provides the ability to extend fast reroute coverage for LDP by using RLFA/TI-LFA computed SR tunnels as backup tunnels. In this scenario all SR routers in the domain would need to be MC's for each to be able to act as a potential point of local repair (PLR).

When functioning as an MC and implementing SR/LDP interworking in SR-OS, the behaviour, and requirements for stitching LDP to SR differ slightly to those when stitching SR to LDP:

i. In the LDP to SR data-plane direction, LDP uses an **export-tunnel-table** command to reference an export policy. The LDP process monitors the tunnel-table and if there is a /32 SR tunnel of type SR-ISIS (or SR-OSPF) that matches a prefix in the export policy, LDP programs an LDP ILM entry and stitches it to the SR Node-SID tunnel endpoint. LDP also originates a FEC for that prefix and advertises it to its peers.

ii. In the SR to LDP data-plane direction the SRMS provides a network-wide policy for the prefixes that SR needs to stitch to a corresponding LDP FEC. As a result, the IS-IS/OSPF segment-routing context generically uses the **export-tunnel-table ldp** command with no reference to a policy. Whenever a /32 LDP tunnel destination matches a prefix for which a Prefix-SID sub-TLV was received from an SRMS, the SR ILM entry is stitched to the corresponding LDP tunnel endpoint.

Output 5-3 shows the configuration applied at R2 to function as an MC and implement LDP to SR stitching. Similar configuration is also applied at R5. For the LDP to SR data-plane direction the LDP context contains the command **export-tunnel-table** which references the policy “dataplane-LDP-to-SR”. This policy looks for tunnel-table entries for 192.0.2.3/32 (R3) and 192.0.2.3.6/32 (R6) that are from protocol IS-IS. If an SR-ISIS tunnel exists to either of those locations R2 will originate an LDP FEC for the prefix(es), program an LDP ILM entry for the prefix(es) and stitch that ILM entry to the corresponding SR Node-SID for the same prefix. The LDP context also has a command **prefer-protocol-stitching true** that requires some explanation. By default, when attempting to resolve an LDP ILM entry to a NHLFE SR-OS will prefer an LDP NHLFE if one exists. In this topology the routers R2, R3, R5, and R6 run both SR and LDP, hence an LDP-advertised NHLFE and an SR-advertised NHLFE will exist at R2 for all of those destinations. As a result, R2 will stitch the LDP ILM entry to the LDP NHLFE and forward the packet using an LDP FEC to its next-hop R3 (creating an end-to-end LDP LSP). Where SR and LDP co-exist like this, it creates an asymmetric forwarding model where one direction (in this case right to left) uses SR where possible, and the other direction (in this case left to right) uses end-to-end LDP. The purpose of the **prefer-protocol-stitching true** command is to instruct R2 to prefer an SR NHLFE to an LDP NHLFE if one exists, and hence R2 will create a stitch between the LDP ILM and an SR Node-SID if it is present.

At the time of writing **prefer-protocol-stitching** is not currently supported on 7250 IXR generation one or generation two platforms. Please check Release Notes for an updated list of 7250 IXR unsupported features.

In the SR to LDP data-plane direction the **isis segment-routing** context simply contains the command **export-tunnel-table ldp** as the prefixes advertised by the SRMS determine the policy to instruct the router to stitch an SR ILM entry to the corresponding LDP tunnel endpoint.

Output 5-3: R2 Mapping Client Configuration

```
    policy-options {
        prefix-list "sr-domain" {
            prefix 192.0.2.3/32 type exact {
            }
            prefix 192.0.2.6/32 type exact {
            }
        }
        policy-statement "dataplane-LDP-to-SR" {
            entry 10 {
                from {
                    prefix-list ["sr-domain"]
                    protocol {
                        name [isis]
                    }
                }
                to {
                    protocol {
                        name [ldp]
                    }
                }
                action {
                    action-type accept
                }
            }
        }
    }
    router "Base" {
        isis 0 {
            segment-routing {
                export-tunnel-table ldp
            }
        }
        ldp {
            prefer-protocol-stitching true
            export-tunnel-table ["dataplane-LDP-to-SR"]
        }
       }
```

Output 5-4 shows the prefix SIDs advertised by the SRMS R6 as seen at router R2. Prefix 192.0.2.1/32 and 192.0.2.4/32 are advertised by the SRMS using the SID/Label Binding TLV on behalf of the non-SR routers R1 and R4 respectively and are identified as such by the Y flag in the SRMS column. The (S) flag indicates that the SRMS Prefix SID has been selected to be programmed. Prefix 192.0.2.6/32 is R6's system address and this is advertised using a conventional Prefix-SID sub-TLV. All prefixes have the Node-SID flag set and the nP flag to indicate that PHP should not be carried out for these prefixes.

Output 5-4: Prefix-SIDs Advertised by R6

```
A:admin@R2# show router isis prefix-sids adv-router R6

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                            SID        Lvl/Typ     SRMS   AdvRtr
                                                          MT     Flags
-------------------------------------------------------------------------------
192.0.2.1/32                      2001       2/Int.       Y(S)  R6
                                                            0       NnP
192.0.2.4/32                      2004       2/Int.       Y(S)  R6
                                                            0       NnP
192.0.2.6/32                      2006       2/Int.       N     R6
                                                            0       NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 3 (3 unique)
-------------------------------------------------------------------------------
```

To verify that the LDP to SR interworking is functioning I'll consider the path from the non-SR router R1 to the SR routers R3 and R6. *Output 5-6* shows the active LDP bindings at R1 for prefix 192.0.2.3/32 (R3) and 192.0.2.6/32 (R6) truncated to show only the push entries (there are also swap entries which are not shown for conciseness). Prefix 192.0.2.3/32 has an egress label of 524283 and a next-hop of R2 (192.168.0.2). Because R1 has ECMP paths to R6 the prefix 192.0.2.6/32 has two entries. The first entry has an egress label of 524279 and a next-hop of R2 (192.168.0.2), and the second has an egress label of 524280 and a next-hop of R4 (192.168.0.10).

Output 5-5: Active LDP Bindings at R1

```
A:admin@R1# show router ldp bindings active ipv4

===============================================================================
LDP Bindings (IPv4 LSR ID 192.0.2.1)
             (IPv6 LSR ID ::)
===============================================================================
LDP IPv4 Prefix Bindings (Active)
===============================================================================
Prefix                                      Op
IngLbl                                      EgrLbl
EgrNextHop                                  EgrIf/LspId
-------------------------------------------------------------------------------
...[snip]
192.0.2.3/32                                Push
  --                                        524283
192.168.0.2                                 1/1/c1/1:100
...[snip]
192.0.2.6/32                                Push
  --                                        524279
192.168.0.2                                 1/1/c1/1:100

192.0.2.6/32                                Push
  --                                        524280
192.168.0.10                                1/1/c2/1:100
```

The LDP bindings at R1 can subsequently be correlated with the LDP to SR stitching in place at R2. *Output 5-6* shows the active LDP bindings at R2 again truncated to show only prefix 192.0.2.3/32 (R3) and 192.0.2.6 (R6). Both prefixes have an 'I' flag set to indicate that they have an SR-ISIS next-hop. Prefix 192.0.2.3/32 has an incoming label of 524283 correlating to the LDP label pushed by R1, and an egress label of R3's Node-SID 14003. There are two entries for prefix 192.0.2.6/32 as R2 has ECMP paths to R6. Both entries have an incoming label of 524279 again correlating to the LDP label pushed by R1, and an egress label of R6's Node-SID 14006.

Output 5-6: LDP Active Bindings at R2

```
A:admin@R2# show router ldp bindings active ipv4

===============================================================================
LDP Bindings (IPv4 LSR ID 192.0.2.2)
             (IPv6 LSR ID ::)
===============================================================================
LDP IPv4 Prefix Bindings (Active)
===============================================================================
Prefix                                      Op
IngLbl                                      EgrLbl
EgrNextHop                                  EgrIf/LspId
-------------------------------------------------------------------------------
...[ snip ]
192.0.2.3/32(I)                             Swap
524283                                      14003
192.168.0.6                                 1/1/c4/1:100

...[snip]
192.0.2.6/32(I)                             Swap
524279                                      14006
192.168.0.6                                 1/1/c4/1:100

192.0.2.6/32(I)                             Swap
524279                                      14006
192.168.0.14                                1/1/c3/1:100
```

To verify that the SR to LDP interworking is functioning I'll use the path from R3 to R1. *Output 5-7* shows the IS-IS Prefix-SID for prefix 192.0.2.1/32 (R1) with SID (index) of 2001. The SRGB in use is 12000-19999, hence R1's Node-SID is {12000 + 2001} 14001. Again, the output shows that the Prefix-SID is advertised by an SRMS and that the advertising router is R6.

Output 5-7: IS-IS Prefix-SID for R1 at Router R3

```
A:admin@R3# show router isis prefix-sids ip-prefix-prefix-length 192.0.2.1/32

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                            SID        Lvl/Typ    SRMS   AdvRtr
                                   Shared     Algo       MT     Flags
-------------------------------------------------------------------------------
192.0.2.1/32                      2001       2/Int.      Y(S)  R6
                                       N.A.       0          0      NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 1 (1 unique)
-------------------------------------------------------------------------------
```

Output 5-8 shows the SR tunnel table at router R2 that is functioning as a Mapping Client. Prefix 192.0.2.1 (R1) has an incoming label of (Node-SID) 14001 learnt through IS-IS, and an outgoing interface representing the Tunnel ID for the LDP-learnt FEC for router R1.

Output 5-8: SR Tunnel Table at R2

```
A:admin@R2# tools dump router segment-routing tunnel
========================================================================================
Legend: (B) - Backup Next-hop for Fast Re-Route
        (D) - Duplicate
label stack is ordered from top-most to bottom-most
========================================================================================
----------------------------------------------------------------------------------------+
 Prefix                                                                                 |
 Sid-Type        Fwd-Type       In-Label  Prot-Inst(algoId)                             |
                 Next Hop(s)                          Out-Label(s) Interface/Tunnel-ID |
----------------------------------------------------------------------------------------+
 192.0.2.1
 Node            Orig/Transit   14001     ISIS-0
                 192.0.2.1                            -            65549(LDP)

 --- [snip] ---
```

With this LDP to SR interworking in place it is entirely possible for BGP and services to use heterogenous end-to-end LSPs where non-SR routers use LDP, SR routers use SR, and the mapping clients provide the stitching capability between the two. This allows operators to start to use SR in areas of the network where it is deployed whilst maintaining connectivity to areas of the network where SR is not yet (or can't be) deployed. This also has the additional benefit of potentially extending LFA coverage for LDP-based LSPs and this is discussed further in chapter 9.

6 SR Traffic Engineering (SR-TE)

The headend of an SR-TE LSP creates source-routed traffic engineered paths by imposing ordered lists of segments, where each segment represents a node, or a link, or an instruction. SR-TE can be applied both to the MPLS data-plane (SR-MPLS) and the IPv6 data-plane (SRv6). This chapter focuses on SR-MPLS. SRv6 is covered in chapter 12.

SR-TE LSPs may use a distributed or centralised control plane. With a distributed control plane, an SR router individually and autonomously computes the path using segments advertised in the IGP or through BGP. With a centralised control plane, a controller is responsible for computing the path which is then communicated to the node using the Path Computation Element Protocol (PCEP) or BGP using the SR-TE Policy address family. This chapter looks at distributed SR-TE followed by a look at centralised SR-TE using PCEP but starts with a look at the implications on label stack imposition.

SR-TE Label Stack

Before discussing either distributed or centralised SR-TE, it’s worth spending some time on an issue that is applicable to both mechanisms, and that is the control of SR label stack depth on the router.

Each SR router will have hardware limitations with regard to the number of labels that it can impose on an SR packet. For FP-based products it is 12, but for merchant-silicon-based products the number will vary based on generation of packet processor and the ability to recycle packets through the forwarding plane to obtain a higher number of imposed labels. Generation one 7250 IXR platforms recycle packets through the forwarding plane to yield a higher MSD, a process referred to as *reflow*. Generation two IXR platforms do not support or require reflow. *Table 6-1* lists the supported label stack depths across both generations of the IXR at both ingress and egress.

Table 6-1: 7250 IXR Supported Label Stack Depths

Role	**Gen One Platforms without Reflow**	**Gen 1 Platforms with Reflow**	**Generation Two Platforms**
Ingress LSR	Push up to 4 labels + 2 entropy labels, of which up to 3 can be transport labels	Push up to 8 labels + 2 entropy labels, of which up to 7 can be transport labels	Push up to 9 labels + 2 entropy labels, of which up to 8 can be transport labels
Egress LSR	Pop 2 transport labels + a service label, plus entropy labels where applicable		Pop 2 transport labels + a service label, plus entropy labels where applicable

For SR-TE, it should be clear that not all labels can be consumed by the source-routed SR-TE path – there are also other labels such as service labels, OAM labels, fast reroute backup labels, and entropy labels that form part of the label stack. If the imposed transport labels leave insufficient space in the routers MSD for other necessary labels, this may result in packet drops at egress. To avoid this SR-OS imposes a limit on the number of packets that can be consumed by transport labels (labels that form part of the SR source-routed path). Each SR-TE LSP, whether distributed or centralised, is configured with the parameter **max-sr-labels** that consists of **label-stack-size** ***<value>*** and **additional-frr-labels** ***<value>*** that that together define the overall maximum transport label stack size. This maximum transport label stack size needs to allow enough headroom for imposition of any other additional labels that might be in use. Some of these additional labels need only be accounted for if an optional particular feature is configured, such as hash label, entropy label (EL plus ELI), and control word. Conversely, some additional labels are always accounted for by the system, and these vary between FP-based platforms and the 7250 IXR because of the way that the underlying hardware constructs the egress label stack. In addition to labels automatically accounted for by the system, some layer 2 services also offer fewer transport labels in order to allow for the insertion of an inner Ethernet header. *Table 6-2* lists the more widely-deployed services together with the maximum available transport labels for FP-based platforms, together with the labels that are automatically accounted for by the system by default. Any optional labels that may be used in addition to the system accounted labels need to be subtracted from the maximum available transport labels. The values in *Table 6-2* should not be exceeded. If a service is bound to an SR-TE LSP whose label stack exceeds these values, BGP routes with next-hops bound to that LSP will not be resolved.

Table 6-2: Available Transport Labels for FP-based Platforms

Service	Maximum Available Transport Labels	System Accounted Labels
IP-VPN (VPRN)	10	Service label, OAM label
EVPN-IFL (VPRN)	10	Service label, OAM label
EVPN (VPLS or VPWS)	7	Service label, control word, ESI label
EVPN-IFF (R-VPLS)	7	Service label, control word
Spoke-SDP Epipe	8	Service label, OAM label
Spoke-SDP VPLS	8	Service label, OAM label

In some instances the labels that the system automatically accounts for by default can be unnecessarily constraining. For example, an EVPN service that does not use control word or ESI label may require use of an SR-TE LSP or Policy that contains more than seven hops, but it is not possible. In this instance it is possible to instruct the system not to automatically account for optional labels such as control word or ESI label using the **dynamic-egress-label-limit** command, which is configurable on a per-service basis. When set to **false**, (default) the system will automatically account for the labels listed in *Table 6-2*. When set to **true**, the system only accounts for service label, and accounts for other optional labels only if applicable (configured). This means that a VPRN service could use an SR-TE LSP or SR Policy consisting of up to 11 transport labels, and an EVPN-VPLS or VPWS could use an SR-TE LSP or SR Policy consisting of up to 9 transport labels.

Output 6-1: Disabling Dynamic Label Stack Allocation

```
    service {
       vpls "one" {
            bgp-evpn {
                mpls 1 {
                    dynamic-egress-label-limit true
                }
            }
        }
    }
```

Calculating the maximum available transport labels for the same services on the IXR platform is a rather simpler process and doesn't warrant listing in a table. Whilst considering the potential implications of reflow with generation one platforms outlined in *Table 6-1*, they can be succinctly summarised as follows:

- The system always accounts for the service label.
- The IXR platform constructs OAM packets in the CPM and injects them directly to the relevant port. Unlike FP-based platforms, these packets are therefore not included in any maximum label stack depth in the egress datapath.
- Generation one IXR platforms support a maximum of seven transport labels.
- Generation two IXR platforms support a maximum of eight transport labels.

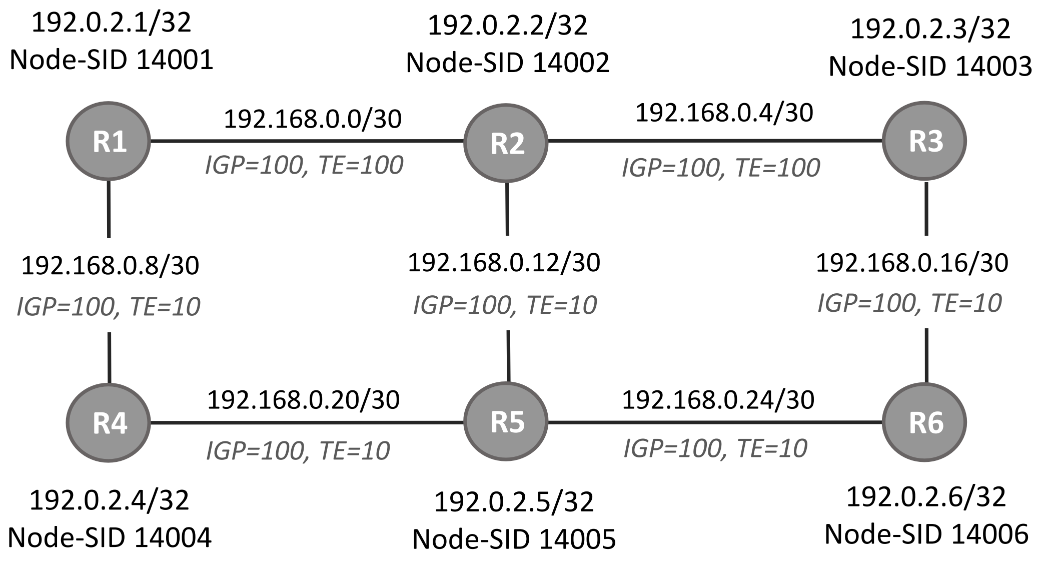
Again, any optional labels such as control word that may be used in addition to the service label need to be subtracted from the maximum available transport labels.

Distributed SR-TE

When using distributed SR-TE the headend is responsible for computing the source-routed path to the destination. It then imposes the relevant segments onto the packet to represent the computed path, where those segments are MPLS labels when SR-MPLS is in use. SR-OS supports two methods for a node to establish an SR-TE LSP. The first is using a local CSPF, and the second is using an explicitly defined path consisting of one or more IP addresses or explicit SID values.

Throughout this section the test topology shown in *Figure 6-1* is used for the purpose of illustration. System IP addresses and Node-SIDs are contained within the figure. The topology uses non-hierarchical IS-IS level 2 unless otherwise stated. SR is enabled with an SRGB of 12000-19999, and Node-SIDs are shown in the figure. All links have an IGP metric of 100, and a TE metric of 10 with the exception of links R1-R2 and R2-R3 which have a TE metric of 100.

Figure 6-1: SR-TE Test Topology



Local CSPF

When a local CSPF is used to compute SR-TE LSP path placement, the traffic engineering database is consulted and by default the system will return an explicit path consisting entirely of Adj-SIDs. Both IS-IS and OSPF can be used, but since there is a reliance on the availability of traffic engineering information the use of local CSPF is restricted to a single area or level. Conventional constraints like admin-group, hop-count, IGP metric and TE-metric can be used in the path calculation, but SR-TE LSPs do not have the ability to reserve bandwidth across a path. The bandwidth parameter may be configured on an SR-TE LSP, but it does not affect a local CSPF path calculation. (With centralised SR-TE however, the bandwidth parameter will be passed to a PCE if configured such that the PCE can reserve the signalled bandwidth in its topology graph.)

Output 6-2 shows the configuration of an SR-TE LSP with local CSPF path computation at R1 for an LSP to R3 (192.0.2.3). Initially an MPLS path is configured, which may or may not include strict or loose hops for the system to include in its path computation. The path “local-cspf” contains no hops and is put into an **admin-state** of **enable**, which simply allows the node to compute the entire path dynamically with no additional ‘*include*’ hops. The LSP “R1-to-R3” has a **type** of **p2p-sr-te** to distinguish it from the default point-to-point RSVP LSP and uses the **path-computation-method** command to select **local-cspf**. The command **local-sr-protection** allows the user to select the required protection needed for adjacencies used in the path computation. The output shows a setting of **mandatory** for this parameter, meaning that all adjacencies in the calculated path need to have a protected SID for each link otherwise the CSPF will fail to compute a path. Since TI-LFA is enabled throughout in the test topology, all adjacencies are automatically signalled as offering protection (the Backup flag is set in all advertised Adj-SIDs). The **preferred** option for this parameter means the system will prefer a protected adjacency over an unprotected adjacency if both exist or will combine both types of adjacencies. The final option is **none**, meaning the system will calculate the path using only unprotected adjacencies (excluding protected adjacencies) and will fail to compute the CSPF if it cannot do so. The **max-sr-labels** context shows a **label-stack-size** value of 5 and an **additional-frr-labels** value of 2 for an overall transport label stack size of 7. The 2 additional fast-reroute labels allow for the use of TI-LFA with a remote P-node with an adjacent Q-node.

SR-TE LSPs that are computed using a local CSPF use a timer-based re-optimisation. The **sr-te-resignal** context contains two system-wide parameters. The **resignal-timer** defines the frequency of the timer-based re-optimisation in minutes. When this timer expires the system will compute a new path using the specified (IGP/TE) metric, and also validate the current labels and hops. If the newly-computed path is considered better than the current path, or one of the current labels/hops is invalidated, the new path is programmed. It is important to note that the system will not immediately react to a failure on the path of the SR-TE LSP that was identified by executing a conventional SPF. That is, a failure on the path of the SR-TE LSP could be identified by the withdrawal of a link or node from the link-state database, but the system will not immediately react to that failure and will wait for the **resignal-timer** to expire before attempting to reoptimise the LSP onto a new path. Reacting to changes in the link-state database is the purpose of the **resignal-on-igp-event** parameter, and this is set to **true**. If there is a failure of a link or node on the path an SR-TE LSP with a path computation method of local CSPF the system will immediately resignal it.

Output 6-2: Configuration for SR-TE LSP with local CSPF

```
    mpls {
        path "local-cspf" {
            admin-state enable
        }
        lsp "R1-to-R3" {
            admin-state enable
            type p2p-sr-te
            to 192.0.2.3
            path-computation-method local-cspf
            local-sr-protection mandatory
            max-sr-labels {
                label-stack-size 5
                additional-frr-labels 2
            }
            primary "local-cspf" {
            }
        }
        sr-te-resignal {
            resignal-timer 30
            resignal-on-igp-event true
        }
    }
```

Output 6-3 shows the path details for the SR-TE LSP. As can be seen it has an admin state of up and an operational state of up. The output provides useful operational data on the state of the LSP and importantly shows the actual hops that the LSP is taking using the default IGP metric. In this case, the LSP is routed through R2 with Adj-SID 524286 and R3 with Adj-SID 524285. The first displayed Adj-SID to R2 (524286) will never actually appear in the data-plane because it is R1's Adj-SIDs for the link R1-R2 and therefore represents a label-switch to implicit-null. The key point to note here however is that by default when using local CSPF the system will always return a path using explicit Adj-SIDs.

Output 6-3: Path Details for SR-TE LSP "R1-to-R3"

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R3 path detail

===============================================================================
MPLS SR-TE LSP R1-to-R3
Path  (Detail)
===============================================================================
Legend :
    S      - Strict                     L      - Loose
    A-SID  - Adjacency SID              N-SID  - Node SID
    +      - Inherited
===============================================================================
-------------------------------------------------------------------------------
LSP SR-TE R1-to-R3
Path  local-cspf
-------------------------------------------------------------------------------
LSP Name    : R1-to-R3
Path LSP ID      : 57856
From             : 192.0.2.1
To               : 192.0.2.3
Admin State      : Up                     Oper State        : Up
Path Name   : local-cspf
Path Type        : Primary
Path Admin       : Up                     Path Oper         : Up
Path Up Time     : 0d 00:01:44            Path Down Time    : 0d 00:00:00
Retry Limit      : 0                      Retry Timer       : 30 sec
Retry Attempt    : 0                      Next Retry In     : 0 sec

PathCompMethod   : local-cspf             OperPathCompMethod: local-cspf
MetricType       : igp                    Oper MetricType   : igp
LocalSrProt      : mandatory              Oper LocalSrProt  : mandatory
LabelStackRed    : Disabled               Oper LabelStackRed: N/A

Bandwidth        : No Reservation         Oper Bandwidth    : 0 Mbps
Hop Limit        : 255                    Oper HopLimit     : 255
Setup Priority   : 7                      Oper SetupPriority: 7
Hold Priority    : 0                      Oper HoldPriority : 0
Inter-area       : N/A

PCE Updt ID      : 0                      PCE Updt State    : None
PCE Upd Fail Code: noError

PCE Report       : Disabled+              Oper PCE Report   : Disabled
PCE Control      : Disabled               Oper PCE Control  : Disabled

Include Groups   :                        Oper IncludeGroups:
None                                          None
Exclude Groups   :                        Oper ExcludeGroups:
None                                          None
Last Resignal    : n/a

IGP/TE Metric    : 200                    Oper Metric       : 200
Oper MTU         : 9178                   Path Trans        : 1
Degraded         : False
Failure Code     : noError
Failure Node     : n/a
Explicit Hops    :
    No Hops Specified
Actual Hops      :
    192.168.0.2(192.0.2.2)(A-SID)               Record Label        : 524286
 -> 192.168.0.6(192.0.2.3)(A-SID)               Record Label        : 524285

BFD Configuration and State
Template         : None                   Ping Interval     : N/A
Enable           : False                  State             : notApplicable
```

```
WaitForUpTimer    : 4 sec                     OperWaitForUpTimer: 0 sec
WaitForUpTmLeft   : 0
StartFail Rsn     : N/A
```

To demonstrate the local CSPF capability the metric used to calculate the LSP is changed from IGP metric to instead use TE metric as a constraint.

Output 6-4: R1-to-R3 LSP with TE-Metric

```
            lsp "R1-to-R3" {
                metric-type te
                }
            }
```

As a result of the change to use of TE metric it can now be observed that the path of the SR-TE LSP has been changed and that the path now routes through R1-R4-R5-R6-R3 based on best TE metric shown in *Figure 6-1*. Again, the path returned uses explicit Adj-SIDs throughout the entirety of the path.

Output 6-5: Modified LSP Path Using TE Metric

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R3 path detail | match "Actual Hops" post-
lines 5
Actual Hops      :
    192.168.0.10(192.0.2.4)(A-SID)              Record Label        : 524287
 -> 192.168.0.22(192.0.2.5)(A-SID)              Record Label        : 524286
 -> 192.168.0.26(192.0.2.6)(A-SID)              Record Label        : 524282
 -> 192.168.0.17(192.0.2.3)(A-SID)              Record Label        : 524286
```

SR-TE LSPs are installed in the tunnel-table with a default preference of 8, which is configurable and is discussed further in chapter 8. When using IGP metric as a constraint the metric reflected in the tunnel-table reflects the actual underlying IGP metric. However, when using TE-metric as a constraint the metric reflected in the tunnel-table is the maximum metric. If there are multiple SR-TE LSPs to the same destination endpoint and metric becomes the deciding factor this LSP will clearly not be selected for use. However, the metric of the SR-TE LSP can be overridden by configuration simply by using the **metric** command at LSP level.

Output 6-6: Tunnel-Table for R1-to-R3 LSP With TE Metric

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol sr-te

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-te     MPLS  655366    8      192.168.0.10   16777215
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

Resignal-on-IGP-event

At the headend R1 recall that the SR-TE LSP "R1-to-R3" is configured with **sr-te-resignal resignal-on-igp-event** in addition to a timer-based reoptimisation. The SR-TE LSP "R1-to-R3" is currently routed along the path R1-R4-R5-R6-R3 using TE metric and has an LSP ID of 57856. To demonstrate what happens when there is a topology change that impacts an SR-TE LSP I will fail the link R5 to R6.

To provide the reader with some dialogue of events during that event the following debug is used, although the subsequent CLI output is heavily truncated to show only the pertinent events. Note that the debug on the route-table prefix 192.168.0.24/30 is purely to provide a timestamp of when the R5 to R6 link is removed from the route-table.

Output 6-7: R1 Debug for SR-TE Resignal

```
*A:R1# /show debug
debug
    router "Base"
        ip
            route-table 192.168.0.24/30
        exit
        mpls lsp "R1-to-R3"
            event
                lsp-setup
                te
            exit
        exit
    exit
exit
```

The truncated debug output is shown in *Debug 6-1* and starts when the link R5 to R6 prefix is removed from the route-table. This triggers a CSPF request with a new LSP ID (57870), which is subsequently returned. The new path is then added, which allows the headend to initiate a make-before-break (MBB). Of course, one could argue that this is not strictly an MBB given that the SR-TE LSP is already down, but from the perspective of the headend the primary path is still up with LSP ID 57856 and a new SR-TE LSP with LSP ID 57870 has now been created. Finally, any state relating to LSP ID 57856 cleaned up and removed.

Debug 6-1: Debug of Events at R1 During IGP Triggered Resignal

```
1 2021/10/20 13:12:36.330 BST minor: DEBUG #2001 Base PIP
PIP: ROUTE
instance 1 (Base), RTM ADD event
   New Route Info
      prefix: 192.168.0.24/30 (0x19da01610)  preference: 18   metric: 400  backup metric: 0
owner: ISIS ownerId: 0
      1 ecmp hops  0 backup hops:
         hop 0:  192.168.0.2 @ if 2, weight 0

4 2021/10/20 13:12:40.001 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Create CSPF request for LspPath R1-to-R3::local-cspf(LspId 57870)

94 2021/10/20 13:12:40.002 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
CSPF Path returned by TE for LspPath R1-to-R3::local-cspf(LspId 57870)
From 192.0.2.1  To 192.0.2.3
Hop 1 -> Label 524287 IpAddr 192.168.0.10 LinkId 0 UpstreamIpAddr 192.168.0.9 UpstreamLinkId 0
RtrAddr 192.0.2.4 Strict
```

```
Hop 2 -> Label 524286 IpAddr 192.168.0.22 LinkId 0 UpstreamIpAddr 192.168.0.21 UpstreamLinkId
0 RtrAddr 192.0.2.5 Strict
Hop 3 -> Label 524284 IpAddr 192.168.0.13 LinkId 0 UpstreamIpAddr 192.168.0.14 UpstreamLinkId
0 RtrAddr 192.0.2.2 Strict
Hop 4 -> Label 524285 IpAddr 192.168.0.6 LinkId 0 UpstreamIpAddr 192.168.0.5 UpstreamLinkId 0
RtrAddr 192.0.2.3 Strict

96 2021/10/20 13:12:40.002 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Add path R1-to-R3::local-cspf(LspId 57870) SR-TE Tunnel parameters for LSP R1-to-R3, TunnelId
655363

101 2021/10/20 13:12:40.141 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
MBB successful for active Primary LspPath R1-to-R3::local-cspf(LspId 57870) for LSP R1-to-R3

103 2021/10/20 13:12:40.141 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Reprogram active path R1-to-R3::local-cspf(LspId 57870) for LSP R1-to-R3

107 2021/10/20 13:12:40.241 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Cleanup old LspPath R1-to-R3::local-cspf(LspId 57856) after successful MBB

108 2021/10/20 13:12:40.241 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Delete path R1-to-R3::local-cspf(LspId 57856) SR-TE Tunnel parameters for LSP R1-to-R3, TunnelId
655363

109 2021/10/20 13:12:40.241 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
SR-TE tunnel has been updated for LspPath R1-to-R3::local-cspf(LspId 57870)
```

Output 6-8 shows the actual hops of the newly calculated path for the R1-to-R3 LSP after the R5-R6 link failure. As can be seen the LSP now takes the path R1-R4-R5-R2-R6-R3 to avoid the R5-R6 link failure but still uses TE metric as a constraint.

Output 6-8: R1-to-R3 LSP Path After R5-R6 Link Failure

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Actual Hops" post-
lines 5
Actual Hops      :
    192.168.0.10(192.0.2.4)(A-SID)              Record Label        : 524287
 -> 192.168.0.22(192.0.2.5)(A-SID)              Record Label        : 524286
 -> 192.168.0.13(192.0.2.2)(A-SID)              Record Label        : 524284
 -> 192.168.0.6(192.0.2.3)(A-SID)               Record Label        : 524285
```

The link R5 to R6 is restored. By default, the headend will wait until expiry of the next resignal-timer before optimising the SR-TE LSP back to most optimal path, but this can be triggered using the following command.

Output 6-9: Manual SR-TE LSP Reoptimisation

```
A:R1# tools perform router mpls resignal sr-te-lsp "R1-to-R3" path "local-cspf"
```

After the reoptimisation command is issued the R1-to-R3 LSP is reverted back to the most optimal path based on TE metric through R1-R4-R5-R6-R3.

Output 6-10: R1-to-R3 LSP Path After Reoptimisation

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R3 path detail | match "Actual Hops" post-
lines 5
Actual Hops      :
    192.168.0.10(192.0.2.4)(A-SID)              Record Label        : 524287
```

```
 -> 192.168.0.22(192.0.2.5)(A-SID)              Record Label        : 524286
 -> 192.168.0.26(192.0.2.6)(A-SID)              Record Label        : 524282
 -> 192.168.0.17(192.0.2.3)(A-SID)              Record Label        : 524286
```

Note: When **sr-te-resignal resignal-on-igp-event** is used the system will attempt to find another path if one exists. It will not however bring the path down if an alternative path cannot be found.

Label Stack Reduction

So far, this section has illustrated the default use of local CSPF which calculates each hop as a strict hop and returns explicit Adj-SIDs for those hops. This approach may result in the calculation of a path where the hop count/label stack exceeds the MSD of the headend router. SR-OS therefore also provides the ability to configure an SR-TE LSP using local CSPF with the command **label-stack-reduction**. When this command is enabled, the CSPF will attempt to replace two or more Adj-SIDs (or Adj-Set SIDs) in the path with a Node-SID whilst ensuring that all ECMP paths to that Node-SID meet the configured LSP constraints. In turn the use of Node-SIDs also provides the additional benefit of load-balancing traffic over any ECMP paths.

To implement label stack reduction SR-OS splits the CSPF into a number of phases. In brief, the first phase calculates one or more explicit paths using Adj-SIDs, while the second phase looks at the CSPF tree for each of the paths returned from phase one and attempts to find the furthest Node-SID in a path segment that can summarise the Adj-SIDs before it.

SR-OS will only attempt label stack reduction when the **local-sr-protection** command is set to **preferred**. This is simply because when explicit hops are used in conjunction with Adj-SIDs the protection state of each Adj-SID (and therefore each link) is advertised into the IGP (using the Backup Flag) and is therefore known throughout the domain. This is not true for Node-SIDs as they have no concept of a Backup Flag and provide no indication as to whether they are protected or not. As a result, the headend calculating the path including Node-SIDs cannot conform to the protection requirement.

In the rather simple topology of *Figure 6-1*, it is difficult to yield a significant result in label stack reduction. However, to demonstrate the use of label stack reduction an SR-TE LSP from R1 to R4 is established that must include the hop R3. This is enforced through the use of a single loose hop to R3 (192.0.2.3) in the path “local-cspf”, which is used by the LSP. Although this is a nonsensical path it serves to illustrate the use of label stack reduction by creating a path with a number of hops. The LSP uses the IGP metric and initially does not include the label stack reduction feature.

Output 6-11: SR-TE LSP Configuration for Label Stack Reduction

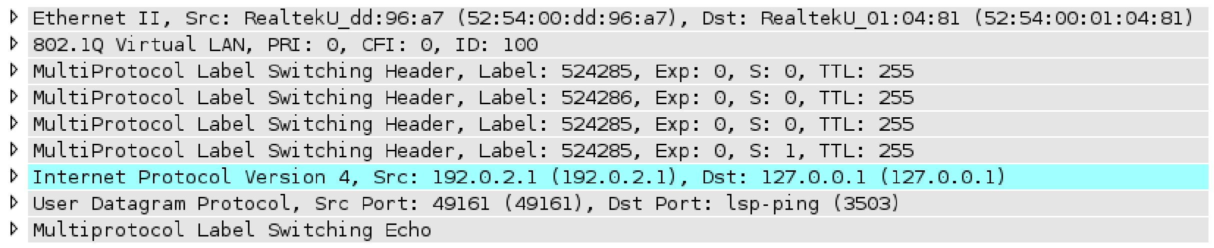
```
        mpls {
            path "local-cspf" {
                admin-state enable
                hop 1 {
                    ip-address 192.0.2.3
                    type loose
                }
            }
            lsp "R1-to-R4" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.4
                path-computation-method local-cspf
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "local-cspf" {
                }
            }
        }
```

With the configuration of *Output 6-11* applied, the headend R1 returns an explicit path containing Adj-SIDs that routes along the path R1-R2-R3-R6-R5-R4. Note that although there are five Adj-SIDs shown in the output only four will be present in the data-plane as the first displayed Adj-SID to R2 will not be imposed in the stack. This is because it is R1's Adj-SID for the R1-R2 link and represents a label-switch to implicit null. This is confirmed in the packet trace shown in *Figure 6-2* taken on the link R1 to R2.

Output 6-12: Path for LSP R1-to-R4 Without Label Stack Reduction

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R4 path detail | match "Actual Hops" post-
lines 6
Actual Hops      :
    192.168.0.2(192.0.2.2)(A-SID)                Record Label        : 524286
 -> 192.168.0.6(192.0.2.3)(A-SID)                Record Label        : 524285
 -> 192.168.0.18(192.0.2.6)(A-SID)               Record Label        : 524286
 -> 192.168.0.25(192.0.2.5)(A-SID)               Record Label        : 524285
 -> 192.168.0.21(192.0.2.4)(A-SID)               Record Label        : 524285
```

Figure 6-2: Packet Trace for LSP R1-to-R4 With Adj-SIDs



```
▷ Ethernet II, Src: RealtekU_dd:96:a7 (52:54:00:dd:96:a7), Dst: RealtekU_01:04:81 (52:54:00:01:04:81)
▷ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 100
▷ MultiProtocol Label Switching Header, Label: 524285, Exp: 0, S: 0, TTL: 255
▷ MultiProtocol Label Switching Header, Label: 524286, Exp: 0, S: 0, TTL: 255
▷ MultiProtocol Label Switching Header, Label: 524285, Exp: 0, S: 0, TTL: 255
▷ MultiProtocol Label Switching Header, Label: 524285, Exp: 0, S: 1, TTL: 255
▷ Internet Protocol Version 4, Src: 192.0.2.1 (192.0.2.1), Dst: 127.0.0.1 (127.0.0.1)
▷ User Datagram Protocol, Src Port: 49161 (49161), Dst Port: lsp-ping (3503)
▷ Multiprotocol Label Switching Echo
```

The label stack reduction feature is subsequently enabled on the R1-to-R4 LSP.

Output 6-13: Enabling Label-Stack-Reduction

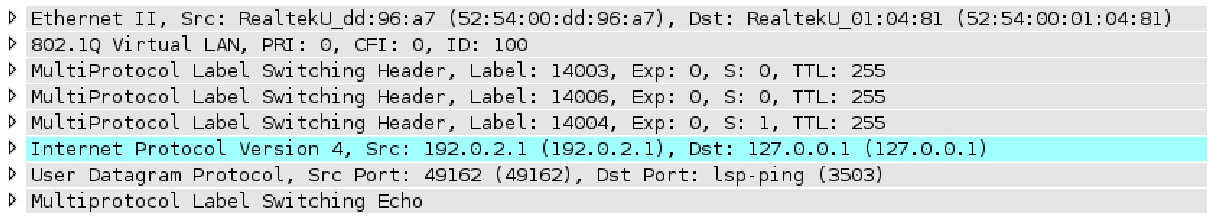
```
        mpls {
            lsp "R1-to-R4" {
                label-stack-reduction true
            }
        }
```

The path is subsequently recalculated by R1 and the new path is shown in *Output 6-14*. The path consists of Node-SIDs for R3, R6, and R4. Hence the path taken by the LSP is exactly the same as the path taken when using an explicit path with Adj-SIDs, but the number of labels imposed is reduced from four to three. Larger and more complex topologies will obviously benefit from a higher level of label stack reduction. For completeness an equivalent packet trace in *Figure 6-3* shows the imposed labels on the link R1 to R2.

Output 6-14: Path for LSP R1-to-R4 With Label Stack Reduction

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R4 path detail | match "Actual Hops" post-
lines 4
Actual Hops      :
    192.0.2.3(192.0.2.3)(N-SID)                  Record Label        : 14003
 -> 192.0.2.6(192.0.2.6)(N-SID)                  Record Label        : 14006
 -> 192.0.2.4(192.0.2.4)(N-SID)                  Record Label        : 14004
```

Figure 6-3: Packet Trace for LSP R1-to-R4 with Node-SIDs



```
▷ Ethernet II, Src: RealtekU_dd:96:a7 (52:54:00:dd:96:a7), Dst: RealtekU_01:04:81 (52:54:00:01:04:81)
▷ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 100
▷ MultiProtocol Label Switching Header, Label: 14003, Exp: 0, S: 0, TTL: 255
▷ MultiProtocol Label Switching Header, Label: 14006, Exp: 0, S: 0, TTL: 255
▷ MultiProtocol Label Switching Header, Label: 14004, Exp: 0, S: 1, TTL: 255
▷ Internet Protocol Version 4, Src: 192.0.2.1 (192.0.2.1), Dst: 127.0.0.1 (127.0.0.1)
▷ User Datagram Protocol, Src Port: 49162 (49162), Dst Port: lsp-ping (3503)
▷ Multiprotocol Label Switching Echo
```

Using SR-TE LSPs with local CSPF can be an effective way to provide a traffic engineering capability, with support for IGP metric, TE metric, admin-groups, and SRLGs. For some it can also be the first step into SR-TE before moving towards a centralised controller. The main drawback of course is that it is constrained to use in a single level or area due to the dependency on TE information that is area-scoped.

Explicit Path Definition

A second approach with distributed SR-TE is to manually define the path that an LSP takes using the MPLS ‘path’ context in SR-OS to define strict or loose hops along the path to the destination. Each of these manually defined hops can take one of two forms:

i. An IP address. As an IP address cannot be used in an SR-MPLS label stack the system automatically converts the IP addresses into a label stack. This is referred to as the Hop-to-Label Translation and the system consults the TE database to extract the information to make this translation.
ii. A SID in the form of a label value. In this case no translation is required.

In the case of an IP address, the hop-to-label translation differs between strict hops and loose hops. A strict hop is always translated to an Adj-SID, whereas a loose hop is always translated to a Node-SID.

Once again using the topology from *Figure 6-1, Output 6-15* shows an example of an SR-TE LSP from R1 to R3 using the hop-to-label translation approach. The MPLS path “R1-to-R3-IPADDR” contains three hops to the system addresses of R4, R5, and R6 respectively. Hop 1 and 3 are strict hops, whilst hop 2 is a loose hop. Note that the path hops could have used any advertised interface address on the remote hops, but the example uses system addresses for the purpose of simplicity.

The LSP R1-to-R3 has a relatively straightforward configuration containing parameters that have already been explained. The notable difference between this example and the previous examples using local CSPF is that there is no **path-computation-method** command configured. This is the default configuration and sets the computation method to be the hop-to-label translation method.

Output 6-15: R1-to-R3 LSP With Hop-to-Label Translation

```
        mpls {
            path "R1-to-R3-IPADDR" {
                admin-state enable
                hop 1 {
                    ip-address 192.0.2.4
                    type strict
                }
                hop 2 {
                    ip-address 192.0.2.5
                    type loose
                }
                hop 3 {
                    ip-address 192.0.2.6
                    type strict
                }
            }
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "R1-to-R3-IPADDR" {
                }
            }
        }
```

Output 6-16 shows the path state and actual hops taken by the R1-to-R3 LSP. The admin and operational states show the LSP is up. The SIDs returned in the actual hops are reflective of the type of each defined hop in the MPLS path. Hop 1 (R4) and hop 3 (R6) are Adj-SIDs as they were defined as strict hops, whilst hop 2 (R5) is a Node-SID as it was defined as a loose hop. The final hop is the destination of the SR-TE LSP hence it will always be a Node-SID.

Output 6-16: R1 to R3 LSP State with Hop-to-Label Translation

```
*A:R1#  show  router  mpls  sr-te-lsp  "R1-to-R3"  path  detail  |  match  expression  "LSP
Name|Actual Hops" post-lines 5
LSP Name    : R1-to-R3
Path LSP ID      : 56328
From             : 192.0.2.1
To               : 192.0.2.3
Admin State      : Up                     Oper State        : Up
Path Name   : R1-to-R3-IPADDR
Actual Hops      :
    192.168.0.10(192.0.2.4)(A-SID)              Record Label        : 524287
 -> 192.0.2.5(192.0.2.5)(N-SID)                 Record Label        : 14005
 -> 192.168.0.26(192.0.2.6)(A-SID)              Record Label        : 524282
 -> 192.0.2.3(192.0.2.3)(N-SID)                 Record Label        : 14003
```

It’s also worth viewing the tunnel-table entry for the R1-to-R3 LSP. As this LSP uses a hop-to-label translation it does not run a full CSPF. As a result, the metric shown in the tunnel-table is the maximum metric. As the test topology runs IS-IS, this is 16,777,215.

Output 6-17: Tunnel-Table Entry for R1-to-R3 LSP

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol sr-te

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-te     MPLS  655366    8      192.168.0.10   16777215
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

When the hop to label translation method is used the system consults the TE database for the relevant SID/label for the defined IP addresses. Because the defined IP addresses may belong to any interface (loopback or other) of another router that may not have an associated SID, the system needs to be able to associate that prefix with a known SID belonging to the same router. To do this it uses the Router ID as a key. If the defined hop address is associated with a SID, then that SID will be used in the hop-to-label translation. If the defined hop address is not associated with a SID, then the system will use a Node-SID (loose hop) or Adj-SID (strict hop) belonging to the router owning the defined hop address. The result of this is simply that TE information must be available, which restricts the use of the IP address/Hop-to-Label Translation to a single level/area. It also enforces that **traffic-engineering** is enabled and that MPLS and RSVP are enabled throughout as recommended in Chapter 2.

The path of the LSP R1-to-R3 is subsequently changed to use SID hop definition in the MPLS path context as illustrated in *Output 6-18*. Each hop is defined as a SID label, and in this case uses the Node-SIDs for R4, R5, and R6 for hop 1, hop 2, and hop 3 respectively. (In effect the path takes the same route through the test topology as the hop-to-label translation example.) Although I’ve used Node-SIDs that are known in the IGP as hops, the defined SID labels can be any SID such as a Node-SID, Adj-SID, Anycast SID, Binding SID, as long as the first hop can be resolved by the headend as known in the IGP. It follows of course that as the SIDs are already defined, the SR label stack can be built, and no translation is required. Finally, it’s worth noting that unlike the IP address definition method, when using SID labels there is no concept of strict or loose. That is a characteristic of the SID itself.

Output 6-18: R1-to-R3 LSP With SID Hop Definition

```
        mpls {
            path "R1-to-R3-SID" {
                admin-state enable
                hop 1 {
                    sid-label 14004
                }
                hop 2 {
                    sid-label 14005
                }
                hop 3 {
                    sid-label 14006
                }
            }
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "R1-to-R3-SID" {
                }
            }
        }
```

The path state and actual hops taken by the R1-to-R3 LSP is shown in *Output 6-19*. The admin and operational states show the LSP is up. Unlike the hop-to-label translation approach the actual hops do not reflect any IP addresses, but rather just show a straightforward label stack.

Output 6-19: R1-to-R3 LSP State with SID Definition

```
*A:R1#  show  router  mpls  sr-te-lsp  "R1-to-R3"  path  detail  |  match  expression  "LSP
Name|Actual Hops" post-lines 5
LSP Name    : R1-to-R3
Path LSP ID      : 56330
From             : 192.0.2.1
To               : 192.0.2.3
Admin State      : Up                    Oper State        : Up
Path Name   : R1-to-R3-SID
Actual Hops      :
    n/a                                        Record Label        : 14004
 -> n/a                                        Record Label        : 14005
 -> n/a                                        Record Label        : 14006
```

Like the hop-to-label translation approach, the SID definition does not run a full CSPF and consequently the metric shown in the tunnel-table is the IGP maximum metric. One advantage that the SID definition approach has over the hop-to-label translation approach is that SR-TE LSPs may span multiple areas/levels/domains as long as each hop in the label stack can resolve the next SID in the label stack.

Centralised SR-TE

When using centralised SR-TE a path computation element (PCE) is responsible for computing the source-routed path to the destination. The use of a controller is not simply about offloading path computation processing or overcoming the limitations that distributed SR-TE has with calculating inter-domain TE paths. It provides other significant benefits such as:

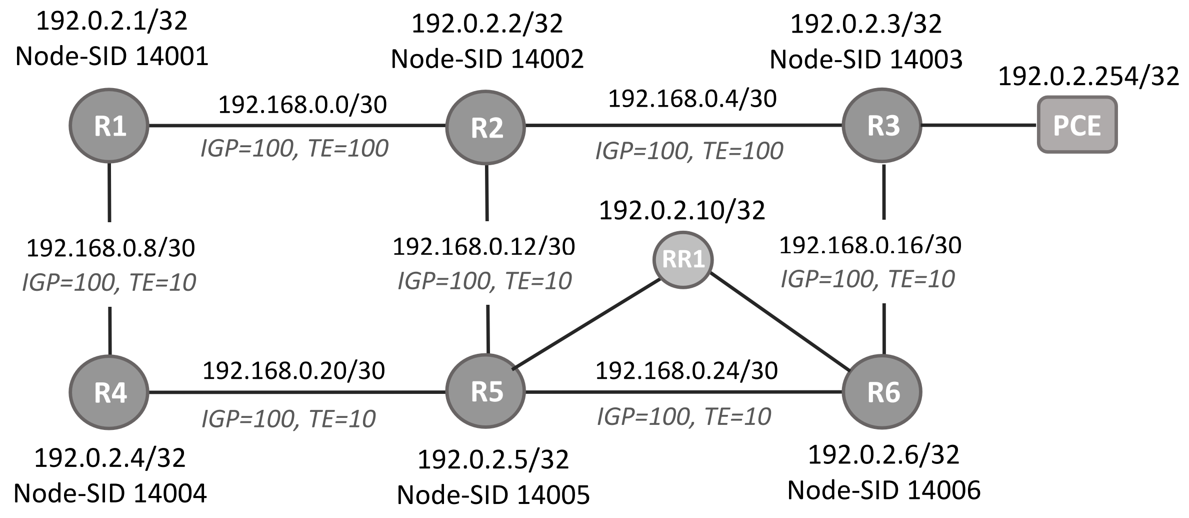
- The ability to enforce constraints on paths that originate/terminate on different network elements, thereby providing path diversity and/or bidirectionality and/or disjointness.

- It can help to avoid collisions, re-tries, and packing problems that have been observed in networks when using distributed TE path calculation, where head-ends make autonomous decisions.
- It can also take a global view of path placement strategies, including the ability to make path placement decisions over a large number of LSPs concurrently as opposed to considering each LSP independently. This allows for 'global' optimisation of network resources.
- It provides an operator with the ability to invoke new traffic engineering rules and behaviour without having to upgrade a single network element. This can simply be enforced by the application of the relevant policy.
- A PCE also allows the notion of bandwidth reservation for SR-TE LSPs. Although bandwidth is not reserved in transit nodes as RSVP-TE does, it is possible for an SR-TE LSP headend to signal a bandwidth requirement to a PCE and for the PCE to subsequently account for that bandwidth in any path computations (it would also need to have a knowledge of other bandwidth being consumed within the network). Alternatively, the PCE can ingest telemetry data of SR-TE LSP statistics and use that as an indication of real-time bandwidth consumption.

This chapter describes the use of Path Computation Element Protocol (PCEP) in SR-OS to instantiate and manage SR-TE LSPs, and in this scenario the router performs the role of a path computation client (PCC). It starts with a brief overview of PCEP and how to enable it on the PCC, and then goes on to describe how to enable BGP Link State (BGP-LS) to propagate network topology information northbound towards the PCE. Following this, the chapter describes the various approaches that a PCC can use to interact with a PCE when initiating the SR-TE LSP, including a model that allows the PCE to initiate an LSP and push it to the PCC.

Throughout this section the test topology shown in *Figure 6-4* is used for the purpose of illustration. System IP addresses and Node-SIDs are contained within the figure. The topology uses non-hierarchical IS-IS level 2 unless otherwise stated. SR is enabled, and the SRGB in use is 12000-19999. All links have an IGP metric of 100, and a TE metric of 10 with the exception of links R1-R2 and R2-R3 which have a TE metric of 100. A PCE is introduced and connected to R3. Static routing exists between R3 and the PCE, which R3 redistributed into IS-IS to ensure reachability between all network elements and the PCE. A route-reflector (RR1) is also introduced which participates in the IS-IS domain. The route-reflector (RR) is intended to be a control-plane-only function hence it does not advertise any SR capability. The introduction of the RR is purely to simplify BGP peering relationships for the advertisement of link-state information towards the PCE. It could be argued that it is not required in this very simple topology but the use of one or more RRs is a more likely deployment model.

Figure 6-4: Network Topology for Centralised SR-TE



Control Plane Requirements for Centralised TE

Before discussing how a PCC may interact with the PCE to establish and maintain SR-TE LSPs some foundations are necessary. BGP-LS is required between one or more network elements in each routing domain and the PCE such that the latter can learn the network topology. Equally, PCEP sessions need to be established before any message exchange can take place. The following sub-sections provide an overview of both, and how they are configured and enabled in SR-OS.

The PCE Communication Protocol (PCEP)

PCEP operates over TCP to provide transport layer reliability and uses the well-known port 4189. When the TCP connection is established the PCC and PCE initiate session establishment during which various session parameters are exchanged in an Open message exchange. The session parameters include timers such as the Keepalive timer and DeadTimer as well as the capabilities that each of the peers is able to support. The base PCEP specification [19] defines a number of messages which are briefly described as follows:

- Open and Keepalive messages area used to initiate and maintain a PCEP session.
- Path Computation Request (*PCReq*): A PCEP message sent by a PCC to a PCE to request a path computation.
- Path Computation Reply (*PCRep*): A PCEP message sent by a PCE to a PCC in response to a PCReq message. A PCRep messge can contain either a computed path if the request was successful, or it can contain a negative reply if the request was unsuccessful.
- Path Computation Notification (*PCNtf*): A PCEP notification message either sent by a PCC to a PCE or sent by a PCC to a PCC to notify of a specific event.

- Path Computation Error (*PCErr*): A PCEP message sent upon the occurrence of a protocol error condition.
- Close message: A message used to close the PCEP session.

PCEP messages consist of a common header followed by a number of objects, some of which are mandatory and some of which are optional. Each object within a message consists of a common object header containing an object class, object type, and a flags field. The flags field contains a Processing-Rule-flag (P-flag) that is used by the PCC in a PCReq message to indicate whether the object must be accounted for by the PCE during path computation or whether it is optional. It also has an Ignore-flag (I-flag) used by the PCE in a PCRep message to indicate to the PCC whether or not an optional object was processed. The common object header is followed by a number of variable-length TLVs. Some examples of PCEP objects include:

- The LSP object, which is used to specify details of the target LSP, the operation to be performed on the LSP, and the LSP delegation status.
- The Request Parameters (RP) object, which is carried within PCReq and PCRep messages to specify various characteristics of the path computation request. It was subsequently renamed the Stateful RP (SRP) object when using a stateful PCE.
- The Explicit Route Object (ERO), which is used to specify the path that a given LSP should take.
- The Metric object, which is used to indicate the cost of a path or to place a bound on the path cost during path computation. Metrics can be IGP, TE, or hop-count.
- The End-Points Object, which is used to specify the source and destination IP addresses of the path.

For each PCEP message type, rules are defined that specify the set of objects that each particular message type can carry.

A PCE can be stateful or stateless. A stateful PCE maintains strict synchronisation between the PCE and not only the networks state (in terms of topology and resource), but also the set of computed paths and reserved resources in the network. That is, the PCE utilises information from the TE database as well as information about existing TE LSPs in the network when processing new requests. Conversely, stateless PCEs do not attempt to remember any computed TE LSPs, and each new request is processed independently of any other. When the concept of a PCE was first introduced a number of years ago there was some concern that the control plane overhead and state maintenance associated with a stateful PCE might introduce scaling issues. Fast forward a decade or more during which time both CPU power and storage space increased exponentially, and those concerns have largely dissipated. The base PCEP specification was therefore extended in [20] to enable a PCC to communicate with a stateful active PCE. The term active referred to the fact that the PCE could make unsolicited changes to existing LSP states on a PCC. This extension introduced several notable new functions. It provided the capability for a PCC to delegate control of one or more LSPs to the PCE, after which the PCE becomes the sole owner of those LSPs attributes as long as the delegation is in force. However, the PCC retains the right to revoke the delegation at any time and take control of the LSP again. The stateful PCE extensions also specified a mechanism for LSP state synchronisation between the PCC and the PCE after a PCEP session is established. This allowed the PCE to learn the state of a PCC's LSPs before it subsequently performed any path computations or updates to LSPs in that PCC.

A new capability, the Stateful-PCE-Capability TLV, was introduced to allow a PCEP speaker to indicate its support for stateful PCE operation in the Open message exchange. The flags field of the Stateful-PCE-Capability TLV included an LSP Update flag (U-flag) to indicate support for unsolicited LSP parameter updates. In addition, two new messages were also added:

- Path Computation State Report (*PCRpt*): A PCEP message sent by a PCC to a PCE to report the status of one or more LSPs such as the LSPs actual path, bandwidth, and operational status. The PCRpt message is also used in delegation or revocation of control of an LSP to/from a PCE.
- Path Computation Update Request (*PCUpd*): A PCEP message sent by a PCE to a PCC to update LSP parameters on one or more LSPs.

PCEP originally defined a method of communication between the PCC and PCE that was intended for use with RSVP-TE LSPs. As SR technology emerged it was again extended in [21] to encompass the setup and maintenance of SR-TE LSPs. The Explicit Route Object (ERO) object used to define the path that a given LSP should take was originally encoded using IPv4/IPv6 addresses or Autonomous System numbers. A new ERO sub-object, the SR-ERO, was therefore defined that is capable of carrying a SID together with the identity of the node/adjacency represented by the SID. The Record Route Object (RRO), which is used by the PCC to indicate the actual route used after a path has been setup was similarly extended to include SR-RRO sub-objects. Previous standardisation work [22] had already defined a Path-Setup-Type Capability TLV to be used in an Open message exchange, through which a PCEP speaker could indicate which path setup types (PSTs) it supports. The same work also defined a Path-Setup-Type TLV, encoded in a mandatory object (the RP object, or SRP object), that could be carried in PCReq and PCRep messages to distinguish the type of LSP being set up. The SR extensions enhanced both the Path Setup Type Capability TLV and the Path Setup Type TLV to include a new PST for SR.

All of the extensions to PCEP discussed so far have been relevant to SR-TE LSPs that were PCC-initiated. A PCE could make changes to the state of delegated SR-TE LSPs, but the initial creation of the LSP was PCC-initiated. As the industry moved into an era of Software-Defined Networking (SDN) the concept of PCE-initiated SR-TE LSPs was introduced. The motivation was to give the PCE the ability to trigger the dynamic creation and teardown of LSPs based on application demand.

To facilitate PCE-initiated LSPs the LSP Initiate Request (PCInit) message was introduced, which a PCE can send to a PCC to request the initiation or deletion of an LSP. In response the PCC would return a PCRpt message to notify the PCE of the state of the LSP and to delegate it. To distinguish between setup and teardown of a PCE-initiated LSP, two flags were introduced:

- The Create flag (C-flag) which is used in the LSP object of a PCRpt message to indicate that this LSP was created via a PCInit message. The intention is that the C-flag would remain set on each PCRpt message (not just during LSP setup) to allow PCEs other than the initiating PCE to be aware of which LSPs were PCE initiated.
- The LSP-Remove (R-flag) which is used in the SRP object of a PCInit message to indicate a request to remove an LSP.

Lastly, the capability to indicate support for PCE-initiated LSPs was needed. The Stateful-PCE-Capability TLV had already been introduced by the SR extensions to be used in the Open message exchange to indicate support for a stateful PCE. To indicate support for PCE-initiated LSPs a new flag was added to this TLV known as the LSP-Instantiation-Capability (I-flag).

PCEP continues to evolve, but this brief overview should provide sufficient background to allow a better understanding of the configuration and outputs that follow during the remainder of this section.

The first step is to enable PCEP on all of the PCCs, which in the topology includes routers R1 to R6. No assumptions are made about the make/vendor of the PCE, and SR-OS is agnostic to whatever PCE is used provided it uses standards-based PCEP. For that reason, the configuration of the PCE is not covered here. *Output 6-20* shows the configuration required to enable PCEP at R1. Within the **pcep** context generic SR-OS only provides a **pcc** option, and within that sub-context the PCEP peers and timers are configured. In my simple topology, only a single **peer** is configured at 192.0.2.254 and placed into an **admin-state** of **enable**. Up to two peers can be configured where redundant PCEs are in place, but only a single PCE is considered primary because an LSP can only be delegated and subsequently controlled by a single entity. Within each **peer** context, a **preference** parameter can be used to assign a preference value to each peer, where the peer with the higher numerical preference value is elected. PCEP also allows for either a PCE or PCC to signal an overload condition using a Notification object within a PCNtf message. If redundant PCEs exist and one is signalling overload condition, SR-OS will always select the other PCE. In the test topology the PCE connectivity is in-band, hence the **local-address** is set to the system address.

The **redelegation-timer** is the period of time a PCC waits for when a PCEP session is terminated before revoking LSPs delegated to the failed PCE and attempting to redelegate those LSPs to an alternate PCE. The redelegation timer applies to both PCC-initiated and PCE-initiated LSPs. The **state-timer timer** is applicable only to PCE-initiated LSPs and is the period of time the PCC waits for when a PCE session is terminated before flushing LSP state associated with that session. The **state-timer** has an associated **timer-action** with options of none or **remove**.

The liveliness of a PCEP session is maintained through keepalives. The PCC context has **keepalive** and **dead-timer** commands to set the values of both in seconds. The default **keepalive** timer is 30 seconds, while the default **dead-timer** is 120 seconds.

Output 6-20: PCEP Configuration

```
    router "Base" {
        pcep {
            pcc {
                admin-state enable
                local-address 192.0.2.1
                redelegation-timer 180
                state-timer {
                    timer 300
                    timer-action remove
                }
                peer 192.0.2.254 {
                    admin-state enable
                }
            }
        }
    }
```

Debug 6-2 shows the Open message exchange between R1 and the PCE, both of which predominantly carry the same information. The Open message contains a single object, namely the Open object, that initially negotiates keepalive timer, deadtime timer, PCEP version, and session identifier. There are then a number of capability TLVs, the salient parts of which are summarised. The presence of the Stateful-PCE-Capability TLV is sufficient to indicate support for stateful PCE, while flags within that TLV indicate support for LSP updates and PCE-initiated LSPs. An SR-PCE-Capability TLV is used to exchange SR capabilities and includes an MSD field to allow the PCC to notify the PCE of its maximum supported label depth. Draft versions of [21] defined the SR-PCE-Capability TLV as a top-level TLV of the Open object, however, as [21] moved towards standards track, the use of the SR-PCE-Capability TLV as a top-level TLV to indicate support for SR was deprecated. Instead, a Path Setup Type Capability TLV [22] was introduced to indicate which path setup types (PSTs) a PCEP speaker supports. When the Path Setup Type Capability TLV indicates support for SR as a supported PST, it should carry the SR-PCE-Capability as a sub-TLV rather than a top level TLV. In order to maintain backward compatibility with both implementations, SR-OS encodes the SR-PCE-Capability TLV as both a top-level TLV and as a sub-TLV of the Path Setup Type Capability TLV. A PCEP speaker receiving both TLVs implements the supported TLV and should ignore the other one.

Note: To advertise base platform capabilities, FP-based platforms signal an MSD of 11 in the Open object SR-PCE-Capability TLV while the 7250 IXR signals an MSD of 7. The MSD is also signalled on a per-LSP basis using an MSD Metric object within a PCReq message, with the MSD value equivalent to the value of the configured **max-sr-labels label-stack-size**. This MSD Metric object has the B (bound) bit set to one to notify the PCE that returned paths should not exceed this value.

In addition to the Stateful-PCE-Capability TLV and SR-PCE-Capability TLV, there is also a Point-to-Multipoint Capability TLV to indicate support for the same, and lastly a Speaker Entity ID TLV. The speaker entity ID is a unique identifier for the node that does not change during the lifetime of the PCEP speaker and identifies the PCEP speaker to its peers even if the speakers IP address is changed.

Debug 6-2: PCEP Open Message Exchange

```
5 2022/10/28 10:19:19.271 BST minor: DEBUG #2001 Base PCC
PCC: TX-Msg: OPEN 000 19:20:41.570
Peer 192.0.2.254
  Open Obj:
    {Version 1 SessionId 132 KeepAlive 30 DeadTime 120}
    {Stateful  1  LspUpdate  1  dbVersion  0  pathProfile  1  triggeredSync  0  deltaSync  0
pceInitiated 1 SegRt 1 pst-cap { rsvp seg-rt}}
    {P2MP 1 p2mpUpdate 1 p2mpInitiate 1 triggeredInitialSync 0 Association 1 Multipath 1
}
    {speakerId 00:01:ff:00:00:00 len 16}


6 2022/10/28 10:19:19.274 BST minor: DEBUG #2001 Base PCC
PCC: RX-Msg: OPEN 000 19:20:41.570
Peer 192.0.2.254
  Open Obj:
    {Version 1 SessionId 1 KeepAlive 30 DeadTime 120}
    {Stateful  1  LspUpdate  1  dbVersion  0  pathProfile  1  triggeredSync  0  deltaSync  0
pceInitiated 1 SegRt 1 <NULL>}
    {P2MP 1 p2mpUpdate 1 p2mpInitiate 1 triggeredInitialSync 0 Association 1 Multipath 1
}
    {speakerId 00:29:ff:00:00:00 len 16}
```

Output 6-21 shows the details of R1's peering session to the PCE just after the session is established. There are no LSPs configured at R1 but note that there has already been a PCRpt message sent by R1. When a PCEP session is initially established, the PCC synchronises the state of its LSPs with the PCE using PCRpt messages. Within these messages the LSP object has a Synchronisation-flag and each PCRpt sent to the PCE during state synchronisation has this flag set to 1. When the state synchronisation is complete, an end-of-synchronisation marker is a PCRpt message with the Synchronisation-flag set to 0. If a PCC has no LSPs it should still send the end-of-synchronisation marker, and the sole PCRpt message in this output is the end-of-synchronisation marker.

Output 6-21: R1's PCEP Peer Status

```
A:admin@R1# show router pcep pcc peer 192.0.2.254 detail

===============================================================================
PCEP Path Computation Client (PCC) Peer Info
===============================================================================
IP Address            : 192.0.2.254     Preference             : 0
Admin Status          : Up              Oper Status            : Up
Peer Capabilities     : stateful-delegate stateful-pce segment-rt-path rsvp-
                        path pce-initiated-lsp p2mp p2mp-delegate p2mp-
                        initiate association multipath
Speaker ID            : 00:f7:ff:00:00:00
Sync State            : done            Peer Overloaded        : False
Session Establish Time: 0d 00:00:50
Oper Keepalive        : 30 seconds      Oper DeadTimer         : 120 seconds
Session Setup Count   : 2               Session Setup Fail Count: 0

-------------------------------------------------------------------------------
Statistics Information
-------------------------------------------------------------------------------
                              Sent                    Received
-------------------------------------------------------------------------------
PC Request Message            0                       0
PC Reply Message              0                       0
PC Error Message              0                       0
PC Notification Message       0                       0
PC Keepalive Message          2                       2
PC Update Message              0                        0
PC Initiate Message            0                        0
PC Report Message              1                        0
Path Report                    1                        0
Path Request                   0                        0
-------------------------------------------------------------------------------
```

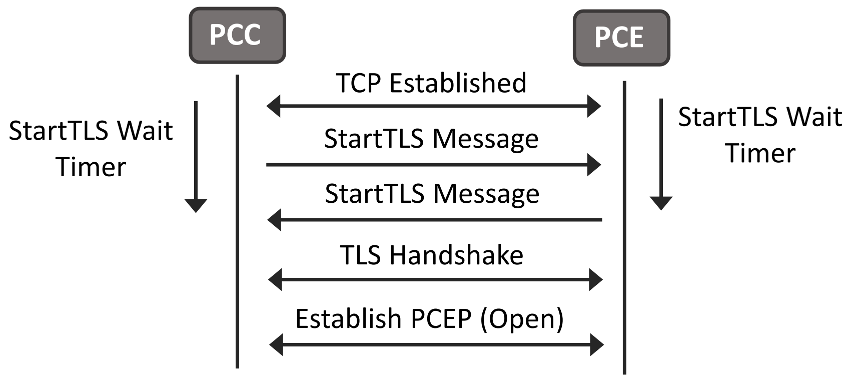
It is worth mentioning that by default when attempting to establish the PCEP session SR-OS firstly checks whether the remote PCE is routable through the management interface. If there is a route within the management routing context, SR-OS will attempt to establish the session through it (out-of-band). If no route exists in the management routing context, the router reverts to using the global routing table (in-band). Within the **pcep pcc peer** context a command **route-preference** allows the user to select **inband** or **outband**, or **both** as the preferred routing instance on a per-peer basis.

PCEP Secure (PCEPS)

To protect PCEP against security attacks the base specification suggests the use of the TCP Authentication Option (TCP-AO) or Transport Layer Security (TLS). With its inherent ability to provide peer authentication, message encryption, and message integrity, SR-OS supports TLS. The rules for the application of TLS to PCEP are defined in [68] which refers to this secure transport for PCEP as PCEPS. The steps involved in establishing a PCEPS session are as follows:

a. Establishment of a TCP connection.
b. Initiation of the TLS procedures using a *StartTLS* message from the PCE to the PCC and from the PCC to the PCE.
c. Negotiation and establishment of a TLS connection.
d. Start exchange of PCEP messages.

Figure 6-5: Establishing a PCEPS Session



As PCEP can operate either with or without TLS, it is necessary for a PCEP speaker to indicate whether it wants to use TLS or not. To this end, a PCEP message known as the *StartTLS* message is introduced. As indicated in the steps above, this message is exchanged after the TCP session is established, but before any PCEP messages are exchanged (including the Open message). The PCC initiates the use of TLS by sending the StartTLS message and the PCE agrees to the use of TLS by responding with its own StartTLS message, after which the peers are ready to start the negotiation and establishment of TLS. After the initial establishment of the TCP connection, each PCEP speaker starts a timer known as the *StartTLSWait* timer. If this timer expires before a StartTLS message or an Open message (if the peer does not support PCEPS) is received a PCErr message is generated, and the TCP session is closed. By default, PCEP peers that support PCEPS should default to strict TLS and not allow non-TLS PCEP sessions to be established. Although [68] allows a PCC that supports PCEPS to revert to a non-PCEPS connection (through the implementation of local policy) if it receives a PCErr message from the PCE, an SR-OS PCC operates in strict mode and does not currently support this. In short, if the StartTLSWait timer expires the problem needs to be resolved with human intervention.

Once the TLS session is established it uses a handshake to accomplish three main functions. It negotiates acceptable cipher suites and parameters to be used for encryption and integrity. It authenticates one or both parties through the exchange of X.509 digital certificates, and it uses asymmetric (public/private) keys for encryption of the exchange but will confidentially generate a shared key to allow subsequent exchanges to use symmetric encryption. TLS is inherently asymmetric in nature and the handshake uses a client/server approach. When applied to PCEPS, the PCC acts as the TLS client while the PCE acts as the TLS server. As a general rule only server-side authentication is used with TLS - the server presents an X.509 digital certificate to the client during the handshake and the client validates its authenticity. However, TLS allows this to be extended to two-way authentication where the client digital certificate is also authenticated, and since [68] specifies a requirement for certificate-based mutual authentication two-way authentication is necessary. The implications of this two-way authentication are not insignificant. Instead of just the PCE requiring an X.509 digital certificate, every PCEP speaker requires an X.509 digital certificate with the overhead that this entails.

As the purpose of this document is to describe how to implement Segment Routing in SR-OS, detailing the steps required to install a valid X.509 digital certificate for use with TLS is not covered here. That said, given that a reader may wish to implement PCEPS it would appear somewhat remiss to exclude that information. Therefore, the steps required to install an X.509 certificate for use with TLS are described in Appendix A, while this section will assume that the necessary digital certificates are installed and available for use with TLS. This section also assumes that the PCE is PCEPS capable and focuses purely on the requirements at the PCC.

Initially, a **ca-profile** is configured within the **pki** context that acts as a container for CA certificates. It references the CA certificate and certificate revocation list (CRL) using the **cert-file** and **crl-file** respectively, both of which are imported into the cf3:/system-pki/ directory using the procedures outlined in Appendix A. The **revocation-check** command allows the user to relax the otherwise mandatory check of the CRL if for example, the CRL has expired or the CRL file does not exist. It is then placed into an **admin-state** of **enable**.

Output 6-22: CA-Profile Configuration

```
    system {
        security {
            pki {
                ca-profile "EJBCA-CA" {
                    admin-state enable
                    cert-file "EJBCAcert"
                    crl-file "EJBCAcrl"
                    revocation-check crl-optional
                }
            }
        }
    }
```

Within the **tls** context, the configuration shown in *Output 6-23* is then applied to the PCC to enable the use of TLS. The **cert-profile** uses the **certificate-file** command to reference the end-entity certificate obtained from the CA together with the **key-file** command to reference the key that was used to generate the certificate request. Both of these files are imported into the cf3:/system-pki/ directory using the procedure described in Appendix A,

Recall that the PCC acts as a client during the TLS handshake, and the **client-cipher-list** contains a list of negotiated and acceptable ciphers and authentication algorithms for the TLS session establishment. The **trust-anchor-profile** is used to reference the previously configured **ca-profile** and is essentially a pointer to the relevant CA certificate and CRL file that the client should use to authenticate the server.

The **client-tls-profile** ties it all together and contains the configuration parameters for authenticating the server and negotiating acceptable TLS session attributes. It references the **cert-profile** containing its own end-entity certificate and key-pair. It references the **cipher-list** containing a list of acceptable ciphers that the session can use, and it references the **trust-anchor-profile** containing the CA certificate and CRL that are used to validate the server certificate when it is presented during the TLS handshake. When the **trust-anchor-profile** is configured under the **client-tls-profile**, the authenticity of the server is validated using the referenced CA certificate and CRL file before the TLS connection can be established. If no **trust-anchor-profile** is configured within the **client-tls-profile**, the client will do basic checks of the presented server certificate and check time validity, certificate type (the certificate is not a CA certificate), keyUsage extensions, and that the DNS name or IP address matches the peer in question. It will not however validate the issuing authority of the server certificate against a CA certificate and CRL. As such, the TLS connection can be established without the server being fully authenticated. Finally the **client-tls-profile** is placed into an **admin-state** of **enable**.

Output 6-23: TLS Configuration

```
    system {
        security {
            tls {
                cert-profile "pcep-tls-certificates" {
                    admin-state enable
                    entry 1 {
                        certificate-file "tls-cert"
                        key-file "tls-key"
                    }
                }
                client-cipher-list "pcep-cipher-list" {
                    tls12-cipher 1 {
                        name tls-rsa-with-aes256-cbc-sha256
                    }
                    tls12-cipher 2 {
                        name tls-rsa-with-aes256-gcm-sha384
                    }                }
                client-tls-profile "pcep-tls-profile" {
                    admin-state enable
                    cert-profile "pcep-tls-certificates"
                    cipher-list "pcep-cipher-list"
                    trust-anchor-profile "tls-trust-anchor"
                }
                trust-anchor-profile "tls-trust-anchor" {
                    trust-anchor "EJBCA-CA" { }
                }
            }
        }
    }
```

To enable PCEPS at the PCC, the **client-tls-profile** is assigned to the **pcc peer**. Within the same **peer** context, the **tls-wait-timer** command allows for configuration of the StartTLSWait timer, with a default value of 60 seconds.

Output 6-24: Assigning Client TLS Profile to PCC Peer

```
    router "Base" {
        pcep {
            pcc {
                peer 192.0.2.254 {
                    tls-client-profile "pcep-tls-profile"
                }
            }
        }
    }
```

Output 6-25 shows the active TLS connections associated with the previously configured **client-tls-profile** assigned to PCEPS. There is a single TLS connection active which is assigned to PCEP peer 192.0.2.254 (the PCE) and uses AES256 for encryption and SHA256 for authentication, which was **cipher 1** in the **client-cipher-list**.

Output 6-25: PCEPS TLS Connection

```
A:admin@R1# show system security tls client-tls-profile "pcep-tls-profile" connections

===============================================================================
Active TLS connections using client-tls-profile "pcep-tls-profile"
===============================================================================
     Cipher           Matched Trust Anchor
       Server IP
-------------------------------------------------------------------------------
Pcep
1    AES256-SHA256    EJBCA-CA
192.0.2.254:4189
-------------------------------------------------------------------------------
Number of TLS connections: 1
```

Finally, the state of the PCEPS session is validated and is both administratively and operationally up. The assigned TLS profile and StartTLSWait timer is also shown in the output.

Output 6-26: Operational State of PCEPS Session

```
A:admin@R1# show router pcep pcc peer 192.0.2.254

===============================================================================
PCEP Path Computation Client (PCC) Peer Info
===============================================================================
IP Address            : 192.0.2.254     Preference             : 0
Admin Status          : Up              Oper Status            : Up
Peer Capabilities     : stateful-delegate stateful-pce segment-rt-path rsvp-
                        path pce-initiated-lsp p2mp p2mp-delegate p2mp-
                        initiate association multipath
Speaker ID            : 00:f7:ff:00:00:00
Sync State            : done            Peer Overloaded        : False
Session Establish Time: 0d 00:08:36
Oper Keepalive        : 30 seconds      Oper DeadTimer         : 120 seconds
Tls Profile           : pcep-tls-profile
Tls WaitTimer         : 60 seconds
Route Preference      : both
```

Advertising Link-State Information Northbound

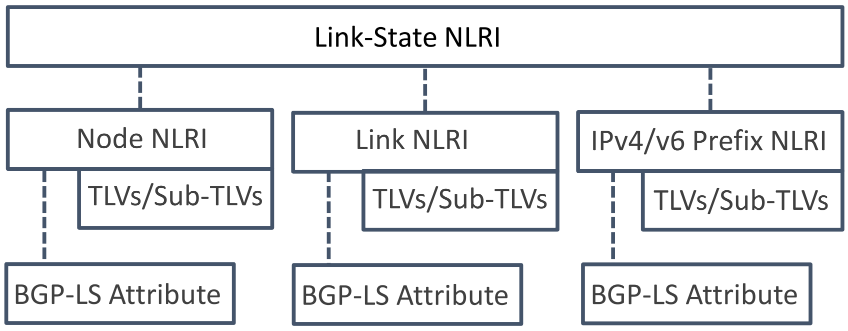
Before the PCE can compute paths, it needs to learn the topology of the network, and to achieve this the BGP Link State (BGP-LS) Address Family [24] is used. A BGP-LS speaker essentially extracts network topology information together with its associated traffic engineering parameters learnt through the IGP and encodes it into Link-State NLRI. Each Link-State NLRI can consist of Node NLRI, Link NLRI, IPv4 Topology Prefix NLRI, and IPv6 Topology Prefix NLRI:

- Node NLRI use Node Descriptor TLVs to encode local nodal parameters such as IGP Router-ID and AS number. An optional non-transitive BGP attribute known as the BGP-LS attribute is attached to the NLRI to encode additional node attributes in TLVs such as Node Name, IPv4/IPv6 Router-ID, and Multi-Topology Identifier.
- Link NLRI use local/remote Node Descriptor TLVs to encode information about the routers either side of a given link and Link Descriptor TLVs to encode link information such as local and remote link identifiers, and IP addressing. The BGP-LS attribute is again used to encode optional link attributes such as TE information. More accurately, Link Descriptor TLVs describe one side of a network link such that two Link NLRIs are required in order to fully describe a point-to-point link.
- IPv4 and IPv6 Topology Prefix NLRI encode IP reachability information and the protocol through those prefixes are learned.

The BGP-LS attribute used to carry optional additional information for each of the Node was extended in [24] to carry additional SR information encoded in TLVs:

- Node attributes were extended to include the SID/Label, SR Capabilities, SR Algorithm, and the SR Local Block TLVs.
- Link attributes were extended to include the Adjacency SID, LAN Adjacency SID, and L2 Bundle Member Attributes TLVs.
- IP Prefix attributes were extended to include the Prefix-SID, Range, Prefix Attribute Flags, source Router Identifier, and Source OSPF Router-ID TLVs.

Figure 6-6: BGP Link-State NLRI Structure



Suffice to say that whenever new objects and TLVs are added to BGP or IS-IS/OSPF, BGP-LS also requires the subsequent extension if the information needs to be propagated northbound to a controller. I am not going to attempt to list them here, and even if I did the list would very soon be out-of-date.

Routers R2 and R6 are selected to be BGP-LS speakers in the test topology, and each will peer in IBGP with the RR as clients. The PCE is also peered in IBGP with the RR as a client, and the RR will reflect routes learnt from R2 and R6 towards the PCE. *Output 6-27* shows the configuration applied at both routers to enable advertisement of link-state information in BGP-LS towards the RR. Under the ISIS or OSPF context, the **database-export** context is used to populate the extended TE database with specific parameters prior to export in BGP-LS. Within this context there is an **igp-identifier** parameter and a **bgp-ls-identifier**. The **igp-identifier** parameter is used to indicate the particular IGP instance if multiple instances are in use. The default value is zero, and as the test topology uses instance zero this parameter is not shown. The **bgp-ls-identifier** is specified by [24] and is used to uniquely identify the BGP-LS domain. All BGP-LS speakers within the flooding scope of the IGP must use the same BGP-LS identifier. The use of the BGP-LS identifier is deprecated by [26], however, implementations should continue to support it for backward compatibility. SR-OS uses a default value of zero, and [26] suggests this is the recommended value if this sub-TLV is advertised when advertising information into BGP-LS.

Under the BGP context, the **link-state-route-import** true command imports the contents of the extended TE database into the BGP RIB. The address-family **bgp-ls** is added on the peering session towards the RR (192.0.2.10), after which the contents of the imported routes will be advertised in BGP.

Output 6-27: BGP-LS Configuration at R2 and R6

```
    router "Base" {
        isis 0 {
            database-export {
                bgp-ls-identifier {
                }
            }
              }
        bgp {
            link-state-route-import true
            group "IBGP-Core" {
                family {
                    bgp-ls true
                }
            }
            neighbor "192.0.2.10" {
                group "IBGP-Core"
            }
        }
    }
```

By checking the BGP summary at R2 it can be observed that there are 57 updates sent towards the RR in BGP-LS in the short space of time that the BGP session has been established.

Output 6-28: BGP Summary at R2

```
A:admin@R2# show router bgp summary | match "BGP Summary" post-lines 12
BGP Summary
===============================================================================
Legend : D - Dynamic Neighbor
===============================================================================
Neighbor
Description
                   AS PktRcvd InQ  Up/Down   State|Rcv/Act/Sent (Addr Family)
                      PktSent OutQ
-------------------------------------------------------------------------------
192.0.2.10
                64496       3    0 00h00m22s 0/0/57 (LinkState)
                           34    0
-------------------------------------------------------------------------------
```

Output 6-29 shows an example of a Link-State NLRI containing Node NLRI advertised by R6 towards the RR. The output has been truncated to remove most of the BGP attributes except the ones pertinent to BGP-LS. Note the difference between the parameters described by the actual NLRI using the Local Node Descriptor TLV and the parameters described by the BGP-LS attribute.

Output 6-29: Link-State NLRI Containing Node NLRI

```
A:admin@R6# show router bgp routes bgp-ls hunt node
===============================================================================
BGP-LS Node NLRIs
===============================================================================
-------------------------------------------------------------------------------
RIB Out Entries
-------------------------------------------------------------------------------
Network:
 Type           : NODE-NLRI
 Protocol       : ISIS Level-2           Identifier     : 0x0
 Local Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  IGP Router Id      : 0x192000002006
Nexthop         : 192.0.2.6
To              : 192.0.2.10
... [snip]
-------------------------------------------------------------------------------
Link State Attribute TLVs :
 Node MSD : MSD-type 2 MSD-value 15
 Node Flag Bits : 0x10
 IS-IS Area Identifier (length 3) :  490001
 IPv4 Rtr-ID of Local Node : 192.0.2.6
 SR Capabilities : Flags: 0xc0
    Label 12000 range 8000
 SR Algorithm :  0
-------------------------------------------------------------------------------
```

A similar output is shown for a Link-State NLRI containing Link NLRI, again advertised by R6. The actual NLRI consists of the Local Node Descriptor TLV, Remote Node Descriptor TLV, and the Link Descriptor TLV, all containing the relevant sub-TLVs. The BGP-LS Attribute contains the link TE attributes and Adj-SID information.

Output 6-30: Link-State NLRI Containing Link NLRI

```
A:admin@R6# show router bgp routes bgp-ls hunt link
===============================================================================
BGP-LS Link NLRIs
===============================================================================
-------------------------------------------------------------------------------
RIB Out Entries
-------------------------------------------------------------------------------
Network:
 Type          : LINK-NLRI
 Protocol      : ISIS Level-2           Identifier     : 0x0
 Local Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  IGP Router Id      : 0x192000002003
 Remote Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  IGP Router Id      : 0x192000002002
 Link descriptor:
  IPV4 Interface Addr: 192.168.0.6
  IPV4 Neighbor Addr : 192.168.0.5
Nexthop        : 192.0.2.6
To             : 192.0.2.10
... [snip]
-------------------------------------------------------------------------------
Link State Attribute TLVs :
 Administrative group (color) : 0x0
 Maximum link bandwidth : 10000000 Kbps
 Max. reservable link bandwidth : 10000000 Kbps
 Unreserved bandwidth0 : 10000000 Kbps
 Unreserved bandwidth1 : 10000000 Kbps
 Unreserved bandwidth2 : 10000000 Kbps
 Unreserved bandwidth3 : 10000000 Kbps
 Unreserved bandwidth4 : 10000000 Kbps
 Unreserved bandwidth5 : 10000000 Kbps
 Unreserved bandwidth6 : 10000000 Kbps
 Unreserved bandwidth7 : 10000000 Kbps
 TE Default Metric : 100
 IGP Metric : 100
 Adjacency Segment Identifier (Adj-SID) :  flags 0x70 weight 0 label 524286
-------------------------------------------------------------------------------
```

Finally, *Output 6-31* shows a Link-State NLRI containing IPv4 Prefix NLRI. The NLRI contains a Local Node Descriptor TLV and Prefix Descriptor TLV; in this case carrying the prefix 192.0.2.4/32 representing R4's system address. The BGP-LS attribute carries the prefix attributes such as prefix metric and Prefix-SID.

Output 6-31: Link-State NLRI Containing IPv4 Prefix NLRI

```
A:admin@R6# show router bgp routes bgp-ls hunt ipv4-prefix
===============================================================================
BGP-LS Ipv4 NLRIs
```

```
===============================================================================
-------------------------------------------------------------------------------
RIB Out Entries
-------------------------------------------------------------------------------
Network:
 Type          : IPv4-Pfx-NLRI
 Protocol      : ISIS Level-2           Identifier     : 0x0
 Local Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  IGP Router Id      : 0x192000002004
 Prefix descriptor:
  IPv4-Addr/Pref-Len : 192.0.2.4/32
Nexthop        : 192.0.2.6
To             : 192.0.2.10
... [snip]
-------------------------------------------------------------------------------
Link State Attribute TLVs :
 IGP Flags (length 1) :  00
 Prefix Metric (length 4) :  00000000
 Prefix SID : flags 0x60 algorithm 0 sid 2004
 IGP Prefix Attributes (length 1) :  30
-------------------------------------------------------------------------------
```

When using a Route-Reflector to advertise Link-State information it is worth remembering that BGP-LS is fundamentally a northbound address family. Because routers R2 and R6 are both configured to support the BGP-LS address family they will unnecessarily receive one another's BGP-LS updates reflected by RR1. There is no requirement for BGP-LS updates to be reflected to any other client other than the PCE, so I use policy on the RR to prevent this from happening. *Output 6-32* shows an example of the policy used at RR1, where the **policy-statement** "NO-EXPORT-BGP-LS" rejects routes from family BGP-LS, which is subsequently applied as an export policy to neighbours 192.0.2.2 (R2) and 192.0.2.6 (R6).

Output 6-32: Policy Applied at RR1

```
    policy-options {
        policy-statement "NO-EXPORT-BGP-LS" {
            entry 10 {
                from {
                    family [bgp-ls]
                }
                action {
                    action-type reject
                }
            }
        }
    }
    router "Base" {
        bgp {
            neighbor "192.0.2.2" {
                export {
                    policy ["NO-EXPORT-BGP-LS"]
                }
            }
            neighbor "192.0.2.6" {
                export {
                    policy ["NO-EXPORT-BGP-LS"]
                }
            }
        }
    }
```

PCC-Initiated LSPs and Interaction with a PCE

An LSP that is configured on the router is referred to as a PCC-initiated LSP. When using PCC-Initiated SR-TE LSPs, the manner in which a PCC can interact with a PCE can vary. The PCC may select to retain control of its LSPs, or it may delegate control of its LSPs to a PCE. Equally, it may compute the path locally, or request that the PCE compute and provide a path. The options for a PCC to interact with a PCE are as follows:

- PCC-initiated and PCC-controlled LSP: An LSP that has its path computed and controlled by the router. The state of the LSP may optionally be reported to a PCE, but the PCE acts as a stateful passive PCE and cannot make unsolicited changes to the path of the LSP.
- PCC-initiated and PCE-computed LSP. An LSP that has its path computed by the PCE at the request of the PCC, but from that point onwards the PCC retains control of the LSP. The state of the LSP may optionally be reported to a PCE, but again the PCE acts as a stateful passive PCE and cannot make unsolicited changes to the path of the LSP.
- PCC-initiated and PCE-controlled LSP. An LSP that has its path computed by the PCE, after which the PCC delegates control of the LSP to the PCE. The PCE may subsequently make unsolicited updates and changes to the path of the LSP.

These options are described in this sub-section. The use of PCC-Initiated LSPs is implicit throughout this section.

PCC-Initiated and PCC-Controlled LSPs

When the path of an SR-TE LSP is computed locally the computed path can be the result of explicit path definition using IP addresses or SIDs, or it can be the result of a local CSPF. Regardless of which approach is used, the resultant path and operational state of the SR-TE LSP can be reported to a PCE whilst the PCC retains control of it. This could be a passive PCE, although given the industry trend towards stateful active PCEs, it is more likely a stateful active PCE operating in passive mode. A use-case for computing SR-TE LSPs locally and reporting the state of the LSP to a PCE might be a first step towards adoption of a PCE. Handing control of path computation to a centralised controller is a reasonably significant step for any operator, so a first phase might be to provide the operator with visibility of its existing LSPs and this can be achieved by using the PCRpt message.

Output 6-33 shows the configuration of an SR-TE LSP at R1 that uses a local CSPF to reach R6. The configuration details for this form of LSP have been covered previously in this chapter, but the point to emphasise in the output is the presence of the **pce-report true** command. This tells the PCC to synchronise the state of this LSP to the PCE, not just after the initial creation, but for every subsequent transition in state.

Output 6-33: PCC-Initiated and PCC-Controlled SR-TE LSP

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "R1-to-R6" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.6
                pce-report true
                path-computation-method local-cspf
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "empty" {
                    admin-state enable
                }
            }
        }
```

Debug 6-3 shows the PCRpt message sent from R1 to the PCE after the LSP is placed into an **admin-state** of **enable**. Although the debug output doesn't strictly categorise fields into PCEP objects, I will do so as I describe the pertinent ones because it gives the output some structure and it is useful to understand the purpose of some of the more common objects.

- The SRP-ID and path setup type (PST) are carried in the SRP object. The SRP-ID is used to correlate errors and LSP state reports to LSP update requests, and the PCC must include the SRP-ID that it received in a PCUpd message in the corresponding PCRpt message. In this case it is set to zero because there was no prior PCUpd message. The path setup type (PST) is SR.
- The LSP object is used to specify the target LSP and define the operation to be carried out on that LSP. The object contains a PLSP-ID and a number of flags, followed by a number of TLVs. The PLSP-ID is a PCEP-specific identifier for the LSP created by the PCC and lasts for the lifetime of a PCEP session. The flags indicate the state of the LSP and any operation to be carried out on the LSP. In *Debug 6-3*, the Administrative flag is set to 1 to indicate that the LSP is operationally active, and the Operational flag is set to 2 to indicate that the LSP is up and carrying traffic (1 represents signalled, while 0 is not active). All other flags are unset. When used in a PCRpt message the LSP object also carries an LSP-Identifiers TLV and a Symbolic-Path-Name TLV. The LSP-Identifiers TLV includes fields such as source and destination address, LSP ID, Tunnel ID, and Extended Tunnel ID which use the same semantics as the Sender Template object in RSVP-TE. The Symbolic-Path-Name TLV (shown simply as '`pathName`') encodes a unique symbolic path name for every LSP created by the PCC, and this remains constant for the lifetime of the LSP. SR-OS concatenates the LSP name and the path name, separated with two colons to create this name.
- The LSP constraints are contained in the Bandwidth object, LSP Attribute (LSPA) object, and Metric objects. Bandwidth is fairly self-evident, and unlike RSVP-TE its value local purely to the head-end for SR, but nonetheless may be useful information to provide to a PCE. The LSPA object specifies various TE LSP attributes and includes the ability to include/exclude admin-groups as well as signalling setup/hold priorities, the default of

which is 7/0. The Metric object appears multiple times, once each for IGP metric, hop-count, and MSD. Each Metric object includes a Bound-flag (B-flag) and a Computed-Metric-flag (C-flag). The B-flag is used in a PCReq message to signal to the PCE a maximum value for the path metric that must not be exceeded when computing the path. The C-flag when used in a PCReq message specifies that the PCE must provide the computed path metric value in the corresponding PCRep message. Each of these flags has an associated False (F) or True (T) indication in the output and a value where appropriate. Note that the MSD Metric has a value of 5 with the B-flag set which corresponds to the **max-sr-labels label-stack-size** command in the LSP configuration.

- Lastly, the ERO and RRO objects respectively encode the path of the TE LSP through the network and the reported route through the network.

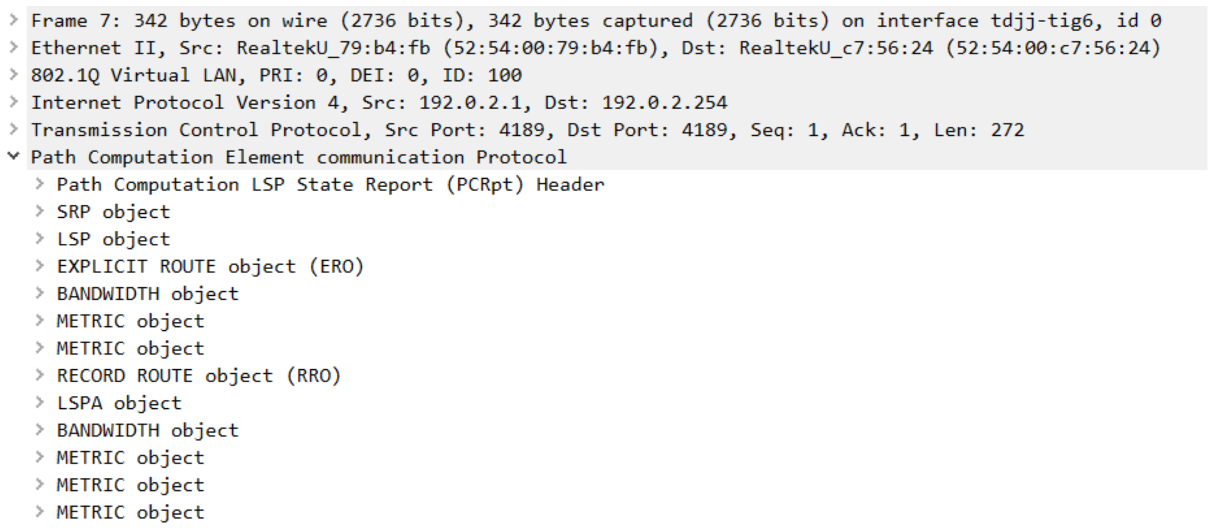
Debug 6-3: PCRpt Message from R1

```
1 2021/11/08 08:57:57.251 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][001 21:20:24.720]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:1 lspId: 55808 tunnelId:1}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 0 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.6 extTunnelId :: pathName R1-to-R6::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
             {{bw 0 isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
      Ero Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524286:0:0:0  NAI  T  Nai:  local
192.168.0.1 remote 192.168.0.2}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524282:0:0:0  NAI  T  Nai:  local
192.168.0.13 remote 192.168.0.14}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524279:0:0:0  NAI  T  Nai:  local
192.168.0.25 remote 192.168.0.26}}
            {{Attr: {bw 0 te-metric 0 igp 300 hop 4}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
      RRO:
            {{ctype SegRt SType 3 Label 524286:0:0:0 NAI T Nai: local 192.168.0.1 remote
192.168.0.2}}
            {{ctype SegRt SType 3 Label 524282:0:0:0 NAI T Nai: local 192.168.0.13 remote
192.168.0.14}}
            {{ctype SegRt SType 3 Label 524279:0:0:0 NAI T Nai: local 192.168.0.25 remote
192.168.0.26}}
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

Reporting LSP state in this manner is a single message mechanism which provides the PCE with visibility of the LSP. To reiterate however, the PCC retains control of the LSP and the PCE is therefore not able to modify the LSP state.

Although not shown in *Debug 6-3* the Bandwidth object and the Metric objects for IGP metric and hop-count are sent twice with potentially different values in the PCRpt message. This is because the path encoded in a PCRpt message is constructed of an *actual-attribute-list* and an *intended-attribute-list*. The actual-attribute-list consists of the actual computed values of the Bandwidth and Metric objects for the path in its current placement. The intended-attribute-list consists of the values of the Bandwidth and Metric objects and represent the initial requested parameters of the path. The reason for including the intended-attribute-list in a PCRpt message is to support a switch from a passive PCE to an active stateful PCE. If the stateful PCE is aware of the originally requested parameters for a given path it is better equipped to make informed decisions on any potential optimisations to that path. The presence of the actual and intended attribute lists can be observed in *Figure 6-7* (which unfortunately cannot be further expanded to show the contents of the objects simply because the output would be too large). The Bandwidth and two Metric objects that follow the Explicit Route object represent the actual-attribute-list. The bandwidth object has a value of zero throughout. The first Metric object is the IGP metric and has a value of 300 while the second Metric object is the hop-count metric which has a value of 4. The Bandwidth and three Metric objects that follow the LSPA object represent the intended-attribute-list. The first Metric object is the IGP metric and that has an infinite value metric. The second Metric object is the hop-count and that is set to a value of 255. The final Metric object is the MSD, and that has a value of 5.

Figure 6-7: Packet Capture of PCRpt Message



```
> Frame 7: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface tdjj-tig6, id 0
> Ethernet II, Src: RealtekU_79:b4:fb (52:54:00:79:b4:fb), Dst: RealtekU_c7:56:24 (52:54:00:c7:56:24)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 192.0.2.254
> Transmission Control Protocol, Src Port: 4189, Dst Port: 4189, Seq: 1, Ack: 1, Len: 272
v Path Computation Element communication Protocol
  > Path Computation LSP State Report (PCRpt) Header
  > SRP object
  > LSP object
  > EXPLICIT ROUTE object (ERO)
  > BANDWIDTH object
  > METRIC object
  > METRIC object
  > RECORD ROUTE object (RRO)
  > LSPA object
  > BANDWIDTH object
  > METRIC object
  > METRIC object
  > METRIC object
```

PCC-Initiated and PCE-Computed LSPs

It is possible of course for a PCC to request the PCE to compute the path but to subsequently retain control of that LSP. Using the same SR-TE LSP from R1 to R6 the **pce-report** command remains set to **true**, and the **path-computation-method** is changed to **pce** to initiate the path computation request. This will be actioned in a three-message exchange consisting of a Path Computation Request (PCReq), Path Computation Reply (PCRep), and a Path Computation LSP State Report (PCRpt).

Output 6-34: PCC-Initiated and PCE-Computed SR-TE LSP

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "R1-to-R6" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.6
                pce-report true
                path-computation-method pce
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "local-cspf" {
                    admin-state enable
                }
            }
        }
```

Debug 6-4 shows the PCReq message from the PCC R1. Many of the fields in the objects have previously been described so I won't repeat them here, however, there are also additional objects in the PCReq message:

- An End-Points object is used in a PCReq message to specify the source and destination IP addresses of the path for which a path computation is requested. (This is in addition to the IPV4-LSP-Identifiers TLV in the LSP object, which also defines a tunnel sender address and tunnel endpoint address.)
- The debug output indicates that a Synchronisation Vector (SVEC) object is present. The SVEC object is used to indicate a set of dependent path computation requests whose path computations cannot be performed independently of each other. The fields in the SVEC object output show examples of that form of path computation with node, link, and SRLG diversity (all set to false). Despite being in the debug output, the SVEC object is not currently supported in SR-OS and is not sent on the wire.

Debug 6-4: PCReq Message from R1

```
19 2021/11/08 14:02:34.383 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REQUEST ][002 02:25:01.850]
  Svec :{numOfReq 1}{nodeDiverse F linkDiverse F srlgDiverse F}
    Request: {id 3 PST(SegRt) 1 srcAddr 192.0.2.1 destAddr 192.0.2.6}
             {PLspId 4 tunnelId:1 lspId: 55818 lspName R1-to-R6::empty}
             {{bw 0 (0) isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
```

The corresponding PCRep message from the PCE is shown in *Debug 6-5* and predominantly shows the calculated path encoded in the ERO. Although not shown, the PCRep message will also contain the following objects:

- An RP object with an SRP-ID that correlates to the SRP-ID received in the PCReq message and a path setup type TLV indicating SR.
- An LSP object containing the PLSP-ID received in the PCReq message.
- An LSPA object indicating the setup/holding priority.
- A Bandwidth object indicating the actual computed bandwidth value.
- One or more Metric objects indicating the actual computed bandwidth values. These values will be sent in the PCRpt message as part of the actual-attribute-list.

The computed path from R1 to R6 in this instance is R1-R4-R5-R6 and each hop is a strict hop using an Adj-SID. If the PCE is unable to calculate a path satisfying the constraints of the PCReq message it will return a PCRep message containing a No-Path object. A 'Nature of

Issue' (NI) field in the No-Path object can be used to report the nature of the reason for failure.

Debug 6-5: PCRep Message from the PCE

```
21 2021/11/08 14:02:34.507 GMT minor: DEBUG #2001 Base PCC
PCC: [RX-Msg: REPLY ][002 02:25:01.980]
Peer 192.0.2.254
  Request: {id 3} {} Response has calculated path
        {{Total Paths: 1}}
          Path: {PST(SegRt) 1}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524284:0:0:0  NAI  T  Nai:  local
192.168.0.9 remote 192.168.0.10}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524283:0:0:0  NAI  T  Nai:  local
192.168.0.21 remote 192.168.0.22}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524279:0:0:0  NAI  T  Nai:  local
192.168.0.25 remote 192.168.0.26}}
            {{Attr: {bw 0 te-metric 0 igp 300 hop 3}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
```

The final message in the three-way message exchange is the PCRpt message sent by the PCC to report the state of the LSP. The contents of this message have already been described in *Debug 6-3* and are therefore not repeated here.

Debug 6-6: PCRpt Message from R1

```
22 2021/11/08 14:02:34.635 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][002 02:25:02.110]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:4 lspId: 55818 tunnelId:1}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 0 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.6 extTunnelId :: pathName R1-to-R6::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
             {{bw 0 isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
      Ero Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524284:0:0:0  NAI  T  Nai:  local
192.168.0.9 remote 192.168.0.10}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524283:0:0:0  NAI  T  Nai:  local
192.168.0.21 remote 192.168.0.22}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524279:0:0:0  NAI  T  Nai:  local
192.168.0.25 remote 192.168.0.26}}
            {{Attr: {bw 0 te-metric 0 igp 300 hop 4}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
      RRO:
            {{ctype SegRt SType 3 Label 524284:0:0:0 NAI T Nai: local 192.168.0.9 remote
192.168.0.10}}
            {{ctype SegRt SType 3 Label 524283:0:0:0 NAI T Nai: local 192.168.0.21 remote
192.168.0.22}}
            {{ctype SegRt SType 3 Label 524279:0:0:0 NAI T Nai: local 192.168.0.25 remote
192.168.0.26}}
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

Although the PCE was responsible for calculating the path of the SR-TE LSP, the PCC retains control of it and as such the PCE cannot make unsolicited updates to the LSP. If a link or node on the path were to fail, it is the responsibility of the PCC to detect it and make another path computation request.

PCC-Initiated and PCE-Controlled LSPs

To allow a PCE to take control of the LSP the PCC must delegate control of it. Once the PCE has control of the LSP, it can make unsolicited updates to it. This might be due to the failure of an element on the path, or as an optimisation due to congestion or increase in measured latency, or just a change in network topology. To delegate the LSP the command **pce-control true** is added to the configuration of the R1-R6 LSP. I should be clear that a **path-computation-method** of **pce** is not a prerequisite for delegation – the path could be computed locally and then delegated. To that extent, **pce-control**, **pce-report**, and **path-computation-method** are all somewhat independent. However, **pce-report** must be enabled when **pce-control** is enabled to allow the PCC to report back the state of the LSP after the initial path computation or any subsequent updates.

Output 6-35: PCC-Initiated and PCE-Controlled SR-TE LSP

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "R1-to-R6" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.6
                pce-control true
                pce-report true
                path-computation-method pce
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "empty" {
                    admin-state enable
                }
            }
        }
```

As the **path-computation-method** is **pce** once again, the message exchange will again be three-way and consist of PCReq, PCRep, and PCRpt messages. As the contents of the PCReq and PCRep messages will look predominantly the same as those shown in *Debug 6-4* and *Debug 6-5* I will not repeat them here for the purpose of conciseness. The objects and fields of a PCRpt message have also been previously described, but there is a notable difference in this PCRpt message which is shown in *Debug 6-7*. The LSP object now has three flags set. The Administrative flag is set to 1 to indicate that the LSP is operationally active, and the Operational flag is set to 2 to indicate that the LSP is up and carrying traffic. Note however that the Delegate flag is also now set to 1 to indicate to the PCE that it now has control of the LSP.

Debug 6-7: PCRpt Message with Delegation

```
71 2021/11/09 09:01:17.835 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][002 21:23:45.310]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:12 lspId: 55818 tunnelId:1}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 1 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.6 extTunnelId :: pathName R1-to-R6::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
             {{bw 0 isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
      Ero Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524284:0:0:0  NAI  T  Nai:  local
192.168.0.9 remote 192.168.0.10}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524283:0:0:0  NAI  T  Nai:  local
192.168.0.21 remote 192.168.0.22}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524279:0:0:0  NAI  T  Nai:  local
192.168.0.25 remote 192.168.0.26}}
            {{Attr: {bw 0 te-metric 0 igp 300 hop 4}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
      RRO:
            {{ctype SegRt SType 3 Label 524284:0:0:0 NAI T Nai: local 192.168.0.9 remote
192.168.0.10}}
            {{ctype SegRt SType 3 Label 524283:0:0:0 NAI T Nai: local 192.168.0.21 remote
192.168.0.22}}
            {{ctype SegRt SType 3 Label 524279:0:0:0 NAI T Nai: local 192.168.0.25 remote
192.168.0.26}}
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

A PCC may revoke delegation at any time over the lifetime of the LSP. If the command **pce-control true** was removed from the R1-to-R6 LSP configuration (or set to **false**), R1 would send a further PCRpt message to the PCE with the delegate bit unset. Equally, the PCC can return an LSP delegation at any point during the lifetime of a PCEP session.

The path returned by the PCE for the R1-to-R6 LSP is routed via R1-R4-R5-R6 and the SR tunnel is instantiated by the Adj-SIDs associated with those hops. If there is a failure anywhere along this path it is the responsibility of the controlling entity to take restorative action. In this example the controlling entity is the PCE, so I'll now look at what happens during a failure condition.

Output 6-36: R1-to-R6 LSP Path

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R6" path detail | match "Actual Hops" post-
lines 3
Actual Hops       :
    192.168.0.10    (A-SID)                      Record Label        : 524284
 -> 192.168.0.22    (A-SID)                      Record Label        : 524283
 -> 192.168.0.26    (A-SID)                      Record Label        : 524279
```

With the R1-to-R6 LSP routed via R1-R4-R5-R6 the link R4-R5 is failed (shut down). The first thing that will occur after the link failure is detected is that R4 will activate the LFA backup path for the Adj-SID if one is available (fast reroute and LFA is covered in chapter 9) minimising the outage time for the LSP. The R4-R5 link will then be flushed from the link-state database, triggering a subsequent withdrawal of the same link in BGP-LS. The controller will now be aware of the failure and can take restorative action for impacted LSPs which are delegated to it and which it has control of. A new path for the R1-to-R6 is computed that avoids the failure, and a PCUpd message is sent to R1 with the newly-calculated path. There are three mandatory objects in a PCUpd message, the SRP object, the LSP object, and an ERO containing the intended path.

- The SRP object carries an SRP-ID, which allows for correlation between the PCUpd message and the PCRpt message that the PCC should send in response to it. It also contains the path-setup-type TLV indicating SR.
- The LSP object contains the PLSP-ID and this is used to target the relevant LSP. The value of the PLSP-ID was assigned by the PCC and signalled to the PCE in a PCRpt message (see *Debug 6-7*). It is constant for the lifetime of a PCEP session.
- The ERO contains the intended (or newly-computed) path. The new path routes via R1-R2-R3-R6 to avoid the failed R4-R5 link.
- Although not shown in *Debug 6-8*, the PCUpd message also contains LSPA, Bandwidth, and Metric objects. These are optional objects, and the intention of including the Bandwidth and Metric objects is to communicate the actual computed costs to the PCC. SR-OS populates the tunnel-table metric with the computed cost contained in the IGP Metric object.

Debug 6-8: Post-Failure PCUpd Message

```
80 2021/11/09 12:22:21.264 GMT minor: DEBUG #2001 Base PCC
PCC: [RX-Msg: UPDATE ][003 00:44:48.730]
Peer 192.0.2.254
  Update : {srpId:748 PST(SegRt):1 PLspId:12 lspId: 0 tunnelId:0}
    {Sync 0 Rem 0 AdminState 1 OperState 0 Delegate 1 Create 0}
    {srcAddr :: destAddr :: extTunnelId :: }
        {{Total Paths: 1}}
          Ero Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524286:0:0:0  NAI  T  Nai:  local
192.168.0.1 remote 192.168.0.2}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524281:0:0:0  NAI  T  Nai:  local
192.168.0.5 remote 192.168.0.6}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524281:0:0:0  NAI  T  Nai:  local
192.168.0.17 remote 192.168.0.18}}
            {{Attr: {bw 0 te-metric 0 igp 300 hop 3}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
```

In response to a PCUpd message R1 will establish the new path for the SR-TE LSP, and SR-OS implements this in a make-before-break manner to minimize traffic interruption. This make-before-break behaviour can be observed in the PCRpt message shown in *Debug 6-9* that R1 sends to the PCE. The PCRpt message contains information about two paths, and because of the potential length of the debug output, together with the fact that I have previously discussed the contents of a PCRpt message, I have significantly truncated the output.

The first PCRpt has an SRP object with an SRP-ID of 748, which means it is in response to the PCUpd message received from the PCC (as the SRP-ID is the same). The LSP object has an LSP-ID of 55826, and the flags have the Administrative flag set to 1 to indicate that the LSP is operationally active, the Operational flag is set to 2 to indicate that the LSP is up and carrying traffic (1 represents signalled, while 0 is not active), and the delegate bit set.

The second PCRpt has an SRP object with an SRP-ID of 0 because it doesn't correlate to any other message. Within the LSP object the PLSP-ID is again 12 because it refers to the same LSP, but the LSP-ID is 55818 hence refers to a different path than the first PCRpt. The flags have the Administrative flag set to 1 to indicate that the LSP is operationally active, but the Operational flag is 0 meaning it is not active. The delegate bit is set, but importantly the

Remove flag is also set. In PCRpt messages this indicates that the LSP has been removed from the PCC and that the PCE should remove all state from its database.

Hopefully it is self-evident that the first PCRpt message is the LSP state report for the 'make', and the second PCRpt message is the LSP state report for the 'break' and subsequent clean-up of state.

Debug 6-9: PCRpt in Response to the PCUpd

```
81 2021/11/09 12:22:21.405 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][003 00:44:48.880]
Peer 192.0.2.254
  Report : {srpId:748 PST(SegRt):1 PLspId:12 lspId: 55826 tunnelId:1}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 1 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.6 extTunnelId :: pathName R1-to-R6::empty}
    {Binding Type: 0 Binding Val : 0}
... [snip]
  Report : {srpId:0 PST(SegRt):1 PLspId:12 lspId: 55818 tunnelId:1}
    {Sync 0 Rem 1 AdminState 1 OperState 0 Delegate 1 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.6 extTunnelId :: pathName R1-to-R6::empty}
    {Binding Type: 0 Binding Val : 0}
... [snip]
```

SR-TE LSPs are installed in the tunnel-table with a default preference of 8. When the path is computed by a PCE the metric reflected in the tunnel-table is the metric calculated by the PCE and returned in the metric object of the PCRep message.

PCEP Failure for PCC-Initiated SR-TE LSPs

If one or more PCC-initiated SR-TE LSPs are delegated to a PCE and the last remaining PCEP session fails, the state of those SR-TE LSPs will remain intact and forwarding will continue as normal. This does however leave those LSPs exposed should there be a further failure. Assume that all PCEP sessions have failed, and a delegated SR-TE LSP is explicitly-routed through an arbitrary topology of R1-R2-R3-R4 using Adj-SIDs. Sometime after the failure of the PCEP session(s) the link R2-R3 fails. R2 will initially activate an LFA backup for the failed link if one exists, however, this Adj-SID repair tunnel will only be active for the period of the **adj-sid-hold** timer which has a default of 15 seconds (Adjacency-SID Protection is covered in further detail in Chapter 9). With no inherent control plane notification of the failure to the headend, the SR-TE LSP has effectively become a traffic black-hole.

To avoid this situation when all available PCEP sessions are down, the possibility exists for the PCC to revoke delegation of the LSPs and fallback to a local path computation. This is implemented in SR-OS at the SR-TE LSP level using the command **fallback-path-computation-method** which provides options for either **local-cspf** or **none**. The former runs a local CSPF as one would anticipate from the name, while the latter implements a hop-to-label translation. The **fallback-path-computation-method** command is only valid for SR-TE LSPs configured with **path-computation-method pce** regardless of whether **pce-control** is configured or not. In the example shown in *Output 6-37* router R1 has an SR-TE LSP to R3 that is PCE-computed and PCE-controlled but has a **fallback-path-computation-method** of **local-cspf**. Note that if **local-cspf** is selected as the fallback action, the CSPF will be IGP-based because the PCC has no knowledge of any other objectives the PCE may have used for path computation.

Output 6-37: PCC-Initiated LSP with Fallback Action

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                pce-control true
                pce-report true
                path-computation-method pce
                fallback-path-computation-method local-cspf
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "empty" {
                    admin-state enable
                }
            }
        }
```

When the PCEP session to the relevant PCE goes down the PCC must first wait for the expiry of the (PCEP) **redelegation-timer**. Once the **redelegation-timer** has expired the delegation of the SR-TE LSP is revoked and control of the LSP is thereafter returned to the PCC. The LSP initially remains on the same path as that computed by the PCE until the expiry of the **sr-te-resignal resignal-timer** after which a local path computation is attempted. The **sr-te-resignal resignal-timer** is the time period between attempts by the system to re-optimise all paths of all SR-TE LSPs.

The LSP from the example configuration in *Output 6-37* is used to illustrate the fallback action. The LSP from R1 to R3 is computed by the PCE and is currently routed on the path R1-R4-R5-R6-R3. R1's **redelegation-timer** is set to 90 seconds, and the **sr-te-resignal resignal-timer** is set to 30 minutes (which is the minimum value). The following example creates a disruption in the PCEP path between R1 and the PCE simulated by placing an IP filter on R3's egress interface towards the PCE which blocks port 4189 for R1's source IP address.

Initially, the path of the R1-to-R3 SR-TE LSP as computed by the PCE is recorded as routing via R1-R4-R5-R6-R3.

Output 6-38: R1-to-R3 LSP PCE-Computed Path

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Actual Hops" post-
lines 5
Actual Hops       :
    192.168.0.10    (A-SID)                        Record Label        : 524284
 -> 192.168.0.22    (A-SID)                        Record Label        : 524283
 -> 192.168.0.26    (A-SID)                        Record Label        : 524279
 -> 192.168.0.17    (A-SID)                        Record Label        : 524285
```

Once the IP filter is applied, R1 declares its PCEP session to the PCE down after the dead-time interval has expired.

Debug 6-10: R1's PCEP Session Declared Down

```
165 2021/11/10 10:06:09.713 GMT minor: DEBUG #2001 Base PCC
PCC: pcepPeerConnectDisable
Disable connection: peer 192.0.2.254:4189 adminState 1:1
```

Once the PCEP redelegation timer has expired (90 seconds), R1 revokes its delegation. Suffice to say, if the PCEP session were operational this would be in the form of a PCRpt message with the Delegate-bit unset, but R1 obviously cannot send that message with the PCEP session down. The process of internally going through the process of revoking the delegation can however be seen in *Debug 6-11*. Note that although the delegation is revoked (`Delegate No`) the path of the LSP remains the same as that computed by the PCE, and the LSP is fully operational.

Debug 6-11: Revocation of Delegation

```
169 2021/11/10 10:07:39.125 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE : PCC
PCE disabled for SR-TE. Reset SR-TE PCE Delegate bit

170 2021/11/10 10:07:39.125 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE : PCC
Send PCC LSP Status for R1-to-R3::empty(LspId 24064)
GenId 1, LspState ACTIVE
UpdateId 0, UpdateFailed N/A, Delegate No, LspErrCode 0
pceInitErrCode 0 PLspId 0 IsPceInit No
LspType SR-TE, TunnelId 4, PathIdx 2
LspId 24064, VrId 1, ExtTunnelId 192.0.2.1
LspName "R1-to-R3::empty"
From 192.0.2.1, To 192.0.2.3
Bandwidth 0Mbps, HopLimit 255, SegRtMSD 5
MetricTypeTe No, LocalProtDesired Yes, SetupPri 7, HoldPri 0
ExcludeAny: 0x0, IncludeAny 0x0, IncludeAll 0x0
IRO:
  No hops
PceAssociations: None
ERO:
  [1] SIDType IPv4 Adj, SID 2147467264, RemoteAddr 192.168.0.10, LocalAddr 192.168.0.9,
Strict
  [2] SIDType IPv4 Adj, SID 2147463168, RemoteAddr 192.168.0.22, LocalAddr 192.168.0.21,
Strict
  [3] SIDType IPv4 Adj, SID 2147446784, RemoteAddr 192.168.0.26, LocalAddr 192.168.0.25,
Strict
  [4] SIDType IPv4 Adj, SID 2147471360, RemoteAddr 192.168.0.17, LocalAddr 192.168.0.18,
Strict
OperBandwidth 0Mbps, HopCnt 5, IgpMetric 400, TeMetric 0
RRO:
  [1] SIDType IPv4 Adj, SID 2147467264, RemoteAddr 192.168.0.10, LocalAddr 192.168.0.9
  [2] SIDType IPv4 Adj, SID 2147463168, RemoteAddr 192.168.0.22, LocalAddr 192.168.0.21
  [3] SIDType IPv4 Adj, SID 2147446784, RemoteAddr 192.168.0.26, LocalAddr 192.168.0.25
  [4] SIDType IPv4 Adj, SID 2147471360, RemoteAddr 192.168.0.17, LocalAddr 192.168.0.18
```

The R1-to-R3 LSP remains on the path computed by the PCE until the expiration of the **sr-te-resignal resignal-time**, which as explained is set to the minimum value of 30 minutes. *Debug 6-12* shows the sequence of events when that timer expires and commences with the first entry (179) initiating a timer-based make-before-break (MBB) resignal for LSP-ID

24064. The second entry (entry 180) indicates that the MBB path is up and uses an incremented LSP-ID of 24066 as this is a new path. Finally, the third entry (entry 181) indicates successful MBB resignal. The newly computed path for the R1-to-R3 LSP is now R1-R2-R3.

Debug 6-12: Timer-Based Resignal of the R1-to-R3 LSP

```
179 2021/11/10 10:11:25.115 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Initiate Timer-based Resignal MBB for LspPath R1-to-R3::empty(LspId 24064)

180 2021/11/10 10:11:25.305 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Timer-based Resignal MBB path R1-to-R3::empty(LspId 24066) is up

181 2021/11/10 10:11:25.305 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Timer-based Resignal MBB successful for LspPath R1-to-R3::empty(LspId 24064)

182 2021/11/10 10:11:25.305 GMT minor: DEBUG #2001 Base MPLS
MPLS: SR-TE : PCC
Send PCC LSP Status for R1-to-R3::empty(LspId 24066)
GenId 1, LspState ACTIVE
UpdateId 0, UpdateFailed N/A, Delegate No, LspErrCode 0
pceInitErrCode 0 PLspId 0 IsPceInit No
LspType SR-TE, TunnelId 4, PathIdx 2
LspId 24066, VrId 1, ExtTunnelId 192.0.2.1
LspName "R1-to-R3::empty"
From 192.0.2.1, To 192.0.2.3
Bandwidth 0Mbps, HopLimit 255, SegRtMSD 5
MetricTypeTe No, LocalProtDesired Yes, SetupPri 7, HoldPri 0
ExcludeAny: 0x0, IncludeAny 0x0, IncludeAll 0x0
IRO:
  No hops
PceAssociations: None
ERO:
  [1] SIDType IPv4 Adj, SID 2147475456, RemoteAddr 192.168.0.2, LocalAddr 192.168.0.1,
Strict
  [2] SIDType IPv4 Adj, SID 2147454976, RemoteAddr 192.168.0.6, LocalAddr 192.168.0.5,
Strict
OperBandwidth 0Mbps, HopCnt 3, IgpMetric 200, TeMetric 0
RRO:
  [1] SIDType IPv4 Adj, SID 2147475456, RemoteAddr 192.168.0.2, LocalAddr 192.168.0.1
  [2] SIDType IPv4 Adj, SID 2147454976, RemoteAddr 192.168.0.6, LocalAddr 192.168.0.5
```

The path via R1-R2-R3 is verified in *Output 6-39*. Recall that a local CSPF will return an explicit routed path using Adj-SIDs by default.

Output 6-39: R1-to-R3 Path after Local-CSPF

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Actual Hops" post-
lines 3
Actual Hops      :
    192.168.0.2(192.0.2.2)(A-SID)                Record Label        : 524286
 -> 192.168.0.6(192.0.2.3)(A-SID)                Record Label        : 524281
```

The beginning of this section provided a short overview of the PCEP protocol during which it mentioned that the stateful PCE extensions specify a mechanism for LSP state synchronisation between the PCC and the PCE after a PCEP session is established. When the PCEP session between R1 and the PCE is brought up again after the IP filter at R3 is removed, the mechanism through which this synchronisation is achieved can be seen. Albeit the state synchronisation in this example is somewhat brief given that R1 only has a single SR-TE LSP (to R3). To synchronise state the PCC takes a snapshot of its LSPs and then sends a sequence of LSP State Report (PCRpt) messages to the PCE describing those LSPs. Each of these PCRpt messages has the Synchronisation-flag (S-flag) in the LSP object set to 1. When the PCC has completed sending the PCRpt messages to describe its set of LSPs it sends an end-of-synchronisation marker, which is a PCRpt message with the Synchronisation-flag set to 0 and the PLSP-ID set to the reserved value of 0.

Debug 6-13 shows the first of those PCRpt messages sent by the PCC after the PCEP session is re-established. It describes the state of the R1-to-R3 LSP using objects present in a conventional PCRpt message but has the Synchronisation-flag in the LSP object set to 1. Note also that the Administrative and Operational flags are set, but the LSP is not delegated as yet. For the purpose of conciseness, some of the remaining objects used to describe the state of the LSP are truncated.

Debug 6-13: PCC LSP State Synchronisation

```
194 2021/11/10 10:13:11.125 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][003 22:35:38.600]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:17 lspId: 24066 tunnelId:4}
    {Sync 1 Rem 0 AdminState 1 OperState 2 Delegate 0 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId :: pathName R1-to-R3::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
            ...[snip]
      Ero Path:
            ...[snip]
      RRO:
            ...[snip]
```

Debug 6-14 shows the following PCRpt message in the sequence. It does not describe any LSP state but is denoted as the end-of-synchronisation marker because it has the PLSP-ID set to 0 and the Synchronisation-flag in the LSP object set to 0. The LSP object also contains the LSP-Identifiers TLV, but it uses a special value of all zeroes. If a PCC has no LSPs or state to synchronise, it should still send the end-of-synchronisation marker after the PCEP session is established.

Debug 6-14: End-of-Synchronisation Marker

```
196 2021/11/10 10:13:11.125 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][003 22:35:38.600]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):0 PLspId:0 lspId: 0 tunnelId:0}
    {Sync 0 Rem 0 AdminState 0 OperState 0 Delegate 0 Create 0}
    {srcAddr :: destAddr :: extTunnelId :: pathName }
    {Binding Type: 0 Binding Val : 0}
      RRO:
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

The final PCRpt message sent by R1 after the PCEP session is established is shown in *Debug 6-15*. It again describes the state of the R1-to-R3 LSP but now that the synchronisation process is complete the PCC can delegate it. As before, some of the remainder of the objects used to describe the state of the LSP are truncated for the purpose of conciseness.

Debug 6-15: Delegation of R1-to-R3 LSP

```
199 2021/11/10 10:13:11.125 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][003 22:35:38.600]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:17 lspId: 24066 tunnelId:4}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 1 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId :: pathName R1-to-R3::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
            ...[snip]
      Ero Path:
            ...[snip]
      RRO:
            ...[snip]
```

PCE-Initiated SR-TE LSPs

As previously described, PCE-Initiated SR-TE LSPs allow the PCE to dynamically create and remove LSPs based on user and or network application requirements. The use of PCE-Initiated LSPs requires the use of the PCInit message and the introduction of two new flags which will be described in this section. Before PCE-Initiated LSPs can be deployed by the PCE their use has to be negotiated. To this end, the Stateful-PCE-Capability TLV used in an Open message exchange introduced an LSP-Instantiation-Capability-flag (I-flag). As SR-OS supports PCE-Initiated LSPs this is a capability that is negotiated by default, however, some configuration work is required before PCE-Initiated LSPs can be created on the system.

Output 6-40 shows the configuration created on router R1 in readiness for PCE-Initiated SR-TE LSPs. The **pce-init-lsp** context only has the option for **sr-te** as this is the only type of PCE-Initiated LSP that is currently supported and must be placed into an **admin-state** of **enable** to allow PCE-Initiated LSPs to be created on the system. As a PCE-Initiated LSP has no configuration on the router, some base parameters need to be provided and these are inherited from an LSP template. Each LSP template has a template ID, and the PCE can reference different the template IDs using the concept of a Path-Profile ID. The use of Path-Profiles is described later in this chapter, but for now it is enough to understand that it is a PCEP object that allows a PCEP peer to invoke a policy on a remote peer using an opaque identifier. For example, if there is a need to use the **lsp-template** with the **template-id** of 1, the PCE can signal this requirement by including a Path-Profile object with a Path Profile ID of 1. If no Path-Profile object is used by the PCE the LSP template with a template ID of ‘default’ is used and therefore this template ID must exist in configuration. The example shows the **lsp-template** named “pce-init-template” which specifies the **template-id** of **default** (meaning there is no requirement to signal a Path Profile ID). The **lsp-template** includes a **type** field to indicate the type of LSP that is to be created and in this case is set to **p2p-sr-te-pce-init**. The **max-sr-labels label-stack-size** and **additional-frr-labels** commands have previously been discussed and are not further described here. The template uses a **default-path** command to reference a **path** that must also exist in the system. Finally, the **lsp-template** itself is put into an **admin-state** of **enable**.

An optional command **max-srte-pce-init-lsps** exists within the **pcep pcc** context that allows the user to constrain how many PCE-Initiated SR-TE LSPs can be created within the system.

Output 6-40: Configuration for PCE-Initiated LSPs

```
    mpls {
        path "empty" {
            admin-state enable
        }
        lsp-template "pce-init-template" {
            admin-state enable
            type p2p-sr-te-pce-init
            default-path "empty"
            pce-report true
            template-id default
            max-sr-labels {
                label-stack-size 5
                additional-frr-labels 2
            }
        }
        pce-init-lsp {
            sr-te {
                admin-state enable
            }
        }
    }
    pcep {
        pcc {
            max-srte-pce-init-lsps 1000
        }
    }
```

A PCE-Initiated SR-TE LSP is created on the PCE from R1 to R3 and pushed to the headend R1 and the output of that message is shown in *Debug 6-16*.

- The SRP object has an SRP-ID of 753 and includes a Path-Setup-Type TLV indicating SR.
- The LSP object uses a PLSP-ID of 0 in the PCInit message as this is a value that the PCC should select. The IPv4-LSP-IDENTIFIERS TLV in the LSP object contains the tunnel sender and tunnel endpoint addresses, and an LSP-ID, Tunnel ID, and Extended Tunnel ID of 0 as again these are values that the PCC should select. The Symbolic-Path-Name TLV is present and in this case is the name used by the PCE when creating the LSP. The flags field has the Create-flag set and the Delegate-flag set.
- The End-Points object contains the source and destination IP addresses for the SR-TE LSP.
- The Explicit-Route object (ERO) contains the intended path using explicit-routed Adj-SIDs via the path R1-R2-R3.
- The LSPA object shows any include/exclude admin-groups and the path setup/holding priority.
- There are two Metric objects that represent the actual values of the calculated path, one for IGP metric (value 200) and one for hop-count (value 2).

Debug 6-16: PCInit Message to R1

```
255 2021/11/10 16:36:58.790 GMT minor: DEBUG #2001 Base PCC
```

```
PCC: [RX-Msg: INITIATE ][004 04:59:26.260]
Peer 192.0.2.254
  PCInit : {srpId:753 PST(SegRt):1 PLspId:0 name: R1-to-R3-PCEInit lspId: 0 tunnelId:0}
    {Create 1 Rem 0 Delegate 1 templateId 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId 0.0.0.0}
    {EP: srcAddr 192.0.2.1 destAddr 192.0.2.3}
        {{Total Paths: 1}}
          Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524286:0:0:0  NAI  T  Nai:  local
192.168.0.1 remote 192.168.0.2}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524281:0:0:0  NAI  T  Nai:  local
192.168.0.5 remote 192.168.0.6}}
            {{Attr: {bw 0 te-metric 0 igp 200 hop 2}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
```

In response to a PCInit message the PCC should send a PCRpt message and this is shown in *Debug 6-17*:

- The SRP object has an SRP-ID of 753 reflecting the SRP-ID used in the PCInit message and thus allowing for correlation between the two messages.
- The LSP object now shows non-zero values for the PLSP-ID, LSP-ID, and Tunnel-ID as these are allocated by the PCC. The flags field is busy. The Administrative flag is set to 1 to indicate that the LSP is operationally active, and the Operational flag is set to 2 to indicate that the LSP is up and carrying traffic. The Delegate flag is set to 1 as control of the SR-TE LSP by the PCE is inherent for PCE-Initiated LSPs. Finally, the Create flag is set to 1 and will remain set to 1 in subsequent PCRpt messages to indicate that this LSP was created by a PCInit message.
- The ERO and RRO objects record the computed and actual paths respectively.
- There are three Metric objects present. The IGP Metric object and the Hop-Count Metric reflect the actual computed values and represent the actual-attributes-list. There is also a Metric object for MSD (not shown) that indicates an MSD of 11. As there is no configuration for this LSP and therefore no configured MSD, the maximum value (without service label) is sent to provide the PCE with this information should a re-optimisation take place.

Debug 6-17: PCRpt Message from R1 in Response to the PCInit

```
256 2021/11/10 16:36:58.905 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][004 04:59:26.380]
Peer 192.0.2.254
  Report : {srpId:753 PST(SegRt):1 PLspId:19 lspId: 50688 tunnelId:16387}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 1 Create 1}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId :: pathName R1-to-R3-PCEInit}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
      Ero Path:
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524286:0:0:0  NAI  T  Nai:  local
192.168.0.1 remote 192.168.0.2}}
            {{ctype  SegRt  isLoose  F,  SType  3  Label  524281:0:0:0  NAI  T  Nai:  local
192.168.0.5 remote 192.168.0.6}}
            {{Attr: {bw 0 te-metric 0 igp 200 hop 3}}}
            {{      {setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0}}}
      RRO:
            {{ctype SegRt SType 3 Label 524286:0:0:0 NAI T Nai: local 192.168.0.1 remote
192.168.0.2}}
            {{ctype SegRt SType 3 Label 524281:0:0:0 NAI T Nai: local 192.168.0.5 remote
192.168.0.6}}
```

```
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

The SR-TE LSP created by the PCInit message is populated into the tunnel-table with the IGP metric calculated by the PCE. It should be noted that PCE-Initiated SR-TE LSPs can only be used for *auto-bind* purposes and as a result their use is constrained to BGP next-hop resolution. They cannot be explicitly referenced or named in a Service Distribution Point (SDP) context typically used for classic Layer 2 services.

Output 6-41: Tunnel-Table Entry for PCInit SRTE LSP

```
A:admin@R1# show router tunnel-table 192.0.2.3 protocol sr-te

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-te     MPLS  671748    8      192.168.0.2    200
-------------------------------------------------------------------------------
```

As the SR-TE LSP was created by the PCE it can also be removed by the PCE. The PCInit message is again used for this purpose, and a Remove-flag (R-flag) in the SRP object is introduced to indicate a delete operation. This can appear a little confusing as there is a Remove-flag in the LSP object flags field and now another Remove-flag in the SRP objects field. To clarify:

- The Remove-flag in the SRP object is used explicitly to request the deletion of an LSP when set.
- The Remove-flag in the LSP object is used to indicate that the speaker has removed all state from its database and that the peer should do likewise.

Debug 6-18 shows the PCInit message sent by the PCE to R1 to remove the R1-to-R3 LSP. The LSP object uses the PLSP-ID to target the object that the operation should be carried out on, and observant readers will notice that PLSP-ID 19 was used in the instantiation of the LSP (see *Debug 6-17*). The Create, Remove, and the Delegate flags are all set to 1 and are grouped together between two curly brackets representing the flags field in the LSP object. The Remove-flag in the SRP object used explicitly for deletion is not shown. The template ID of 0 indicates that no Path Profile object is present and therefore the default template ID is used. The PCInit message also carries and End-Points object containing the source and destination IP addresses for the LSP.

Debug 6-18: PCInit Message for LSP Deletion

```
258 2021/11/11 08:30:36.070 GMT minor: DEBUG #2001 Base PCC
PCC: [RX-Msg: INITIATE ][004 20:53:03.540]
Peer 192.0.2.254
  PCInit   :   {srpId:754   PST(SegRt):1   PLspId:19   name:   R1-to-R3-PCEInit   lspId:   0
tunnelId:16387}
    {Create 1 Rem 1 Delegate 1 templateId 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId 192.0.2.1}
    {EP: srcAddr 192.0.2.1 destAddr 192.0.2.3}
        {{Total Paths: 0}}
```

Following the removal of the LSP the PCC responds with a PCRpt message with an SRP-ID that matches that of the corresponding PCInit message, and the PLSP-ID of 19 which had been used for the lifetime of the LSP. The flags field in the LSP object has the Remove-flag set to indicate that the PCC has removed all state relating to this LSP and the Create-flag set to indicate that it was originally created by a PCInit message.

Debug 6-19: PCRpt Message Confirming LSP Deletion

```
259 2021/11/11 08:30:36.071 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][004 20:53:03.540]
Peer 192.0.2.254
  Report : {srpId:754 PST(SegRt):1 PLspId:19 lspId: 0 tunnelId:0}
    {Sync 0 Rem 1 AdminState 0 OperState 0 Delegate 0 Create 1}
    {srcAddr :: destAddr :: extTunnelId :: pathName }
    {Binding Type: 0 Binding Val : 0}
      RRO:
      {Lsp Err NA RsvpErr 0 LspDbVersion 0}
```

Application of Policy

A PCE can have a number of locally held policies that have varying constraints and objectives. These policies may define a set of common constraints that can be applied to a number of LSPs autonomously, or they may define a set of constraints that require LSPs to be grouped together when considering how their paths are computed. The policy may not necessarily directly impact how a particular path is computed. It may just define some other function such as the activation of statistics collection. These policies and configuration parameters can be centrally managed on the PCE and made effective across multiple PCCs.

A PCC may need to request the application of one of these policies without explicit signalling as explicitly signalling the constraints and objectives of a path may not always be possible. For example, the PCEP speaker may not support the relevant extensions to do so, or those extensions may not even exist. Moreover, the constraints and objectives of a given path may not be directly related to the path computation, but to some other functionality related to the path such as statistics gathering. A generic mechanism for signalling policy without having to even know the specifics of the policy is therefore the preferred approach.

SR-OS supports two methods for signalling policy, *Path-Profiles* and *Association Groups*. Path-Profiles [27] was implemented in SR-OS as a pre-standard specification that ultimately did not progress to standards track and was superseded by use of the more extensible Association Groups [28]. Both are discussed in the following sub-sections.

Path-Profiles

A Path Profile is simply a PCEP object that allows a PCC to signal a Path Profile identifier as an opaque value which could then be mapped to a locally held policy on the PCE and applied to the path. The PCEP object also allows for the use of an optional Extended-ID field and uses an X-flag to indicate whether it is present or not. To indicate support for Path Profiles, a Path-Profile-Capability TLV is added to the Open object.

Output 6-42 shows the configuration of an SR-TE LSP from R1 to R3 using a Path Profile to specify a policy held in the PCE. The **path-profile** command simply takes a number in the range [1-4294967295] to represent the profile ID. In this example, the PCC uses the profile ID of 20 which is held locally on the PCE and uses TE Metric as an optimisation objective. There is an optional **path-group** argument that follows the **path-profile** command which allows the user to make use of the Extended ID field. Again, this takes a numeric value in the range [1-4294967295]. The Extended ID field (or **path-group**) is used to associate LSPs together in a group. For example, assume a path-profile with value 10 specifies LSP diversity. LSP one and LSP two belong to customer X and have a requirement to be diverse from each other. LSP three and LSP four belong to customer Y and also need to be diverse from each other. In this case LSP one and LSP two use path-profile 10 path-group 1, while LSP three and LSP four use path-profile 10 path-group 2. In other words, there are simply multiple iterations of the same policy.

Output 6-42: Path-Profile Configuration

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                pce-control true
                pce-report true
                path-computation-method pce
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                path-profile 20 {
                }
                primary "empty" {
                    admin-state enable
                }
            }
        }
```

Debug 6-20 shows the PCReq message signalled by the PCC R1. The last line indicates that the Path Profile object is present with a value of 20, and that the Extended ID is not present.

Debug 6-20: Signalling Path Profile in the PCReq

```
261 2021/11/12 10:09:04.313 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REQUEST ][005 22:31:31.780]
  Svec :{numOfReq 1}{nodeDiverse F linkDiverse F srlgDiverse F}
    Request: {id 15 PST(SegRt) 1 srcAddr 192.0.2.1 destAddr 192.0.2.3}
             {PLspId 20 tunnelId:5 lspId: 20992 lspName R1-to-R3::empty}
             {{bw 0 (0) isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
             {{path profile isOpt: F}}
                 {{profileId 20 extProfileId 0}}
```

Output 6-43 shows the path of the LSP after the PCE has returned the path computed on TE metric. The path routes via R1-R4-R5-R6-R3 which as shown in *Figure 6-4* is the shortest path route using TE metric.

Output 6-43: R1-to-R3 LSP Path

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Actual Hops" post-
lines 4
Actual Hops       :
    192.168.0.10    (A-SID)                       Record Label        : 524284
 -> 192.168.0.22    (A-SID)                       Record Label        : 524283
 -> 192.168.0.26    (A-SID)                       Record Label        : 524279
 -> 192.168.0.17    (A-SID)                       Record Label        : 524285
```

Association Groups

Association Groups define a method to associate LSPs with other LSPs that they interact with. It creates a generic mechanism to group LSPs, which can be used to define associations between sets of LSPs or between a set of LSPs and a set of attributes. There are various situations where multiple LSPs need to share common information, for example:

- When performing a make-before-break an association between the original path and the re-optimised path is required.
- If end-to-end protection is used an association between the primary and protection LSPs is required.
- For LSP diversity/disjointness, an association between the two LSPs is required.
- A grouping of LSPs may be useful in order to apply a common set of LSP parameters.
- Symmetrical path routing in order to maintain symmetric upstream/downstream latencies for delay-sensitive applications.

The Association Groups specification defines an Association object to consist of an Association Type, Association ID, and Association Source. Multiple Association Types can exist and [28] does not define these, but rather defines the format of the vanilla Association object which subsequently created Association Types can use. Some examples of Association Types that have so far been defined are given below:

- The Path Protection Type [29] (Association Type 1) specifies a method to associate two or more LSPs for the purpose of setting up path protection. The Association object for Path Protection contains an optional Path Protection Association TLV with a flags field to indicate whether the LSP is the protecting LSP (Protecting-flag) or the standby LSP (Secondary-flag).
- The Disjoint Association Type or DAT [30] (Association Type 2) describes a mechanism to associate a group of LSPs with the purpose of computing diverse (disjointed) paths for those LSPs. The Association object for DAT carries a Disjointness-Configuration TLV containing a number of flags to indicate the level of disjointness required, which may be link disjoint, node disjoint SRLG disjoint, or node and SRLG disjoint. It also carries a flag to indicate whether the disjointness should be

strictly enforced or whether the disjointness can be relaxed if the PCE fails to find a suitable path.
- The Policy Association Type or PAT [31] (Association Type 3) defines membership for a set of LSPs that share the same policy assignment, such as constraints, diversities, and optimisation criteria. This membership is referred to as the Policy Association Group (PAG). The Association object for PAT carries an optional Policy-Parameters-TLV that carries opaque information if required to apply a PCE policy correctly.

Each Association Type has its own Association ID space, which together with the Association Type and an Association Source uniquely identifies the association group. The identification of an association group is comparable to the numbering convention used with Path Profiles, where the association ID is equivalent to the path group ID, and the tuple {association type, association ID, association source} is equivalent to the path profile ID. The Association Source contains an IP address to identify the node that originated the association. To indicate support for Association groups, the Open message exchange is extended to include an optional Assoc-Type-List TLV to indicate the list of supported Association Types. It also defines an Op-Conf-Assoc-Range TLV to define an Association ID range for every supported Association Type.

SR-OS supports the Disjoint association type to signal diversity for PCC-initiated SR-TE LSPs, and the Policy association type for both PCC-initiated and PCE-initiated SR-TE LSPs. When used for PCE-initiated SR-TE LSPs the association ID is used by the PCE to signal which LSP template to bind to a PCE-initiated SR-TE LSP. Associations are configured in the **pce-associations** context and can be either **policy** or **diversity**. *Output 6-44* shows the configuration of a **policy** association type. Within the **policy** type, an **application-id** and **application-source** must be configured. For the policy association type the **application-id** has global meaning within a domain. That is, although the tuple {application type, application ID, application-source} is used to uniquely identify a given association, the application ID is used independent of the application source. For example, although the tuple {association-type=3, association-id=1, association-source=192.0.2.1} and the tuple {association-type=3, association-id=1, association-source=192.0.2.2} uniquely identify different associations, a PCE will use only the **application-id** to apply the relevant policy for the policy association type (3) and in this case policy 1 will be applied to both. The **application-source** defines the address of the node that originated the association and would typically be the same IP address as that used for the PCC source address.

Output 6-44: Policy Association Type Configuration

```
    router "Base" {
        pcep {
            pcc {
                pce-associations {
                    policy "Policy-One" {
                        association-id 1
                        association-source 192.0.2.1
                    }
                }
            }
        }
```

The association is applied to an SR-TE LSP using a **pce-associations** context within the relevant SR-TE LSP. In addition, associations can be applied to on-demand SR-TE LSPs by using the same **pce-associations** context within an **lsp-template**. Up to five associations can be applied to a given SR-TE LSP.

Output 6-45: Application of Association Type to LSP

```
        mpls {
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                pce-control true
                pce-report true
                path-computation-method pce
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                pce-associations {
                    policy "Policy-One" { }
                }
                primary "empty" {
                }
            }
        }
```

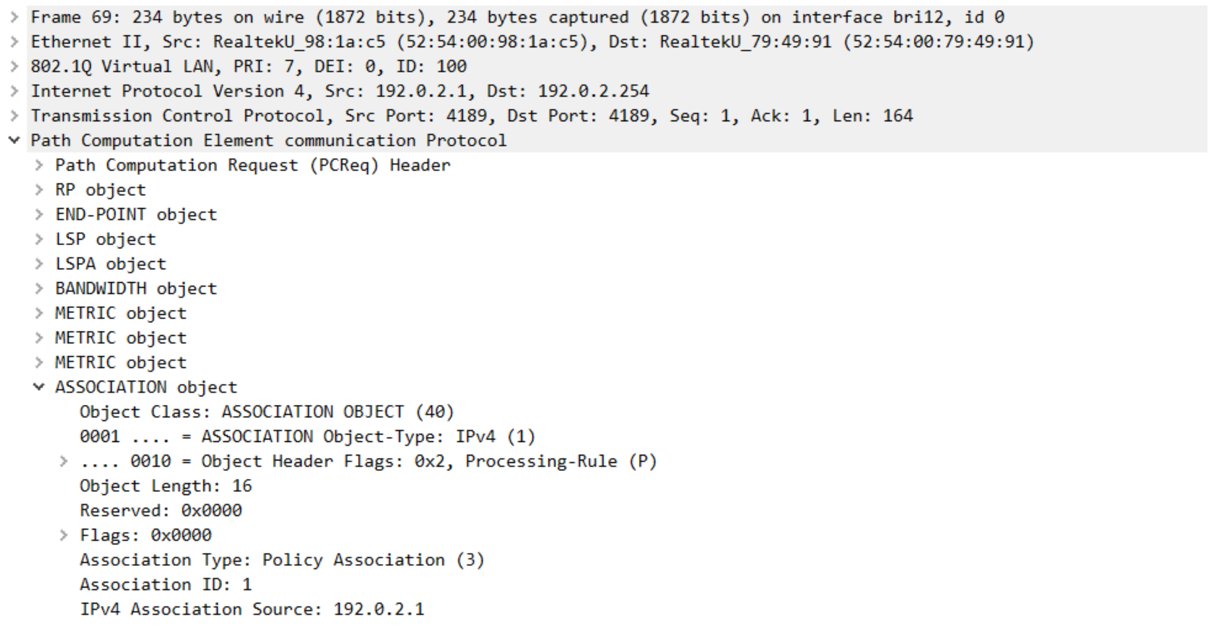
Output 6-46 shows the debug output of the PCReq message from the PCC towards the PCE with the policy association type configured. Although the debug output uses the classic 'path-profile' nomenclature, the contents of association ID, association type, and association source clearly identify this as an association object.

Output 6-46: Debug of PCReq Message with Association Object

```
4 2022/05/31 17:11:17.353 BST minor: DEBUG #2001 Base PCC
PCC: TX-Msg: REPORT 000 07:30:41.950
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:1 lspId: 51712 tunnelId:2}
    {Sync 0 Rem 0 AdminState 1 OperState 0 Delegate 1 Create 0}
    {srcAddr 192.0.2.1 destAddr 192.0.2.3 extTunnelId :: pathName R1-to-R3::empty}
    {Binding Type: 0 Binding Val : 0}
    Lsp Constraints:
             {{bw 0 isOpt: F}}
             {{setup 7 hold 0 exclAny 0 inclAny 0 inclAll 0 isOpt: F}}
             {{igp-met 16777215 B:F BVal:0 C:T isOpt: F}}
             {{hop-cnt 0 B:T BVal:255 C:T isOpt: F}}
             {{sr-msd 5 B:T BVal:5 C:F isOpt: F}}
             {{path profile isOpt: F}}
                 {{associd 1 assoctype 3 assocSrc: 192.0.2.1 remove 0}
}
      RRO:
      {Lsp Err unknown RsvpErr 0 LspDbVersion 0}
```

Figure 6-8 shows a packet capture of the same PCReq message where the association object can be clearly seen. A PCC may include the association object in a PCReq or a PCRpt message and a PCE may reflect the same association objects in a PCRep or PCUpd message.

Figure 6-8: PCReq Message with Policy Association Type



```
> Frame 69: 234 bytes on wire (1872 bits), 234 bytes captured (1872 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_98:1a:c5 (52:54:00:98:1a:c5), Dst: RealtekU_79:49:91 (52:54:00:79:49:91)
> 802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 100
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 192.0.2.254
> Transmission Control Protocol, Src Port: 4189, Dst Port: 4189, Seq: 1, Ack: 1, Len: 164
v Path Computation Element communication Protocol
  > Path Computation Request (PCReq) Header
  > RP object
  > END-POINT object
  > LSP object
  > LSPA object
  > BANDWIDTH object
  > METRIC object
  > METRIC object
  > METRIC object
  v ASSOCIATION object
      Object Class: ASSOCIATION OBJECT (40)
      0001 .... = ASSOCIATION Object-Type: IPv4 (1)
    > .... 0010 = Object Header Flags: 0x2, Processing-Rule (P)
      Object Length: 16
      Reserved: 0x0000
    > Flags: 0x0000
      Association Type: Policy Association (3)
      Association ID: 1
      IPv4 Association Source: 192.0.2.1
```

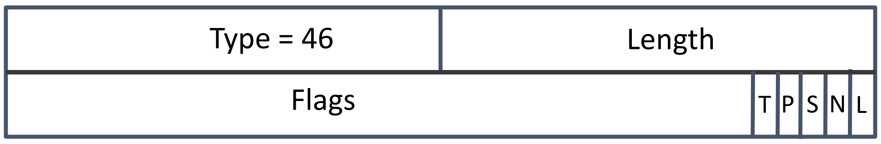
The application of diversity using Disjoint association type is much the same as for policy with one or two minor differences. When creating the **diversity** association within the **pce-associations** context the same **application-id** and **application-source** parameters exist that have been previously described, but there are some additional configuration options. The **diversity-type** is a required parameter, and can be one of **link**, **node**, **srlg-link**, or **srlg-node**. The **disjointness-type** specifies whether the PCE should return a no path object if the diversity constraints cannot be met or whether the diversity constraints can be relaxed and can be either **strict** or **loose** with a default value of **loose**. Finally, the **disjointness-reference** parameter is used to indicate if this LSP path is the reference for the disjoint set of paths in the association group. When set to **true** the PCE first computes the path of this LSP and then computes the paths of all other paths in the association group. The default is no **disjointness-reference**. For completeness, *Output 6-47* shows the configuration of the Disjoint association type. It is thereafter applied to an SR-TE LSP in the same manner as the previously described for the Policy association group (*Output 6-45*).

Output 6-47: Disjoint Association Type Configuration

```
    router "Base" {
        pcep {
            pcc {
                pce-associations {
                    diversity "Disjoint-LSPs" {
                        association-id 2
                        association-source 192.0.2.1
                        disjointness-type loose
                        diversity-type node
                    }
                }
            }
        }
    }
```

In order to signal the diversity requirements the Disjoint Association object must carry a Disjoint-Configuration TLV. The format of the Disjoint Configuration TLV is shown in *Figure 6-9*. It has a fixed length of 4-bytes and consists solely of a Flags field. Some explanation of the defined flags is given below the figure.

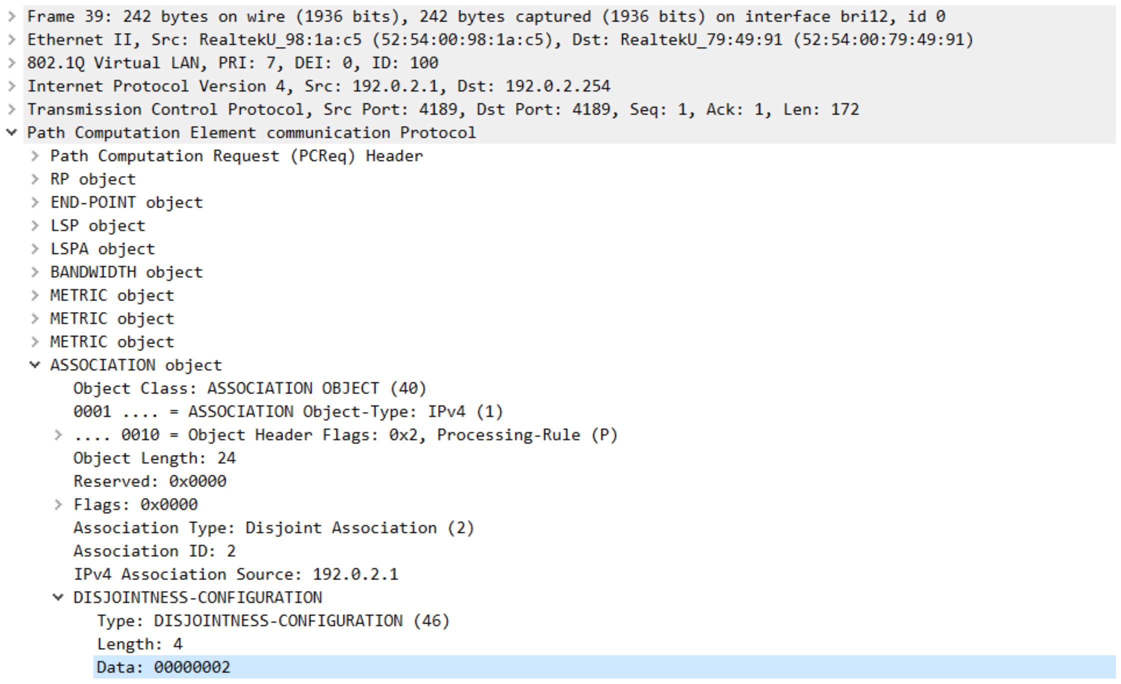
Figure 6-9: Disjoint Configuration TLV



L	Link Diverse bit. When set it indicates that the computed paths within the Disjoint association group must not have any link in common.
N	Node Diverse bit. When set it indicates that the computed paths within the Disjoint association group must not have node in common.
S	SRLG Diverse bit. When set it indicates that the computed paths within the Disjoint association group must not share any SRLG.
P	Shortest Path bit. When set it indicates that the computed path of the LSP should satisfy all the constraints and objective functions first without considering the diversity constraint.
T	Strict Disjointness bit. When set, if disjoint paths cannot be found the PCE must return no path for the LSPs that could not be disjoint. When unset the PCE is allowed to relax disjointness.

Figure 6-10 shows a PCReq message from the PCC containing the Association object carrying the Disjoint association type (2) consisting of the tuple association type, association ID, and association source, as well as the Disjoint Configuration TLV. Although the packet capture has failed to decode the flags field, the value 00000002 means that only the N-flag (Node Diverse bit) is set.

Figure 6-10: PCReq Message with Disjoint Association Type



```
> Frame 39: 242 bytes on wire (1936 bits), 242 bytes captured (1936 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_98:1a:c5 (52:54:00:98:1a:c5), Dst: RealtekU_79:49:91 (52:54:00:79:49:91)
> 802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 100
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 192.0.2.254
> Transmission Control Protocol, Src Port: 4189, Dst Port: 4189, Seq: 1, Ack: 1, Len: 172
v Path Computation Element communication Protocol
  > Path Computation Request (PCReq) Header
  > RP object
  > END-POINT object
  > LSP object
  > LSPA object
  > BANDWIDTH object
  > METRIC object
  > METRIC object
  > METRIC object
  v ASSOCIATION object
      Object Class: ASSOCIATION OBJECT (40)
      0001 .... = ASSOCIATION Object-Type: IPv4 (1)
    > .... 0010 = Object Header Flags: 0x2, Processing-Rule (P)
      Object Length: 24
      Reserved: 0x0000
    > Flags: 0x0000
      Association Type: Disjoint Association (2)
      Association ID: 2
      IPv4 Association Source: 192.0.2.1
    v DISJOINTNESS-CONFIGURATION
        Type: DISJOINTNESS-CONFIGURATION (46)
        Length: 4
        Data: 00000002
```

7 BGP SR Policy

At the time of writing 7250 IXR generation one and generation two platforms provide support for SR Policy, but only support a single segment list per policy. Please check Release Notes for an updated list of 7250 IXR unsupported features.

The SR Policy Architecture [34] is a generic framework that describes the procedures and processes that a headend node should carry out when instantiating such a path. An SR Policy consists of an ordered list of segments on a node, sufficient to implement a traffic engineered path through the network. The segments can contain any type of Segment Identifier (SID), including Adjacency-SIDs, Node-SIDs, Anycast-SIDs, and Binding SIDs. The headend can then steer traffic into the SR Policy as appropriate, and SR Policy has inherent mechanisms allowing for traffic to be dynamically associated with a given policy.

An SR Policy can have one or multiple candidate paths, and SR-OS provides support for both. When explicit candidate paths are used, each path will contain one or more Segment-Lists, where each Segment-List contains the ordered set of SIDs required to provide the source-routed path from headend to destination. When a candidate path contains multiple Segment-Lists, each is assigned a weight for the purpose of weighted load-balancing. Candidate paths can be instantiated through a variety of mechanisms, including PCEP, BGP, or static local configuration. SR-OS provides support for instantiating SR Policies using static local configuration and through BGP. Both are described in this chapter which will start with a brief overview of SR Policy. The term BGP SR Policy is interchangeably used with BGP SR TE Policy.

SR Policy Overview

An SR Policy is identified through the tuple {headend, color, endpoint}. The headend is the node where the SR policy is instantiated, and the node that will be responsible for steering traffic into the policy with the relevant SID stack. From the perspective of the headend, the SR Policy can be identified using the {color, endpoint} tuple.

Color

Each SR Policy is associated with a color and it is a fundamental part of the policy. It is a 32-bit numerical value that the headend uses to associate the SR Policy with a characteristic, such as low-latency or high-throughput, for the purpose of traffic steering. Color is also a 32-bit transitive extended community [36] that can be attached to a given BGP Update message in order to associate itself with a corresponding SR Policy. For example, if headend H learns a BGP route R with {next-hop N, color extended community C, and VPN label V} and headend H has a valid SR Policy P to {endpoint N, color C}, it can associate BGP route R with the SR Policy P. When H receives packets with a destination matching R, it forwards them using the instructions contained within SR Policy P.

Endpoint

The endpoint is the destination of the policy specified as an IPv4 or IPv6 address, although 'wildcard' destinations can be used and are discussed later in this chapter.

Discriminator

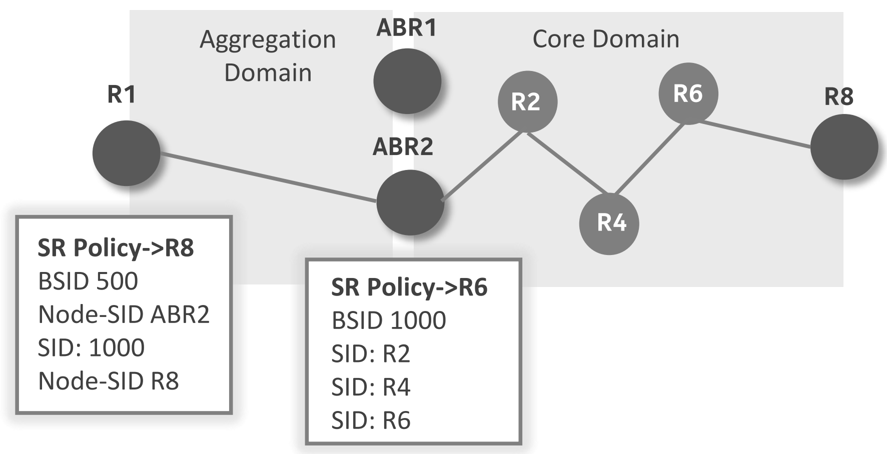
Each candidate path of an SR Policy is associated with a 32-bit discriminator value to uniquely identify it within the context of that SR Policy. When static SR Policies are used the discriminator has local significance only. When BGP is used to advertise SR Policies [35] a *distinguisher* field forms part of the NLRI, and this serves as the discriminator value. Because it forms part of the NLRI its purpose is to make the candidate path unique from an NLRI perspective. BGP SR Policy uses standard BGP propagation and best-path selection, hence a unique distinguisher ensures that best-path selection does not unnecessarily suppress SR Policy advertisements (at a Route-Reflector for example). Multiple candidate paths can exist for a given SR Policy, although only one path can be selected as the best path of the policy and become the active path. If several candidate paths of the same SR Policy {endpoint, color} are advertised via BGP to a headend, unique distinguishers for each NLRI are recommended.

Binding SID (BSID)

The Segment Routing architecture defines the use of a Binding Segment, or BSID. A BSID is bound to an SR Policy, and packets arriving at a node with an active label equal to the BSID are steered into that SR Policy. This action may well mean swapping the incoming active label with a number of outgoing labels representing the SR Policy path. When used in this manner, the Binding SID serves as an anchor point, sometimes referred to as a '*BSID anchor*' that allows one domain to be isolated from another domain. This is illustrated in *Figure 7-1*, where ABR2 is acting as a BSID anchor between the aggregation domain and the core domain. ABR2 has an SR Policy to R8 with the path R2-R4-R6 and with a BSID of 1000. R1's resulting SR Policy to R8 consists of the path {Node-SID ABR2, 1000, Node-SID R8}. When a packet is forwarded into this policy and arrives at ABR2, it will pop the Node-SID ABR2, and swap (BSID) label 1000 for the label stack {R2, R4, R6}.

Note that if ABR2 had an SR policy that included the path R2-R4-R6 and the Node-SID of R8, the label stack in R1's resulting SR policy could be even further reduced to {Node-SID ABR2, 1000}.

Figure 7-1: Binding SID (BSID) Anchor



The BSID serves as an anchor point which allows one domain to be isolated from the churn of another domain. If something changes in the path R2-R4-R6, ABR2 can repair the path locally without needing to change the BSID value known at R1. R1 is therefore protected from the churn in the core domain. The BSID also serves to reduce the number of segments/labels that the headend needs to impose to create an end-to-end traffic engineered path.

Segment List

The Segment List encodes a single path towards the endpoint. Multiple Segment Lists may be included in each SR Policy candidate path. Each Segment List may contain multiple Segments and may carry a Weight sub-TLV. Multiple Segments can be concatenated together within a Segment List to constitute an end-to-end path of the SR Policy. There are several types of the Segment that allow it to be encoded as variants of IPv4/IPv6 node address or local/remote address, and with a SID in the form of an MPLS label or IPv6 address. This chapter focuses only on the Type A encoding, which is represented as a SID in the form of an MPLS label and is currently the only encoding supported by SR-OS. The SID contained within each Segment can be any form of SID, including Node-SID, Adjacency-SID, Anycast-SID, or Binding SID.

The optional Weight field within a Segment List is used to implement (weighted) load-balancing in the presence of multiple Segment Lists. By default, SR-OS assigns a weight value of 1 to each Segment List and will consequently treat multiple Segment Lists as equal-cost by default.

Preference

The preference of a candidate path is used to select the best candidate path for an SR Policy in relation to other candidate paths. Multiple candidate paths can exist in an SR Policy, but only one candidate path can be selected as the best and active path. When multiple candidate paths exist that are considered valid, the candidate path with the highest preference will be selected. The default value of the preference is 100. If multiple candidate paths have the same preference, the protocol origin (PCEP, BGP SR Policy, static local configuration) may be considered, followed by the lower value of originator, followed by the higher value of discriminator. The default values for protocol origin are 30 for static SR policies, 20 for BGP SR policies, and 10 for PCEP signalled SR policies, with the highest value being preferred.

Advertising SR TE Policy in BGP

To allow a candidate path for an SR Policy to be advertised in BGP the 'SR TE Policy' (SAFI 73) is defined [35] and is carried in an update message using BGP multiprotocol extensions. The AFI must be IPv4 or IPv6. An SR Policy candidate path may be advertised from a centralised controller, or it may simply be advertised by a router; for example, an egress router advertising one or more paths towards itself. The structure of the SR TE Policy NLRI is shown in *Figure 7-2* and encodes the parameters of the SR Policy that have been previously outlined.

Figure 7-2: SR TE Policy NLRI

Path Attribute:	MP_REACH_NLRI <Distinguisher, Policy Color, Endpoint>		
Path Attribute:	Origin, AS_PATH, Local_PREF, etc		
Path Attribute:	Tunnel_Encapsulation_Attribute: Tunnel Type SR Policy		
	Sub-TLV:	Binding SID	
	Sub-TLV:	Preference	
	Sub-TLV:	Segment List	
		Sub-TLV:	Segment
		Sub-TLV:	Segment

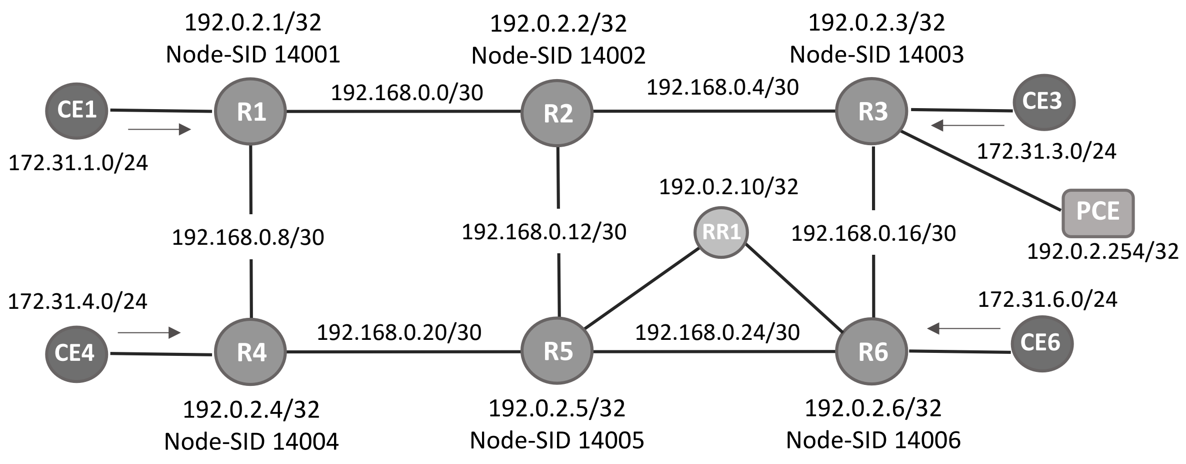
The SR TE Policy NLRI is used to identify an SR Policy candidate path, and as it uses Multi-Protocol BGP it is carried in an MP_REACH/UNREACH_NLRI path attribute. The NLRI contains the color, endpoint, and distinguisher values described above.

The other parameters of the SR Policy candidate path such as BSID, Segment List, and Preference are carried as sub-TLVs of the Tunnel Encapsulation Attribute [36] using a new Tunnel-Type known as 'SR Policy'.

Test Topology

The generic network topology in *Figure 7-3* is used to illustrate the use of both BGP SR policy and static SR policy. Routers R1 to R6 are in a flat IS-IS Level 2 domain and the relevant system addresses are shown. SR is enabled with an SRGB of 12000-19999, and Node-SIDs are shown in the figure. A Route-Reflector is used to simplify propagation of SR Policies when BGP is used, and to that extent Routers R1 to R6 and the PCE are configured as clients of the Route-Reflector for numerous Address Families including the IPv4 SR Policy Address Family. When sourcing SR Policies in BGP they will interchangeably be advertised by routers and/or the PCE. Finally, a number of CE routers are introduced and each of them advertises a single IPv4 prefix to its connected (PE) router. These will be used for the purpose of illustrating traffic steering with SR Policy.

Figure 7-3: Test Topology for SR Policy



SR Policy Local Block

A Binding SID may be allocated either a local or a global SID. In SR-OS they are always are allocated a local SID, in which case the BSID needs to be within the range of a locally-configured Segment Routing Local Block (SRLB). In SR-MPLS SRLBs are reserved label blocks used for specific local purposes such as SR Policy/BSID, Adjacency Set SIDs, and assignment of static Adjacency SIDs. A dedicated SRLB is required per-application and has local significance only. Since they have local significance, the same values could be used on all SR routers in the domain for operational ease. Ranges for each SRLB are taken from the dynamic label range. As the SRGB in the topology is {12000,19999} the SR Local Block {20000, 23999} is allocated to the use of SR Policy.

Initially the SR Local Block is created using the **reserved-label-block** command within the **mpls-labels** context. Within the context of that local block, the **start-label** and **end-label** define the range of the block. Once the SRLB is defined it is dedicated to the specific application, which in this case is SR Policies. After the **reserved-label-block** has been assigned, **sr-policies** must then put into an **admin-state** of **enable**.

Output 7-1: Creation of SR Local Block for SR Policy

```
    router "Base" {
        mpls-labels {
            reserved-label-block "sr-policy" {
                start-label 20000
                end-label 23999
            }
        }
        segment-routing {
            sr-policies {
                admin-state enable
                reserved-label-block "sr-policy"
            }
            }
        }
```

Static SR Policy

As previously described, SR Policies can be applied through static configuration or they can be learned through BGP. In this section the necessary steps are shown for instantiation of an SR policy with static configuration using router R1 as a headend.

Output 7-2 shows the configuration of a static SR policy at R1 (192.0.2.1) with an endpoint of R3 (192.0.2.3).

Output 7-2: Static SR Policy from R1 to R3

```
        segment-routing {
            sr-policies {
                static-policy "r1-to-r5-color100" {
                    admin-state enable
                    color 100
                    endpoint 192.0.2.3
                    head-end local
                    binding-sid 20001
                    distinguisher 12345
                    segment-list 1 {
                        admin-state enable
                        segment 1 {
                            mpls-label 524284
                        }
                        segment 2 {
                            mpls-label 524282
                        }
                        segment 3 {
                            mpls-label 14003
                        }
                    }
                }
            }
        }
```

The static SR policy is initially created within the **sr-policies** context and begins by assigning a **binding-sid** of 20001. In this example the SR Policy is local to R1, and the BSID value is therefore within the range of R1's SRLB allocated to SR Policies. If this static policy were to be advertised into BGP, the advertised BSID value must be in the range of the SRLB configured on the target headend. The policy **color** is 100 and is a 4-octet value that can be configured in the range 1-4294967295. The **distinguisher** represents the discriminator value and is also a 4-octet value with the same range and is configured simply as 12345. As previously described, the purpose of the discriminator value is to uniquely identify a candidate path within the context of an SR Policy, and when BGP is used to advertise SR Policies the NLRI uses a distinguisher value that serves as the discriminator. SR-OS therefore uses the command **distinguisher** as a generic term to mean both discriminator and distinguisher.

The **endpoint** is the IPv4 or IPv6 address of the destination for the policy and is configured as R3's address 192.0.2.3. There are special circumstances where the value 0.0.0.0 or 0::0 is allowed as an endpoint. This is referred to as *color-only steering* and is described later in the chapter. The **head-end** is the target node where the SR Policy is to be instantiated. If the SR Policy is statically configured on the headend itself for forwarding of traffic locally into that policy, the value **local** should be used as shown in this example. If the SR Policy is configured somewhere other than the headend for the purpose of advertising into BGP towards the headend, the value of the **head-end** parameter should be the IPv4 address of that headend. When the SR Policy is advertised in BGP, the headend address is also encoded as an IPv4 address specific Route-Target Extended Community, which allows for potential constraining of route propagation.

The final parameter is the **segment-list**. The configuration output above shows the segment list consisting of three segments which represent R1's Adj-SID towards R4, R4's Adj-SID towards R5, followed by R3's Node-SID. There is no particular rationale for using this SID stack, but it serves to illustrate that both Adj-SIDs and Node-SIDs with strict or loose hops can be used. The segment list has an optional weight parameter used for load-balancing across multiple segment lists. In this example only a single segment list exists, hence the default weight value of 1 is retained.

Finally, both the segment list and the static-policy are put into an **admin-state** of **enable**. The following output shows the operational state of the static SR policy. The `active` field shows whether this candidate path is the selected path in the presence of multiple candidate paths. The SR Policy segment list is considered valid if the headend is able to perform path resolution for the first SID in the segment list into one or more outgoing interfaces and next-hops. The segment 1 label is 524284, and the state is shown as `resolved-up`, indicating that this is a valid segment list.

Output 7-3: Operational State of Static SR Policy to R3

```
A:admin@R1# show router segment-routing sr-policies static end-point 192.0.2.3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active           : Yes                    Owner           : static
Color            : 100
Head             : 0.0.0.0                Endpoint Addr   : 192.0.2.3
RD               : 12345                  Preference      : 100
BSID             : 20001
TunnelId         : 917518                 Age             : 1586
Origin ASN       : 0                      Origin          : 0.0.0.0
NumReEval        : 0                      ReEvalReason    : none
NumActPathChange: 0                       Last Change     : 11/18/2021 08:40:27
Maintenance Policy: N/A

Path Segment Lists:
Segment-List     : 1                      Weight          : 1
S-BFD State      : Down                   S-BFD Transitio*: 0
Num Segments     : 3                      Last Change     : 11/06/2021 11:37:33
  Seg 1 Label    : 524284                 State           : resolved-up
  Seg 2 Label    : 524282                 State           : N/A
  Seg 3 Label    : 14003                  State           : N/A

===============================================================================
* indicates that the corresponding row element may have been truncated.
```

If the policy is considered valid, it is populated in the tunnel-table with an owner of `sr-policy`. The entry indicates the destination and color and always has a metric value of 0 regardless of how the SR Policy is instantiated. The 0 metric is used simply because there is no effective way for the headend to determine a more reflective value for an SR Policy when learnt through BGP or statically configured. The preference value assigned to SR Policy is 14 and is currently non-configurable.

Output 7-4: Static SR Policy Tunnel-Table Entry

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol sr-policy

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-policy MPLS  917518    14     192.168.0.10   0
   100
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

Traffic Steering into SR Policies

A headend can potentially steer traffic into an SR policy as a midpoint (or BSID anchor) or as an ingress router using color-based traffic steering:

- At a midpoint or BSID anchor, if an incoming packet has an active label that matches the BSID of a valid policy, the incoming label is swapped for the label(s) contained in the active path of that policy and traffic is forwarded along that path.
- At an ingress router, if a BGP or service route is received containing a Color Extended Community with a value corresponding to a valid local SR policy, and the endpoint of that policy matches the next-hop of the BGP/service route, traffic is forwarded into the associated policy.

This sub-section describes the use of the Color Extended Community to implement traffic steering at an ingress router and begins with an overview of the structure of the Color Extended Community.

The Color Extended Community has two flags known as the Color-Only (CO) bits that allow for a headend to optionally steer traffic into an SR policy without the need to explicitly define a policy endpoint that matches the next-hop of a BGP or service route. In this case the endpoint can be the null address (0.0.0.0 for IPv4 and ::0 for IPv6) and traffic is steered into a policy based purely on correlation of color. *Table 7-1* details the destination steering options based on the setting of the CO-bits.

Table 7-1: Use of Color-Only (CO) Bits

CO-bits=00	CO-bits=01	CO-bits=10

If there is a valid SR policy (N, C) where N is the IPv4 or IPv6 endpoint address and C is a color, steer into SR policy (N, C);	If there is a valid SR policy (N, C) where N is the IPv4 or IPv6 endpoint address and C is a color, steer into SR policy (N, C);	If there is a valid SR policy (N, C) where N is the IPv4 or IPv6 endpoint address and C is a color, steer into SR policy (N, C);
Else, steer on the IGP path to the next-hop N	Else if there is a valid SR policy (null endpoint, C) of the same Address-Family as N, steer into SR policy (null endpoint, C);	Else if there is a valid SR policy (null endpoint, C) of the same Address-Family as N, steer into SR policy (null endpoint, C);
	Else if there is any valid SR policy (any Address-Family null endpoint, C), steer into SR policy (any null endpoint, C);	Else if there is any valid SR policy (any Address-Family null endpoint, C), steer into SR policy (any null endpoint, C);
	Else, steer on the IGP path to the next-hop N	Else if there is any valid SR policy (any endpoint, C) of the same Address Family as N, steer into SR policy (any endpoint, C);
		Else if there is any valid SR policy (any Address-Family endpoint, C), steer into SR policy (any Address-Family endpoint, C);
		Else, steer on the IGP path to the next-hop N

Per-Destination Traffic Steering

When incoming packets match a BGP/service route with a next-hop that resolves to an SR Policy, it is referred to as per-destination traffic steering. The previously configured static policy at R1 with Color 100 is used to illustrate how it is applied.

VPRN '*one*' is configured on routers R1, R3, R4, and R6 with import/export Route-Target 64496:1 to connect the CE routers shown in *Figure 7-3*. Since there is currently only a single static SR Policy to R3 (endpoint 192.0.2.3) this example shows how it is used by R1 to resolve the prefix 172.31.3.0/24 advertised by CE3 and consequently R3.

Output 7-5 shows the **auto-bind-tunnel** at R1 with the **resolution-filter** set to **sr-policy**. Chapter 8 fully describes how SR tunnels in general are applied to services and covers the use of **auto-bind-tunnel** in more detail, hence this is somewhat pre-empted here but necessary for the purpose of illustration.

Output 7-5: Auto-Bind-Tunnel Resolution-Filter at R1

```
    service {
        vprn "one" {
            bgp-ipvpn {
                mpls {
                    auto-bind-tunnel {
                        resolution filter
                        resolution-filter {
                            sr-policy true
                        }
                    }
                }
            }
        }
    }
```

CE3 is locally connected to R3 and advertises prefix 172.31.3.0/24 into IPv4 BGP, which R3 subsequently advertises as a VPN-IPv4 route. In addition to attaching the Route-Target Extended Community to the VPN-IPv4 route, R3 also attaches a Color Extended Community with color value 100 to match that of the SR Policy at R1. R3's VRF export policy is shown in *Output 7-6*. Note that when configuring the Color Extended Community the syntax "*color:co:value*" is used. Therefore, in the example configuration the CO bits are 00 and the color value is 100.

Output 7-6: VRF-Export Policy at R3

```
    policy-options {
        community "color-100" {
            member "color:00:100" { }
        }
        community "vrf-one" {
            member "target:64496:1" { }
        }
        policy-statement "vrf-one-export" {
            entry 10 {
                from {
                    protocol {
                        name [bgp]
                    }
                }
                to {
                    protocol {
                        name [bgp-vpn]
                    }
                }
                action {
                    action-type accept
                    community {
                        add ["vrf-one" "color-100"]
                    }
                }
            }
        }
    }
```

At R1 the VPN-IPv4 route with {next-hop R3, color extended community 100} resolves to R1's static SR policy with {endpoint R3, color 100}. The VPN-IPv4 route is therefore imported into the VPRN route-table with an indication that it is resolved to the SR Policy with tunnel ID 917518, which is the tunnel ID of the previously configured static SR policy. Traffic from R1 to VPN-IPv4 prefix 172.31.3.0/24 will therefore be forwarded into the SR Policy using the label stack defined in the segment list.

Output 7-7: VPRN 'one' Route-Table at R1

```
A:admin@R1# show router 1 route-table 172.31.3.0/24

===============================================================================
Route Table (Service: 1)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
172.31.3.0/24                                 Remote  BGP VPN   00h11m12s  170
       192.0.2.3 (tunneled:SR-Policy:917518)                        0
```

```
-------------------------------------------------------------------------------
No. of Routes: 1
```

Color-Only Traffic Steering

SR-OS provides support for Color-Only traffic steering using a null endpoint SR policy, but it should be understood that its use is limited only to unlabelled BGP address families. The reason for this is simply that when an egress router advertises a downstream label in a labeled BGP Update (i.e., VPN-IPv4/IPv6, EVPN, BGP Labeled Unicast etc) that egress router needs to see that label in received packets in order to be able to demultiplex into the relevant service/next-hop and forward traffic towards the destination. If a headend is forwarding traffic into an SR Policy with a null endpoint, that headend is unaware of the egress router and therefore cannot impose the relevant downstream-advertised BGP/service label into the label stack. For that reason, Color-Only traffic steering can only be applied to unlabelled BGP Updates.

To illustrate the use of Color-Only traffic steering, R3 advertises an IPv4 prefix 172.16.3.1/32 to R1 with the Color Extended Community 01:100. The intention at R1 is to use the previously configured static SR policy to resolve this route. As shown in *Table 7-1*, with the CO-bits set to 01 the headend will use an SR Policy with (null endpoint, C) if no valid (N, C) SR Policy exists.

Output 7-8: IPv4 Prefix Advertised by R3

```
*A:R3# show router bgp routes 172.16.3.1/32 hunt | match expression
"Network|Nexthop|Community"
Network         : 172.16.3.1/32
Nexthop         : 192.0.2.3
Res. Nexthop    : n/a
Community       : color:01:100
```

At R1 the static SR policy to R3 configured in *Output 7-2* is reconfigured such that the endpoint is no longer an explicit endpoint of 192.0.2.3 (R3) but instead uses a null endpoint (0.0.0.0).

Output 7-9: Static SR Policy with Null Endpoint

```
        segment-routing {
            sr-policies {
                static-policy "r1-to-r3-color100" {
                    endpoint 0.0.0.0
                    }
                }
            }
        }
```

Finally, since R3 advertised an IPv4 BGP prefix, R1 also enables the use of BGP shortcuts at global level, with a **resolution-filter** that only permits only the use of SR Policy.

Output 7-10: BGP Shortcuts with SR Policy Resolution

```
        bgp {
            next-hop-resolution {
                shortcut-tunnel {
                    family ipv4 {
                        resolution filter
                        resolution-filter {
                            sr-policy true
                        }
                    }
                }
            }
        }
```

The tunnel-table of R1 shows that there is a single SR Policy active with a destination of 0.0.0.0/32 (null) and color 100.

Output 7-11: R1 SR Policy Tunnel-Table Entry

```
A:admin@R1# show router tunnel-table protocol sr-policy

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
0.0.0.0/32            sr-policy MPLS  917519    14     192.168.0.10   0
   100
-------------------------------------------------------------------------------
```

The status of the IPv4 prefix 172.16.3.1/32 received from R3 can be observed in the following output at R1. The output shows that the route is Used/Valid/Best, and that the resolving protocol is SR Policy, whilst the resolving next-hop is 0.0.0.0. Hence a BGP next-hop has been resolved to a null endpoint SR Policy using the CO-bits.

Output 7-12: IPv4 Prefix Resolved to SR Policy with Null Endpoint

```
A:admin@R1# show router bgp routes 172.16.3.1/32 detail
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP IPv4 Routes
===============================================================================
Original Attributes

Network         : 172.16.3.1/32
Nexthop         : 192.0.2.3
Path Id         : None
From            : 192.0.2.10
Res. Protocol   : SR-POLICY              Res. Metric    : 0
Res. Nexthop    : 0.0.0.0 (SR-POLICY)
Local Pref.     : 100                    Interface Name : NotAvailable
Aggregator AS   : None                   Aggregator     : None
Atomic Aggr.    : Not Atomic             MED            : None
AIGP Metric     : None                   IGP Cost       : 0
Connector       : None
Community       : color:01:100
Cluster         : 192.0.2.10
Originator Id   : 192.0.2.3              Peer Router Id : 192.0.2.10
```

```
Fwd Class      : None                   Priority       : None
Flags          : Used Valid Best IGP Group-Best In-RTM
Route Source   : Internal
AS-Path        : No As-Path
Route Tag      : 0
Neighbor-AS    : n/a
Orig Validation: NotFound
Source Class   : 0
 ---[ snip ]---
```

BGP SR Policy

BGP SR Policies can be advertised either from a centralised controller or from an SR-OS router; perhaps an egress router advertising a preferred downstream path to one or more ingress routers. Although the PCE in the test topology is capable of sourcing BGP SR Policies, this section will source the BGP SR Policy from a router to demonstrate how this is done.

When an SR-OS router advertises SR Policies into BGP they must first be statically configured in order to provide the relevant information to populate the BGP path attributes. To that end, a static policy is configured at router R2 creating an SR Policy from R4 to R6. The policy has a **head-end** of R4 (192.0.2.4), an **endpoint** of R6 (192.0.2.6), and a **color** of 200. The **binding-sid** is 200001 to fit within the range of the SRLB configured at R4, and a **distinguisher** value of 12345. If a second candidate path were to be advertised for this SR Policy a different distinguisher value would be required to avoid route suppression at the Route-Reflector. The **segment-list** contains four segments representing a path routes R4-R1-R2-R3-R6 and contains strict hops with Adj-SIDs.

Output 7-13: Configuration at R2 for R4-to-R6 SR Policy

```
        segment-routing {
            sr-policies {
                static-policy "r4-to-r6-color200" {
                    admin-state enable
                    color 200
                    endpoint 192.0.2.6
                    head-end 192.0.2.4
                    binding-sid 20001
                    distinguisher 12345
                    segment-list 1 {
                        admin-state enable
                        segment 1 {
                            mpls-label 524286
                        }
                        segment 2 {
                            mpls-label 524286
                        }
                        segment 3 {
                            mpls-label 524281
                        }
                        segment 4 {
                            mpls-label 524281
                        }
                    }
                }
            }
        }
```

To advertise the static policy into BGP, two steps are required at R2. Firstly, the command **sr-policy-import true** must be configured within the global BGP context. This command instructs BGP to import all statically configured non-local SR Policies from the SR database into the BGP RIB such that they can be advertised towards BGP peers supporting the SR TE Policy Address Family. Secondly, the BGP peering with the Route-Reflector RR1 (192.0.2.10) must have the **sr-policy-ipv4 true** command enabled within the **family** context to be able to support the SR TE Address Family. (Although not shown, the **sr-policy-ipv4** address family is enabled on all routers for the RR to peer to all of its client and reflect SR TE Policies.)

Output 7-14: BGP Configuration at R2 for Advertising SR Policies

```
        bgp {
            sr-policy-import true
            group "IBGP-Core" {
                family {
                    sr-policy-ipv4 true
                }
            }
            neighbor "192.0.2.10" {
                group "IBGP-Core"
            }
        }
```

Note: SR-OS currently supports propagation of SR Policy routes across internal BGP peers. SR Policy routes are not currently advertised to external BGP peers.

Debug 7-1 shows the BGP Update for the SR Policy advertised towards RR1 (192.0.2.10). The address family is SR_POLICY_IPv4 (SAFI 73), and the NLRI contains the distinguisher (shown as `RD`), color, and endpoint. Note the presence of an IPv4 address specific Route-Target Extended Community encoding the headend R4's system address (192.0.2.4), which as previously described allows for potential constraining of route propagation as required. Finally, the SR Policy attributes such as Preference, BSID, and Segment List are encoded as sub-TLVs of the Tunnel-Encapsulation Attribute.

Debug 7-1: SR Policy Update Advertised by R2

```
1 2021/11/18 14:25:06.273 GMT minor: DEBUG #2001 Base Peer 1: 192.0.2.10
Peer 1: 192.0.2.10: UPDATE
Peer 1: 192.0.2.10 - Send BGP UPDATE:
    Withdrawn Length = 0
    Total Path Attr Length = 118
    Flag: 0x90 Type: 14 Len: 22 Multiprotocol Reachable NLRI:
        Address Family SR_POLICY_IPV4
        NextHop len 4 NextHop 192.0.2.2
        [Len 96] RD: 12345 Color: 200 Endpoint: 192.0.2.6
    Flag: 0x40 Type: 1 Len: 1 Origin: 0
    Flag: 0x40 Type: 2 Len: 0 AS Path:
    Flag: 0x40 Type: 5 Len: 4 Local Preference: 100
    Flag: 0xc0 Type: 16 Len: 8 Extended Community:
        target:192.0.2.4:0
    Flag: 0xc0 Type: 23 Len: 64 Tunnel-Encapsulation-attr:
       Tunnel-Encap-Type:- sr-policy
       Preference SubTlv:-
          flags: 0x0  value: 100
       Binding Sid SubTlv:-
```

```
           len: 6  flags: 0x0  Label: 20001
        Segment List 0 SubTlv:- len: 41 Weight: 1
           Segment 0 -Type 1 Flags 0x0 Label: 524286 TC:0 S:0 ttl:255
           Segment 1 -Type 1 Flags 0x0 Label: 524286 TC:0 S:0 ttl:255
           Segment 2 -Type 1 Flags 0x0 Label: 524281 TC:0 S:0 ttl:255
           Segment 3 -Type 1 Flags 0x0 Label: 524281 TC:0 S:0 ttl:255
```

Note: Although SR-OS provides support for receiving multiple candidate paths for the same SR Policy, it does not currently support advertisement of multiple candidate paths for the same SR Policy.

When a BGP SR Policy Update is received by an SR-OS router, it is believed to be targeted at a given router as the headend if either:

– The Update has no Route-Target Extended Community and a NO-ADVERTISE standard community
– The Update has an IPv4 address-specific Route-Target Extended Community with an IPv4 address matching the system address of this router.

When R4 receives the BGP SR Policy Update, the second bullet is applicable and as such it interprets the route as targeted towards it as the headend. The following output is taken at R4 and shows that the SR Policy is active and as the first SID in the segment list has been correctly resolved. Note also that the owner is BGP.

Output 7-15: R4-to-R6 SR Policy at R4

```
A:admin@R4# show router segment-routing sr-policies bgp color 200 end-point 192.0.2.6

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active           : Yes                   Owner            : bgp
Color            : 200
Head             : 0.0.0.0               Endpoint Addr    : 192.0.2.6
RD               : 12345                 Preference       : 100
BSID             : 20001
TunnelId         : 917508                Age              : 610
Origin ASN       : 64496                 Origin           : 192.0.2.2
NumReEval        : 0                     ReEvalReason     : none
NumActPathChange: 0                      Last Change      : 11/18/2021 14:25:07
Maintenance Policy: N/A

Path Segment Lists:
Segment-List     : 1                     Weight           : 1
S-BFD State      : Down                  S-BFD Transitio*: 0
Num Segments     : 4                     Last Change      : 11/17/2021 16:43:22
  Seg 1 Label    : 524286                State            : resolved-up
  Seg 2 Label    : 524286                State            : N/A
  Seg 3 Label    : 524281                State            : N/A
  Seg 4 Label    : 524281                State            : N/A
```

Verification is also made at R4 that the SR Policy is correctly populated in the tunnel-table. The destination and color are shown. The non-configurable preference is 14, and the metric is again 0 as previously discussed.

Output 7-16: SR Policy Tunnel-Table Entry at R4

```
A:admin@R4# show router tunnel-table 192.0.2.6 protocol sr-policy
```

```
===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.6/32          sr-policy MPLS  917508    14     192.168.0.9    0
   200
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

The procedure for traffic steering into an SR Policy learnt through BGP is exactly the same as traffic steering into a static SR Policy and is therefore not repeated here.

BSID Anchor

The statically configured and BGP advertised SR Policies used so far in this chapter have been instantiated on a headend that uses the Color Extended Community to steer traffic into the policy. An alternative method of steering traffic into a policy is through use of the BSID. If an incoming packet has an active label that matches the BSID of a valid SR policy, the packet is forwarded into that policy and the incoming label is swapped for the label(s) that policy contains. Using a BSID in this way is considered useful at domain interconnects such as ABRs or ASBRs and was discussed earlier in this chapter. It provides opacity between the domains and protects the churn from one domain from rippling to another domain. It's use is not constrained to domain interconnects though. In large networks it has the additional benefit of reducing the number of labels an ingress router needs to impose, simply because the BSID can expand a single incoming SID/label stack (the BSID) into a much larger outgoing SID/label stack. This is particularly useful when SR TE is used within access and aggregation domains where the hardware capabilities of deployed devices is not likely to support the imposition of a large number of labels.

The test topology in *Figure 7-3* is entirely IS-IS Level 2 and therefore not constructed of multiple domains, however, it is still sufficient to illustrate the use of BSID traffic steering. In the following example R2 will become a BSID anchor for an SR Policy path extended between R1 and R3. This requires the instantiation of two SR Policies:

a) An SR Policy at R2 with a segment list that constructs the required path to R3. Like every SR Policy it requires a BSID, but in this case the BSID is programmed in the Incoming Label Map (ILM) table and will have a Next-Hop Label Forwarding Entry (NHLFE) that includes the segments (labels) in the segment list. This SR Policy will route via R2-R5-R6-R3 using strict hops with Adj-SIDs.
b) An SR Policy at R1 with a segment list specifying a path to R2, followed by a segment that references the relevant BSID programmed at R2 in a) above.

The following output shows the SR Policy advertised in BGP to R2. It uses color 300 and has an endpoint of R3 (192.0.2.3). Since traffic steering at R2 into the SR Policy is achieved using the BSID, the policy color is largely irrelevant and any color value could be used (although different colors may be needed to represent different path characteristics). Packets are classified upstream of R2 at R1, and the result of that classification selects the relevant BSID to meet the path requirements. The segment list programs a path to R3 using Adj-SIDs along the path R2-R5-R6-R3. Lastly, the BSID value is 23001.

Output 7-17: SR Policy to R3 at BSID Anchor R2

```
A:admin@R2# show router segment-routing sr-policies bgp color 300

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active          : Yes                   Owner           : bgp
Color           : 300
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
RD              : 23456                 Preference      : 100
BSID            : 23001
TunnelId        : 917530                Age             : 80
Origin ASN      : 64496                 Origin          : 192.0.2.4
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 11/18/2021 15:17:53
Maintenance Policy: N/A

Path Segment Lists:
Segment-List    : 1                     Weight          : 1
S-BFD State     : Down                  S-BFD Transitio*: 0
Num Segments    : 3                     Last Change     : 11/06/2021 11:39:20
  Seg 1 Label   : 524282                State           : resolved-up
  Seg 2 Label   : 524279                State           : N/A
  Seg 3 Label   : 524285                State           : N/A
```

Output 7-18 shows the SR Policy advertised in BGP to R1. It uses color 300 and also has an endpoint of R3 (192.0.2.3). The segment list programs a path that contains the Adj-SID of the R1-R2 link (524286) followed by the BSID of the SR Policy used at R2 for the R2-to-R3 SR Policy (23001). When R2 receives this label (either as the active label, or after it popped a label above it), it will swap label 23001 for the label stack contained in the SR Policy of that BSID.

Output 7-18: SR Policy at R1 to BSID Anchor R2

```
A:admin@R1# show router segment-routing sr-policies bgp color 300

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active          : Yes                   Owner           : bgp
Color           : 300
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
RD              : 34567                 Preference      : 100
BSID            : 20002
TunnelId        : 917520                Age             : 501
Origin ASN      : 64496                 Origin          : 192.0.2.4
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 11/18/2021 15:17:53
Maintenance Policy: N/A

Path Segment Lists:
Segment-List    : 1                     Weight          : 1
S-BFD State     : Down                  S-BFD Transitio*: 0
```

```
Num Segments     : 2                    Last Change      : 11/06/2021 11:37:33
  Seg 1 Label    : 524286               State            : resolved-up
  Seg 2 Label    : 23001                State            : N/A
```

Using a BSID anchor in this way means that R1 only needed to impose two labels to have a strict-routed traffic engineered path towards R3 that in total would have required four labels. Obviously for networks of larger span the advantages for label reduction are much greater.

To verify the forwarding path of the SR Policy from R1 to R3 through the BSID anchor at R2 an LSP ping is initiated at R1 through the policy.

Output 7-19: LSP Ping through R1 to R3 SR Policy

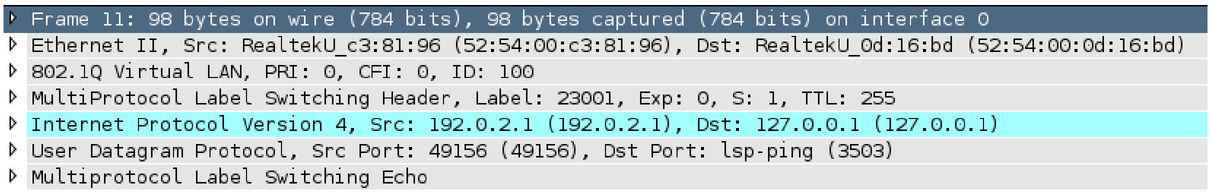
```
A:R1# oam lsp-ping sr-policy color 300 endpoint 192.0.2.3 send-count 5
LSP-PING color 300 endpoint 192.0.2.3: 76 bytes MPLS payload
Seq=1, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=4.08ms rc=3 (EgressRtr)
Seq=2, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=4.27ms rc=3 (EgressRtr)
Seq=3, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=4.33ms rc=3 (EgressRtr)
Seq=4, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=5.74ms rc=3 (EgressRtr)
Seq=5, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=4.41ms rc=3 (EgressRtr)
```

Recall that the SR Policy at R1 routes to R2 at which point the BSID 23001 is swapped for the SID stack of R2's Policy that routes R2-R5-R6-R3.

- R1's Policy consists of a segment list containing the Adj-SID of the R1-R2 link (524286) followed by the BSID of R2's Policy (23001).
- R2's Policy consists of a segment list containing the Adj-SID of the R2-R5 link (524282), followed by the Adj-SID of the R5-R6 link (524279), followed by the Adj-SID of the R6-R3 link (524285).

Figure 7-4 shows a packet capture taken on the R1-R2 link while the LSP ping is executed. As R1 does not impose the Adj-SID for the R1-R2 link the SID stack consists of a single label, 23001, representing the BSID of the R2-to-R3 SR Policy.

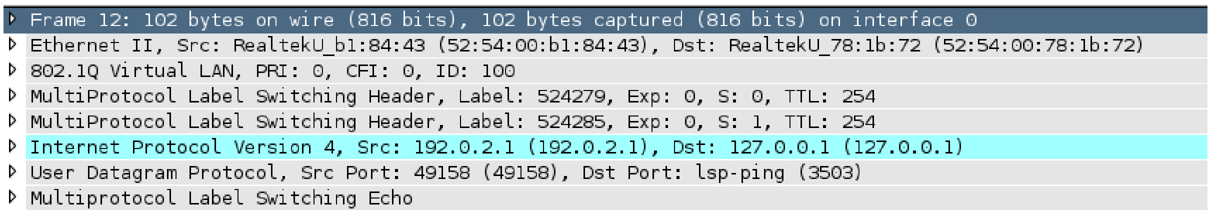
Figure 7-4: Packet Capture of LSP ping on R1-R2 Link



```
▷ Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▷ Ethernet II, Src: RealtekU_c3:81:96 (52:54:00:c3:81:96), Dst: RealtekU_0d:16:bd (52:54:00:0d:16:bd)
▷ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 100
▷ MultiProtocol Label Switching Header, Label: 23001, Exp: 0, S: 1, TTL: 255
▷ Internet Protocol Version 4, Src: 192.0.2.1 (192.0.2.1), Dst: 127.0.0.1 (127.0.0.1)
▷ User Datagram Protocol, Src Port: 49156 (49156), Dst Port: lsp-ping (3503)
▷ Multiprotocol Label Switching Echo
```

Figure 7-5 shows a packet capture taken on the R2-R5 link, again while the LSP ping is executed. As R2 does not impose the Adj-SID for the R2-R5 link, the SID stack consists of two labels, 524279 representing the Adj-SID for the R5-R6 link, followed by 524285 representing the Adj-SID for the R6-R3 link.

Figure 7-5: Packet Capture of LSP Ping on R2-R5 Link


```
▷ Frame 12: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
▷ Ethernet II, Src: RealtekU_b1:84:43 (52:54:00:b1:84:43), Dst: RealtekU_78:1b:72 (52:54:00:78:1b:72)
▷ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 100
▷ MultiProtocol Label Switching Header, Label: 524279, Exp: 0, S: 0, TTL: 254
▷ MultiProtocol Label Switching Header, Label: 524285, Exp: 0, S: 1, TTL: 254
▷ Internet Protocol Version 4, Src: 192.0.2.1 (192.0.2.1), Dst: 127.0.0.1 (127.0.0.1)
▷ User Datagram Protocol, Src Port: 49158 (49158), Dst Port: lsp-ping (3503)
▷ Multiprotocol Label Switching Echo
```

As the SR Policy at R1 has an endpoint of 192.0.2.3 (R3) it can be used to resolve BGP next-hops in the same way as any other SR Policy. CE3 is locally connected to R3 and advertises prefix 172.31.3.0/24 into IPv4 BGP, which R3 subsequently advertises to the members of VPRN 'one' as a VPN-IPv4 route. In addition to attaching the Route-Target Extended Community to the VPN-IPv4 route, R3 also attaches a Color Extended Community with color value 300 to match that of the SR Policy at R1. R1's **auto-bind-tunnel resolution-filter** is configured for **sr-policy**, and therefore the VPN-IPv4 route with {next-hop R3, color extended community 300} resolves to R1's static SR policy with {endpoint R3, color 300}. The VPN-IPv4 route is therefore imported into the VPRN route-table. Traffic from R1 to VPN-IPv4 prefix 172.31.3.0/24 will thereafter be forwarded into the SR Policy through the BSID anchor R2.

Output 7-20: VPRN 'one' Route-Table for CE3's Prefix

```
A:admin@R1# show router 1 route-table 172.31.3.0/24

===============================================================================
Route Table (Service: 1)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
172.31.3.0/24                                 Remote  BGP VPN   00h00m24s  170
       192.0.2.3 (tunneled:SR-Policy:917520)                        0
-------------------------------------------------------------------------------
No. of Routes: 1
Flags: n = Number of times nexthop is repeated
       B = BGP backup route available
       L = LFA nexthop available
       S = Sticky ECMP requested
```

Weighted Equal Cost Multipath (ECMP)

Weighted equal cost multipath (ECMP) is supported with SR Policies using multiple segment lists. Each segment list contains a path from the headend to the endpoint, and each segment list contains a weight used to influence ECMP forwarding. *Output 7-21* shows an SR Policy advertised by the PCE to R1. The SR Policy is destined for the endpoint R3 (192.0.2.3), has a color of 400, and has multiple segment lists. Segment-list 1 has a weight of 70 and contains a single segment of the Node-SID of R3 (14003). As such will follow the shortest path R1-R2-R3. Segment-list 2 has a weight of 30 and encodes a path consisting of Node-SIDs along the path R1-R4-R6-R3.

Output 7-21: SR Policy with Weighted ECMP Operational State

```
A:admin@R1# show router segment-routing sr-policies bgp color 400 end-point 192.0.2.3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active          : Yes                   Owner           : bgp
Color           : 400
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
RD              : 34567                 Preference      : 100
BSID            : 20004
TunnelId        : 917523                Age             : 21
Origin ASN      : 64496                 Origin          : 192.0.2.2
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 11/19/2021 09:11:32
Maintenance Policy: N/A

Path Segment Lists:
Segment-List    : 1                     Weight          : 70
S-BFD State     : Down                  S-BFD Transitio*: 0
Num Segments    : 1                     Last Change     : 11/06/2021 11:37:33
  Seg 1 Label   : 14003                 State           : resolved-up

Segment-List    : 2                     Weight          : 30
S-BFD State     : Down                  S-BFD Transitio*: 0
Num Segments    : 3                     Last Change     : 11/06/2021 11:37:33
  Seg 1 Label   : 14004                 State           : resolved-up
  Seg 2 Label   : 14006                 State           : N/A
  Seg 3 Label   : 14003                 State           : N/A
```

The tunnel-table at R1 shows two tunnels to R3 (192.0.2.3) owned by SR Policy with color 400. Both entries reference the same tunnel ID of 917524 with reconciles with the tunnel ID associated with the SR Policy, but the first entry has a next-hop of R3 (192.0.2.3) while the second entry has a next-hop of R4 (192.0.2.4).

Output 7-22: Tunnel Table Entries for SR Policy with Weighted ECMP

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol sr-policy

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-policy MPLS  917524    14     192.0.2.3      0
   400
192.0.2.3/32          sr-policy MPLS  917524    14     192.0.2.4      0
   400
-------------------------------------------------------------------------------
```

SR Policies provide an effective mechanism for instantiating traffic engineered SR tunnels that may be advertised into BGP from either a controller or a router. Segments of paths constructed using SR Policies can be loose or strict using any combination of SIDs. Although some would argue that using BSIDs breaks some of the objectives of SR because it creates midpoint state, they are extremely useful. They provide a mechanism to interconnect domains while retaining opacity and reduce the label stack imposition required at ingress routers. Another significant benefit of SR Policies is that it uses BGP to both advertise and instantiate the SR Policies, as well as providing a method of steering traffic into those SR Policies.

8 Using SR for BGP and Services

This chapter looks at how SR is applied to BGP prefixes and VPN services. It is a relatively short chapter, but one without which the document would seem somewhat incomplete.

As a general rule (there are minor exceptions such as static routes), applying SR tunnels to BGP and services can be split into two cases:

- BGP learnt prefixes: These can be automatically bound to an SR tunnel with additional controls to select which SR protocol to use if required.
- Non-BGP services: Services that for example are built through targeted LDP, cannot use a next-hop from a learnt prefix to automatically resolve to an SR tunnel and therefore require static assignment of the SR tunnel type to an SDP.

The binding of a received BGP prefix to an SR-based tunnel revolves around BGP next-hop resolution to a tunnel, and the mechanisms used to control how that next-hop is resolved. Labeled and unlabeled BGP prefixes allow for dynamic binding of a prefix to an SR tunnel using the **resolution filter** construct that allows for the selection of the preferred tunnel mechanism or filtering of allowed tunnel mechanisms. When the command **resolution any** is selected it means that all available tunnels in the tunnel-table are eligible for use for next-hop resolution, and the selection of the used tunnel is made based purely on the tunnel-table preference. When the command **resolution filter** is used, it allows for filtering of selected tunnel protocols, so that only those selected are eligible for use. If multiple tunnel types are selected, then the tunnel-table preference is again used to select the preferred tunnel. If **resolution** option is set to **none** the default binding is used, which varies depending on the address family in question.

Non-BGP services that use the SDP construct, such as Epipe (VPWS) pseudowires, or LDP-based VPLS pseudowires require static definition of the SR tunnel type within the parent SDP context. SR is equally applicable to these types of services, but the assignment of the SR tunnel type cannot be dynamic and requires explicit configuration.

This chapter will now look at examples of how BGP and services are configured to use SR tunnels, starting with next-hop resolution for BGP before moving to non-BGP-based services that bind to SR tunnels using the SDP construct.

Binding BGP Prefixes to SR Tunnels

Resolving IPv4 unlabeled BGP routes to SR tunnels requires **shortcut-tunnel** to be enabled within the **next-hop-resolution** context. The relevant address family is selected, which in this case uses IPv4 BGP prefixes with the **family ipv4** command. The **resolution** command can be set to **any**, **filter**, or **none**. If **none** is selected for unlabeled BGP routes, the next-hop is simply resolved using a route-table lookup. This example uses the **resolution filter** option and enters the **resolution-filter** context under which the relevant tunnelling mechanisms are entered. In this case, **sr-te** is selected.

Output 8-1: Next-Hop Resolution to SR for Unlabeled BGP Routes

```
    router "Base" {
        bgp {
            next-hop-resolution {
                shortcut-tunnel {
                    family ipv4 {
                        resolution filter
                        resolution-filter {
                            sr-te true
                        }
                    }
                }
            }
        }
    }
```

The FIB entry for an unlabeled BGP prefix with the above configuration is shown in *Output 8-2*. The next-hop for the prefix is the BGP next-hop 192.0.2.6 which is resolved to an SR-TE LSP with tunnel ID 655363. The same tunnel ID will be evident in the tunnel-table.

Output 8-2: Unlabeled BGP Prefix Resolved to SR-TE

```
A:admin@R1# show router fib 1 ip-prefix-prefix-length 172.31.6.0/24

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
172.31.6.0/24                                               BGP
  192.0.2.6 (Transport:SR-TE:655363)
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Labeled BGP prefixes are resolved using **labeled-routes** followed by a **transport-tunnel** context. The relevant address family is selected, which in this example uses labeled IPv4 prefixes with the **family label-ipv4** command, but other options include **label-ipv6** or **vpn** prefixes. The **resolution** command is set to **filter**, and if left at the default **none** will use LDP to resolve BGP prefixes. Within the **resolution-filter** context **sr-isis** is selected, and **ldp** is disabled.

Output 8-3: Next-Hop Resolution to SR for Labeled BGP Routes

```
    router "Base" {
        bgp {
            next-hop-resolution {
                labeled-routes {
                    transport-tunnel {
                        family label-ipv4 {
                            resolution filter
                            resolution-filter {
                                ldp false
                                sr-isis true
                            }
                        }
                    }
                }
            }
        }
    }
```

The FIB entry for a labeled BGP prefix with the above configuration is shown in *Output 8-4*. The next-hop for the prefix is the BGP next-hop 192.0.2.3 which is resolved to SR-ISIS with tunnel ID 524328.

Output 8-4: Labeled IPv4 Prefix Resolved to SR-ISIS

```
A:admin@R1# show router fib 1 ip-prefix-prefix-length 172.17.0.149/32

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
172.17.0.149/32                                             BGP_LABEL
  192.0.2.3 (Transport:SR-ISIS:524328)
-------------------------------------------------------------------------------
Total Entries : 1
```

Services such as VPRN and EVPN that use BGP for auto-discovery of VPN membership and to exchange IP/MAC reachability also use the **resolution filter** concept but within the service context it is accessible through the **mpls auto-bind-tunnel** context. In the following example of a VPRN service the **resolution-filter** is configured to allow both **sr-isis** and **sr-te**. If both tunnel types are present in the tunnel-table for a given next-hop, an SR-TE LSP will be used as it has a lower preference value than that of SR-ISIS. If the SR-TE LSP should fail for any reason (and is detected by the system as having failed), the SR-ISIS LSP will be used in its place and to that extent can be considered a fallback or failover action. If the **resolution** command is left at the default setting of **none** the only protocol enabled for next-hop resolution is BGP.

Output 8-5: EVPN and VPRN Services with auto-bind-tunnel

```
    service {
        vprn "one" {
            bgp-ipvpn {
                mpls {
                    auto-bind-tunnel {
                        resolution filter
                        resolution-filter {
                            sr-isis true
                            sr-te true
                        }
                    }
                }
            }
        }
    }
```

The FIB entry for a received VPN-IPv4 prefix 172.31.3.0/24 is shown in *Output 8-6*. The next-hop is 192.0.2.3, and the VPN-IPv4 prefix carries the service label 524284. The corresponding transport LSP is an SR-TE LSP with tunnel ID 655362.

Output 8-6: VPN-IPv4 Prefix Resolved to SR-TE

```
A:admin@R1# show router 1 fib 1 ip-prefix-prefix-length 172.31.3.0/24
```

```
===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
172.31.3.0/24                                               BGP_VPN
  192.0.2.3 (VPRN Label:524284 Transport:SR-TE:655362)
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Binding SDP-based Services to SR Tunnels

Layer 2 services that do not use BGP can bind to SR tunnels through explicit configuration within the corresponding SDP. Services supported using explicitly provisioned SDPs include Epipe and LDP-based VPLS. BGP-AD, BGP-VPLS, and BGP-VPWS services can also use explicitly provisioned SDPs when a **pw-template** referenced by that service is configured for **provisioned-sdp use** or **provisioned-sdp prefer**. In this case, when a valid Layer 2 NLRI is received from a peer, the system looks for explicitly-defined SDPs with a corresponding **far-end** IP address.

Output 8-7 shows an SDP with a **delivery-type** of **mpls** statically configured to reference an SR-TE LSP with the **lsp** command, where that LSP tailend is equivalent to the SDP **far-end** IP address. Other options for SDPs include **sr-isis** or **sr-ospf**, and when enabled the system will look through the tunnel-table for an SR-ISIS or SR-OSPF tunnel to the configured **far-end** address. Binding SDP-based services to SR Policies is not supported in SR-OS given the requirement to associate a Color Extended Community.

Output 8-7: Binding SR Tunnels to SDPs

```
    service {
        sdp 2003 {
            admin-state enable
            delivery-type mpls
            far-end {
                ip-address 192.0.2.3
            }
            lsp "R1-to-R3" { }
        }
    }
```

The operational state of the SDP and it's binding to SR tunnels is as follows and indicates that the SDP is operationally up and bound to SR-TE LSP 'R1-to-R3' which is also operationally up.

Output 8-8: SDP Bound to SR-TE LSP

```
A:admin@R1# show service sdp 2003 detail

===============================================================================
Service Destination Point (Sdp Id : 2003) Details
===============================================================================
-------------------------------------------------------------------------------
```

```
 Sdp Id 2003  -192.0.2.3
-------------------------------------------------------------------------------
Description           : (Not Specified)
SDP Id               : 2003                   SDP Source         : manual
Admin Path MTU       : 0                      Oper Path MTU      : 9174
Delivery             : MPLS
Far End              : 192.0.2.3              Tunnel Far End     : n/a
Oper Tunnel Far End  : 192.0.2.3
LSP Types            : SR-TE

Admin State          : Up                     Oper State         : Up
 --- [snip] ---
-------------------------------------------------------------------------------
Segment Routing
-------------------------------------------------------------------------------
ISIS                 : disabled
OSPF                 : disabled
TE-LSP               : enabled

===============================================================================
SR-TE LSPs
===============================================================================
Lsp                             Admin   Oper    Time Since
                                                Last Trans
-------------------------------------------------------------------------------
R1-to-R3                        Up      Up      03d21h20m
```

Binding Static Routes to SR tunnels

Packets can also be forwarded over an SR tunnel using a static-route with an indirect next-hop. In the following example the prefix 172.25.0.0/20 has an indirect next-hop of 192.0.2.6, which is an egress LSR of the SR domain. The **tunnel-next-hop** context then gives access to the **resolution filter** that has been previously described in this chapter. This example binds the static-route to an SR-TE LSP with the **lsp** command within the **sr-te** context. This is an optional step and if left unpopulated with LSP names the system will automatically select the tunnel to the indirect next-hop from the tunnel-table.

Output 8-9: Binding Static-Routes to SR Tunnels

```
    router "Base" {
        static-routes {
            route 172.25.0.0/20 route-type unicast {
                indirect 192.0.2.6 {
                    admin-state enable
                    tunnel-next-hop {
                        resolution filter
                        resolution-filter {
                            sr-te {
                                lsp "R1-to-R6" { }
                            }
                        }
                    }
                }
            }
        }
    }
```

The FIB entry for the static-route is shown in *Output 8-10*. The (indirect) next-hop is 192.0.2.6, and the corresponding transport LSP is an SR-TE LSP with tunnel ID 655363.

Output 8-10: FIB Entry for Static-Route

```
A:admin@R1# show router fib 1 ip-prefix-prefix-length 172.25.0.0/20

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
172.25.0.0/20                                               STATIC
  192.0.2.6 (Transport:SR-TE:655363)
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Tunnel-Table Preferences

Any resolved Node-SID, SR-TE LSP, or SR Policy is entered into the tunnel-table and is programmed into the data-path. The tunnel entries in the tunnel-table are made available to all users of it, including auto-bind-tunnel and explicit SDP bindings. Entries in the tunnel-table are assigned a preference, which is used to select the preferred tunnel in the presence of multiple tunnel entries created by different protocols to a given destination. The lower preference value is preferred. *Table 8-1* lists the MPLS-based protocols and their default preferences.

Table 8-1: Default Tunnel Table Preferences

Protocol	Preference
RSVP	7
SR-TE	8
LDP	9
SR-OSPF	10
SR-ISIS	11
BGP	12
SR Policy	14

When deploying SR, it is highly likely that one or more other MPLS-based protocols are already operational in the network. For example, if LDP is already deployed and used within a network and shortest path SR is subsequently enabled, it will not be used because the tunnel-table preference is higher than that of LDP. Similarly, if RSVP is already deployed and used and SR-TE is enabled, it again will not be used by default because RSVP has a lower preference than SR-TE.

When migrating BGP and services to use SR there are multiple options on how this can be done, which can be loosely considered as granular or coarse. Indeed, both approaches might be used over the course of a migration. A ‘toe-in-the-water’ approach would be to use the **resolution-filter** to select the relevant SR protocol (**sr-isis**, **sr-ospf**, **sr-policy**, **sr-te**) and remove any other protocols in use that have a lower preference. This approach would be applied at BGP route or service level and might be used at the beginning of any migration to SR to validate that the network is operating as anticipated. As confidence grows, a second more coarse approach might reduce the migration overhead, and that involves configuring the tunnel-table preference to be something other than the default value. For example, assume a network is currently using LDP, and the requirement is to move to SR-OSPF. The default values are 9 for LDP and 10 for SR-OSPF, so one might consider making LDP a higher preference, or SR-OSPF a lower preference. The tunnel-table preferences are configurable for SR-ISIS, SR-OSPF, and SR-OSPF3 using the IGP **segment-routing tunnel-table-pref** command. The tunnel-table preferences are also configurable for SR-TE and RSVP-TE using the **mpls tunnel-table-pref {rsvp-te|sr-te}** command.

9 Fast Reroute

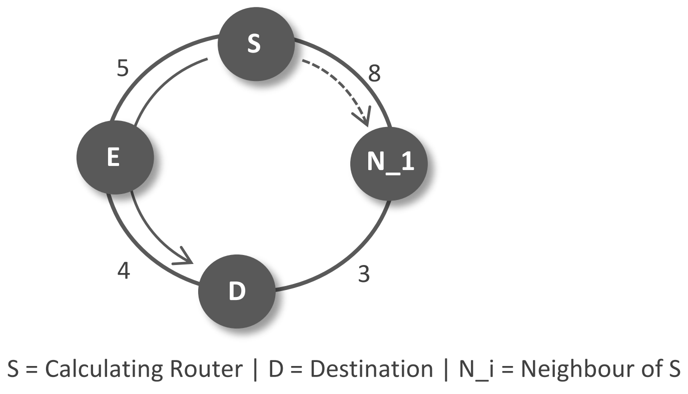
The ability for the network to rapidly reconverge following a failure has become largely a standard requirement in IP networks today as many applications using the network are sensitive to traffic loss of more than tens of milliseconds. Unlike RSVP-TE that has control plane extensions to establish and manage fast reroute repair tunnels for protection, SR has no control plane to signal to backup paths or to notify the headend that a path has been locally repaired. SR therefore relies on IP fast reroute techniques using Loop-Free Alternates (LFA), which allow a router to locally compute a backup path without any control plane interaction.

LFA Overview

The basic premise of IP Fast Reroute is to use a pre-computed alternate next-hop so that when a failure is detected with the primary next-hop, the alternate can be rapidly used until an SPF is run and a new primary next-hop is installed.

To illustrate the use of IP Fast Reroute consider the example in *Figure 9-1* which uses the terminology from the IP Fast Reroute Framework [13] and Basic Specification for IP Fast Reroute [14]. S is used to indicate the calculating router, N_i is a neighbour of S, and D is the destination under consideration. In this example router S computes the shortest path to router D and uses E as its primary next-hop. Router S also computes that router N_1 is a feasible alternate next-hop for destination D and installs it as its alternate next-hop. If the link between S and E should fail, router S (and E) will be the first to detect it at which point S will stop sending traffic towards E and start forwarding traffic to its pre-computed alternate next-hop N_1 until a new SPF is run and installed.

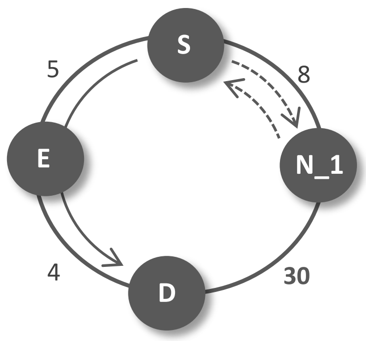
Figure 9-1: IP Fast Reroute: Loop-free Alternate


S
5
8
E
N_1
4
3
D
S = Calculating Router | D = Destination | N_i = Neighbour of S

Now consider the example in *Figure 9-2* where the metric between N_1 and D is increased to 30. In this case router N_1 would no longer be a suitable loop-free alternate for router S because the cost of the path from N_1 to D via would be 17 whilst the cost from N_1 directly to D would be 30. In short, if the link S to E failed and router S started to forward packets to

N_1, router N_1 would simply forward them back to S and a transient loop would exist until the SPF is complete.

Figure 9-2: No Suitable Loop-free Alternate



To determine if a neighbour N can provide a loop-free alternate (LFA), the following loop-free criterion is used, where *distance_opt(X,Y)* is used to indicate the shortest distance between X and Y:

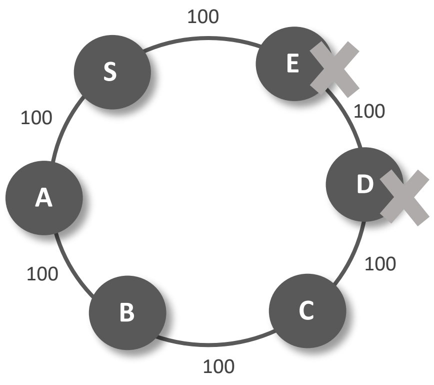
Distance_opt(N, D) < Distance_opt(N, S) + Distance_opt(S, D)

This criterion can be further extended to provide node protection. For an alternate next-hop N to provide node failure protection of a primary neighbour E for destination D, N must be loop-free with respect to both E and D. In short, N's path to D cannot transit through E.

Distance_opt(N, D) < Distance_opt(N, E) + Distance_opt(E, D)

The basic LFA described above provides good protection coverage where networks are highly meshed, but many topologies (especially rings) do not get good coverage from basic LFA alone. Consider the case in *Figure 9-3*. In this simple (but not unusual) topology, link S-E cannot be fully protected by S. Router C is ECMP from S and so traffic can be protected when the S-E link fails, but traffic to D and E are not protectable using basic LFA. If S should forward packets destined for D or E to router A upon failure of the S-E link, router A will simply forward them back to S.

Figure 9-3: Basic LFA in a Ring Topology

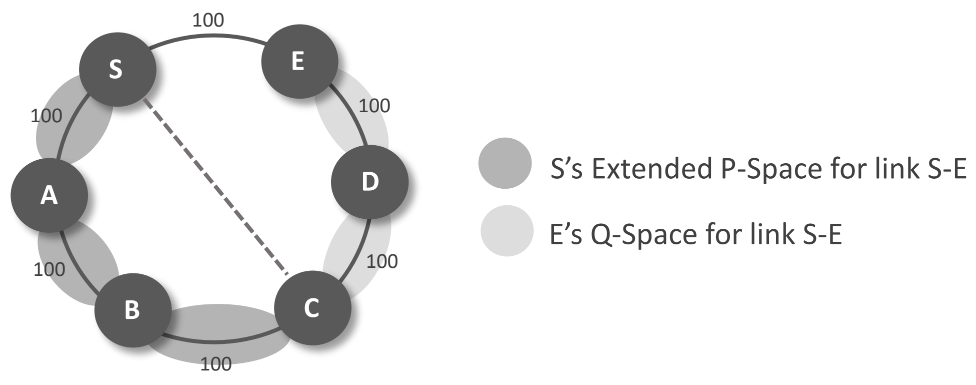


Remote LFA (RLFA) [15] extends the basic LFA repair mechanism to increase fast reroute coverage. If a link cannot be protected with local LFA neighbours, RLFA tries to create a virtual LFA by using a (repair) tunnel to carry packets to a point in the network where they will not be looped back. To compute an RLFA repair path (tunnel endpoint) for the link S-E, it is necessary to determine the set of routers which can be reached from S without traversing the link S-E, and match that with the set of routers that can reach E without traversing the link S-E. This is known as S's *Extended P-space*, and E's *Q-space*.

- Extended P-space: The generic P-space is the set of routers that S can reach without traversing the link S-E. The Extended P-space (often denoted as P' space) of S with respect to the link S-E is the union of the P-spaces of its neighbours (excluding E) in respect to link S-E, and in this example, A is the only neighbour. Extended P-space works on the premise that since S will only use a repair path when link S-E has failed, the initial hop of the repair path does not need to be subject to S's normal forwarding decision process. In other words, once a packet destined to prefix P is forced to the neighbour by S it is a lower cost for it to continue on to P by any other path except one that takes it back to S and then across the S-E link. The use of Extended P-space provides better repair coverage and for that reason is the preferred approach.
- Q-space: The set of routers that can reach E without going through link S-E.

In *Figure 9-4* S's Extended P-space is calculated by running an SPT rooted at each of S's neighbours (N) to calculate which nodes can reach the neighbour without going through the link S-E. This equates to A, B, and C. E's Q-space is a reverse SPT rooted at E to compute the routers that can reach E without going through link S-E. A reverse SPT is the cost towards the root rather than from it and yields the best paths towards the root from other nodes in the network. E's Q-space equates to D and C (B is ECMP towards E and may forward packets to E via the link S-E, hence it is not included in E's Q-space). The point at which the Extended P-space and the Q-space intersect is referred to as the PQ-node and is the point to which a repair tunnel can be built. In this case it is router C. This repair tunnel can only be used for repair traffic.

Figure 9-4: RLFA Repair Tunnel



RLFA makes no assertions about the tunnel mechanism that is used to build the repair tunnel, however, it's clear that in an MPLS-enabled network a label stack may be used to build the repair tunnel. Because RSVP-TE has its own Fast Reroute mechanisms, the use of RLFA is applicable (or rather extended to) LDP-based MPLS networks. One approach to building the RLFA repair tunnel is to use targeted LDP, however, this approach was not favoured by many operators because of the requirement for tunnel-in-tunnel (LDP-over-RSVP or LDP-over-LDP), and also for the ad-hoc nature of the setup and teardown of targeted LDP sessions which may change every time the topology changes. With the introduction of SR an RLFA repair tunnel is a very simple concept constructed simply by pushing the Node-SID of the repair tunnel endpoint (PQ-node) onto the SR label stack.

Despite the fact that RLFA provides significant protection coverage over that of basic LFA, it could not guarantee full network coverage; there are situations where the calculation of Extended P-space and Q-space cannot derive an intersect point, or PQ-node. In addition to this, operational concerns were raised [16] regarding the use of LFA and RLFA because links used transiently by LFA/RLFA repair paths could become points of congestion that were difficult to capacity plan.

Topology Independent LFA (TI-LFA) using SR [17] address these issues. TI-LFA extends on the concepts of LFA and remote LFA, to provide guaranteed coverage in any topology. TI-LFA computes the repair paths over the post-convergence path from the point of local repair (PLR) to the protected destination. That is, the protection path will automatically follow the actual backup path that would be used after the SPF has been executed and the network has converged. This allows operators to better capacity plan the backup paths and avoids multiple path changes (pre-convergence path to fast reroute path, and then fast reroute path to post-convergence path). To do this the PLR initially computes the post-convergence path to the destination whilst removing the protected (link/node) resource from the topology. The outcome is referred to as *SPT_new* and yields the shortest path tree to the destination rooted at the calculating node in the state of the network after the protected resource has failed. The PLR then computes the extended P-space and the Q-space with respect to the protected resource. The post-convergence repair path from the PLR to the destination D is subsequently found by computing the intersection of the candidate P-Q nodes with SPT_new.

In situations where no PQ-node intersect can be found between the P-node and Q-node a repair tunnel can still be built with an SR label stack consisting of a Node-SID to the P-node followed by an explicit path to the Q-node consisting of one or more Adj-SIDs. For example, if the P-node and the Q-node are adjacent neighbours, the repair tunnel would be constructed of two SIDs; the Node-SID of the P-node followed by the Adj-SID representing the P-Q adjacency. (Historically, this approach of forcing a packet over a particular adjacency when it is decapsulated from the repair tunnel was called Directed LFA or DLFA and sometimes this terminology is still used.)

The TI-LFA repair tunnel consists of an outgoing interface and a list of segments to insert on the SR header. The list of segments (or repair list) encodes the explicit post-convergence path to the destination avoiding the protected resource. The type of repair tunnel computed by the PLR will determine the number of segments that are imposed in that repair list:

- If the LFA backup is an adjacent neighbor there is no requirement for a tunnel and no additional labels are pushed.

- If the P-node and Q-node resolve to the same router that is a non-adjacent neighbor it equates to a remote LFA and requires that a single additional label is pushed; the Node-SID of the PQ-node.
- If the P-node is an adjacent neighbor of the Q-node it equates to a directed LFA and requires that two labels are pushed; the Node-SID of the P-node followed by the Adj-SID of the Q-node. If the P-node is not adjacent to the Q-node multiple Adj-SIDs can theoretically be pushed to explicitly define the path between the P-node and the Q-node but this, then becomes a trade-off between LFA coverage and repair tunnel overhead.

Analysis on real service provider and large enterprise networks [16] indicates that for link protection a one SID repair path delivers more than 99% coverage, and for node protection a two SID repair path yields 99% coverage.

Enabling LFA for SR in SR-OS

Enabling LFA for use with SR is a relatively simple process, and the primary focus within this section will be enabling TI-LFA due to the previously outlined benefits that it brings in conjunction with SR. Unlike vanilla IP fast reroute and LDP fast reroute that both require configuration of a ‘fast-reroute’ command, SR simply requires that **loopfree-alternate** is enabled within the IS-IS or OSPF context.

Output 9-1 shows the configuration required to enable TI-LFA for SR. Within the **loopfree-alternate** context further configuration contexts exist for both **remote-lfa** and **ti-lfa** and in this example both are enabled. It is entirely possible to enable and run both **remote-lfa** and **ti-lfa** concurrently and the interaction between the two is derived from the sequence through which the LFA SPF is executed:

a. First, compute a basic LFA SPF.
b. Next, if **ti-lfa** is configured run a TI-LFA SPF regardless of the outcome of the first step. If a TI-LFA next-hop is found it overrides the result from the first step.
c. Finally, if **remote-lfa** is configured run RLFA for the next-hop of prefixes that remain unprotected after the previous two steps.

Although it is unlikely that prefixes may remain unprotected after step b, it is possible of course that the computing router was unable to compute a repair tunnel on the explicit post-convergence path to the destination. This could be perhaps as a result of local policy as described later in this section, and since an RLFA SPF will only occur if this happens it is enabled as a ‘catch all’. Note that SR-OS will not install an alternate next-hop if ECMP paths are available. In this case prefixes are installed with multiple primary next-hops which provides the equivalent level of protection.

Under the **ti-lfa** context the **max-sr-frr-labels** command provides the ability to limit the repair tunnel overhead by specifying a value between 0 and 3. The configured value of 2 allows for a repair tunnel to a non-adjacent P-node with an adjacent Q-node, formed of a Node-SID to the P-node followed by the Adj-SID of the P-Q link.

At SR-TE LSP level the **max-sr-labels** context provides the similar ability to limit the number of imposed labels for a primary LSP and repair tunnel. Within that context the **label-stack-size** command allows the user to specify a value between 1-11 for the primary LSP, while the **additional-frr-labels** command allows the user to specify a value between 0-3 to limit the number of labels imposed by a repair tunnel. A good rule of thumb is to ensure that the values configured within the IGP and SR-TE LSPs are aligned to the objectives of the network.

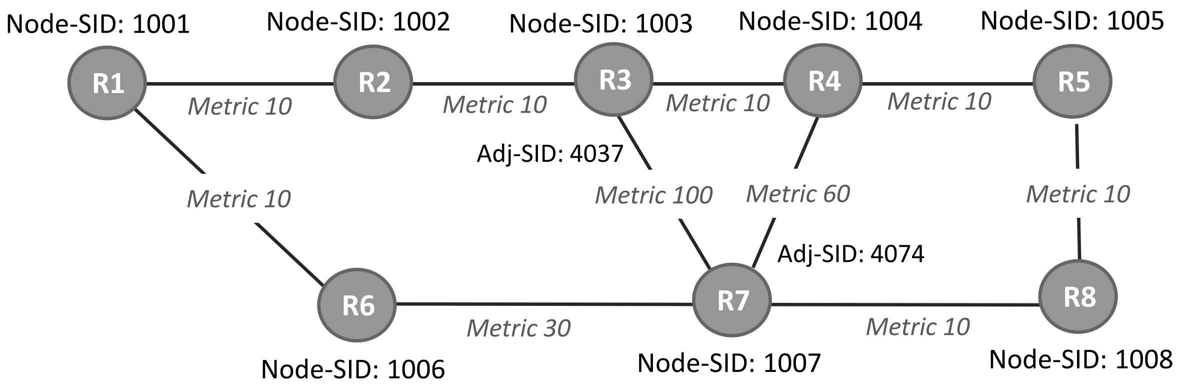
Output 9-1: TI-LFA Configuration

```
        isis 0 {
            loopfree-alternate {
                remote-lfa {
                }
                ti-lfa {
                    max-sr-frr-labels 2
                }
            }
        }
```

Once TI-LFA is enabled it will attempt link-protection by default and will provide a backup for both shortest-path SR LSPs and SR-TE LSPs. The TI-LFA link protection algorithm searches for the closest Q-node to the computing node and then selects the closest P-node to this Q-node, bounded by the value of **ti-lfa max-sr-frr-labels**.

Within the **ti-lfa** context there is an option to enable node-protection with the **node-protect** command, and if enabled the system will prefer a node-protect repair tunnel over a link-protect repair tunnel if both can be found. The initial instinct may be to enable node-protection to get a better level of protection, however, it is worth understanding that node-protection cannot provide protection to any SR-TE LSP that contains either a Node-SID of the protected node or an Adj-SID belonging to the protected node. To understand why consider the example in *Figure 9-5* with an explicit path from R1 to R5 through the path R1-R2-R3-R7-R4-R5. To construct this path the label stack at R1 uses a combination of Node-SIDs and Adj-SIDs to form the label stack {1003, 4037, 4074, 1005}.

Figure 9-5: Node-Protection for Adj-SID Explicit Path



When the packet reaches R3 it switches the packet towards R7. R3's next-next-hop in the path is R4, and to provide protection against the failure of R7, R3 would need to send the packet to R4 without going through R7. However, the only way R3 can learn that the next-next-hop for this particular packet is R4 is to look at the next label in the stack, label 4074. Once it has done this R3 can interrogate the TE database to ascertain that R7 advertised Adj-SID 4074 for the link R7-R4. With this information R3 knows that a backup path can merge back onto the original explicit path at R4. To implement node-protecting backup paths for SR-TE LSPs a given node needs to construct context tables for each of its neighbours containing the incoming label mappings advertised by the neighbour and the actions corresponding to those labels. The labels advertised by each of the neighbours may not be unique across the SR domain, so these tables need to be retained in separate tables built for each neighbour to represent the label FIBs of a particular neighbour. In the example of *Figure 9-5*, to provide a node-protecting backup path R3 recognises that label 4037 corresponds to a next-hop of R7, so it installs a backup entry corresponding to the failure of the link to R7 whereafter it needs to examine that next label in R7's context table. *Figure 9-6* shows R3's incoming label mapping pertinent to this example. In non-failure mode incoming label 4037 is popped and forwarded to R7. In failure mode when the backup is activated, label 4037 is popped and R3 examines the new top label (4074) in R7's context table. R7's context table illustrated in *Figure 9-7* which shows that R7 would pop this label and forward the packet to R4. R3 can thereafter derive that a backup path for this packet can safely merge at R4.

Figure 9-6: R3's Incoming Label Mapping

In Label	**Outgoing Label Action**
4037	Primary: Pop, forward to R7 Backup: Pop, lookup 'context-R7'

Figure 9-7: R3's Context Table for R7

In Label	**Outgoing Label Action**
1005	Swap 1005, forward to R4
4074	Pop, forward to R4

It should be reasonably obvious that constructing context label FIBs in this manner is both computationally expensive and has the potential to consume a significant amount of label FIB space. It is not currently supported in SR-OS and as a result node protection for SR-TE LSPs with either Node-SIDs or Adj-SIDs attached to the protected node cannot be provided.

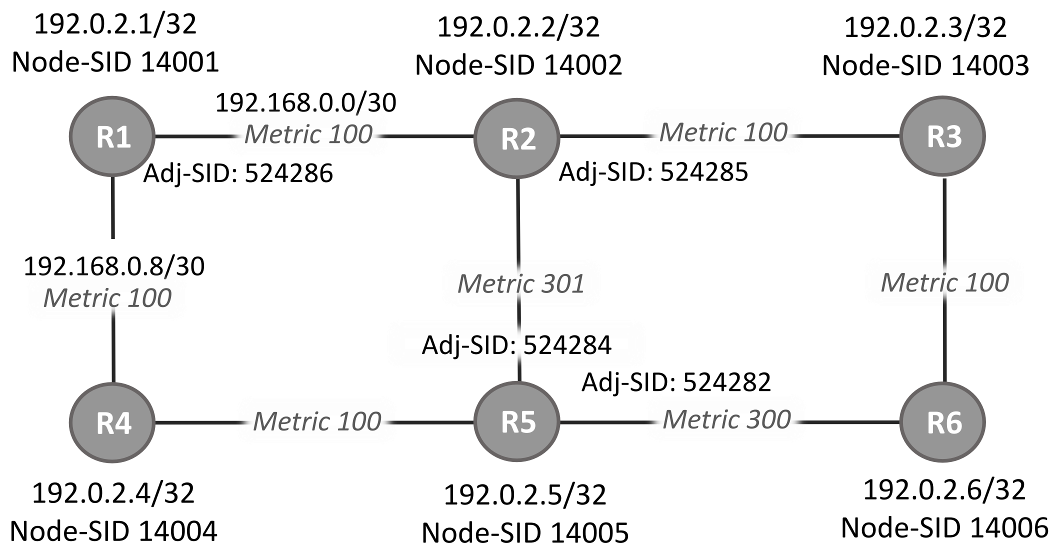
As previously described in chapters 3 and 4 Adj-SIDs are advertised with the backup flag (B-flag) not set when **loopfree-alternate** is disabled and set to 1 when **loopfree-alternate** is enabled. For completeness, *Output 9-2* shows the IS-IS LSP advertised by router R1 truncated to show only the IS Neighbours TLVs to R4 and R2. In the Adj-SID sub-TLVs the B-flag is now set indicating that the Adj-SID is now eligible for protection. This allows a headend to select a path that only includes protected or unprotected adjacencies as required.

Output 9-2: Adj-SID Flags with LFA Enabled

```
A:admin@R1# show router isis database R1.00-00 level 2 detail
...[ snip ]]
TLVs :
  TE IS Nbrs   :
    Nbr   : R4.00
    Default Metric  : 100
    Sub TLV Len     : 76
    IF Addr   : 192.168.0.9
    Nbr IP    : 192.168.0.10
    MaxLink BW: 10000000 kbps
    Resvble BW: 10000000 kbps
    Unresvd BW:
        BW[0] : 10000000 kbps
        ...[ snip ]
    Admin Grp : 0x0
    TE Metric : 10
    Adj-SID: Flags:v4BVL Weight:0 Label:524287
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 76
    IF Addr   : 192.168.0.1
    Nbr IP    : 192.168.0.2
    MaxLink BW: 10000000 kbps
    Resvble BW: 10000000 kbps
    Unresvd BW:
        BW[0] : 10000000 kbps
        ...[ snip ]
    Admin Grp : 0x0
    TE Metric : 100
    Adj-SID: Flags:v4BVL Weight:0 Label:524286
...[ snip ]
```

To provide an example of LFA protection the topology depicted in *Figure 9-8* is used. Routers R1 through R6 form part of the same IS-IS level 2 area, and all link metrics are 100 with the exception of the link R2-R5 which has a metric of 301 and the link R5-R6 which has a metric value of 300. All routers within the domain have the LFA configuration applied as described throughout this section. SR is enabled with an SRGB of 12000-19999 and the associated Node-SIDs are shown together with some pertinent Adj-SIDs.

Figure 9-8: Test Topology for TI-LFA



To validate that LFA is active and illustrate how protection is applied, two SR LSPs are used from R1 to R3 :

- An SR-ISIS LSP that follows the shortest path R1-R2-R3. As it is a shortest path SR LSP the label stack contains only R3's Node-SID, 14003.

- An SR-TE LSP computed with local CSPF using a metric constraint of IGP, which follows the path R1-R2-R3. As shown in *Output 9-3* this LSP contains explicit Adj-SIDs of 524286 (R1-R2 link) and 524285 (R2-R3 link). Note that only Adj-SID 524285 (R2-R3 link) will appear on-the-wire when R1 forwards traffic into this LSP since the local Adj-SID 524286 is a label-swap to implicit null at R1.

Output 9-3: R1-to-R3 SR-TE LSP

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Actual Hops" post-
lines 2
Actual Hops      :
    192.168.0.2(192.0.2.2)(A-SID)                Record Label        : 524286
 -> 192.168.0.6(192.0.2.3)(A-SID)                Record Label        : 524285
```

Output 9-4 shows the status of those LSPs with TI-LFA enabled. Detailed explanation on the programming of each of the LSPs is provided immediately following the output.

Output 9-4: R1's FP Tunnel-Table for Destination R3

```
A:admin@R1# show router fp-tunnel-table 1 192.0.2.3/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                 Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                     Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.3/32                                SR-ISIS-0            524299
  14003
    192.168.0.2                                                  1/1/c1/1:100
  14003/524282/14005
    192.168.0.10(B)                                              1/1/c2/1:100
192.0.2.3/32                                SR-TE                655366
  524285
    192.168.0.2                                                  SR
-------------------------------------------------------------------------------
Total Entries : 2
-------------------------------------------------------------------------------
```

The first entry is the SR-ISIS LSP. It has a primary next-hop of 192.168.0.2 (R2) with a label stack of {14003} representing the Node-SID of R3. It then has a backup next-hop indicated by the (B) notation with a next-hop of 192.168.0.10 (R4) and a label stack of {14005, 524282, 14003} from top to bottom. When computing a backup path for the failure of the R1-R2 link for destination R3, R1's post-convergence path is R1-R4-R5-R6-R3.

- R1's extended P-space includes only R5 and the top label of 14005 represents R5's Node-SID.
- R1's Q-space in respect of the R1-R2 link for destination R3 includes only R6. As a result, the P-node and Q-node are adjacent and the next label in the stack 524282 represents R5's Adj-SID for the R5-R6 link.
- The final label 14003 is the Node-SID for the destination R3. In effect this is a directed LFA.

The second entry is the R1-R3 SR-TE LSP. It has a primary next-hop of 192.168.0.2 (R2) with a label stack of 524285 representing R2's Adj-SID for the R2-R3 link. SR-TE LSPs are internally modelled as hierarchical constructs. Although an SR-TE LSP may have a number of labels (segments) imposed, the top label is the only label that impacts the forwarding decision; the rest of the stack is not meaningful to that forwarding decision. The label stack is referred to as a super NHLFE (described in chapters 3 and 4) and is tunnelled over the NHLFE of the first hop. Unlike the previous SR-ISIS LSP the output does not indicate that the SR-TE LSP has a backup, and this is because the SR-TE LSP uses this hierarchical construct. The primary next-hop is 192.168.0.2 (R2) and the super NHLFE is tunnelled over the NHLFE for this next-hop. When LFA is enabled Adj-SIDs are automatically afforded link-protection in order to protect SR-TE LSPs containing explicit hops using Adj-SIDs, and as R1 has an Adj-SID for the adjacency to R2 it is protected. The previous command is therefore repeated for this next-hop NHLFE in *Output 9-5*. As can be observed it has a primary next-hop of 192.168.0.2 (R2) with a label value of 3 indicating a label switch to implicit-null. It then has a backup next-hop of 192.168.0.10 (R4) with the label stack {14005, 524284, 14002} from top to bottom. The top label 14005 equates to the Node-SID of R5 and is followed by label 524284 representing R5's Adj-SID for link R5-R2. The final label in the stack is R2's Node-SID 14002, which R2 will pop for further processing as required.

Output 9-5: R1's FP Tunnel-Table for R2's Adj-SID

```
A:admin@R1# show router fp-tunnel-table 1 192.168.0.2/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                 Protocol          Tunnel-ID
  Lbl/SID
    NextHop                                                   Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.168.0.2/32                              SR                 524289
  3
    192.168.0.2                                                1/1/c1/1:100
  14002/524284/14005
    192.168.0.10(B)                                            1/1/c2/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Adjacency SID protection

As briefly described in the previous section, when LFA is enabled SR-OS will automatically provide link-protection to Adj-SIDs. This allows SR-TE LSPs containing explicit hops with Adj-SIDs to be protected, noting that only link-protection is possible because packets either terminate or have a segment endpoint on the next-hop router. That is, the Adj-SID belongs to the next-hop router. In addition, the Adj-SID sub-TLVs for all adjacencies are subsequently advertised with the backup flag (B-flag) set to one. If there is a requirement to have a particular adjacency unprotected, it can be set at IS-IS/OSPF interface level using the **sid-**

protection false command. (By default, the **sid-protection** command is set to **true**, and just the process of enabling LFA will trigger the advertisement of Adj-SIDs with the B-flag set.)

Output 9-6: Disabling Adj-SID Protection

```
        isis 0 {
            interface "link-to-R2" {
                sid-protection false
            }
        }
```

Setting **sid-protection** to **false** clears any backup NHLFE for the adjacency, and also sets the B-flag to zero. For example, the default NHLFE programming at R1 for the adjacency to R2 in *Output 9-5* consisted of a primary and backup next-hop. With **sid-protection** set to false for this interface it can be compared to the NHLFE programming in *Output 9-7* where no backup next-hop is present.

Output 9-7: Adj-SID Protection Disabled

```
A:admin@R1# show router fp-tunnel-table 1 192.168.0.2/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                  Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                      Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.168.0.2/32                               SR                   524289
  3
    192.168.0.2                                                   1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

When a repair tunnel for an Adj-SIDs is activated, it is transient and is intended to provide protection until the network reconverges. The primary NHLFE is programmed as a result of an SPF, and the backup NHLFE is programmed as a result of an LFA SPF. Both SPFs rely on the topology advertised in the IGP. When a link fails the corresponding LSA/LSP together with the Adj-SID sub-TLV will be flushed by the routers at each end of the link. In turn, this will flush the primary/backup NHLFE entries from the FIB and the repair tunnel is removed. The amount of time that those NHLFEs will be retained in the FIB after the LSA/LSP is flushed is determined by the **adj-sid-hold** timer within the IS-IS/OSPF **segment-routing** context. By default, this value is 15 seconds.

Output 9-8: IS-IS Adj-SID Hold Timer

```
        isis 0 {
            segment-routing {
                adj-sid-hold <1-300 seconds>
                }
            }
        }
```

By way of illustration, in *Output 9-9* R1's port supporting the interface to R2 is disabled. 15 seconds later the FIB is interrogated for R1's entry to the R2 adjacency and both the primary and backup NHLFEs are gone.

Output 9-9: LSA/LSP Flushing Impact on FIB

```
A:admin@R1# configure port 1/1/c1/1 admin-state disable
*(pr)[/]
A:admin@R1# commit

15 seconds later

A:admin@R1# show router fp-tunnel-table 1 192.168.0.2/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                 Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                     Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
No Matching Entries
```

It's important to understand that Adj-SID protection is not infinite. The headend or a controller has the amount of time determined by the **adj-sid-hold** timer to take restorative action and recompute a new path around the failure.

LFA Policies

At the time of writing LFA Policies are not currently supported on 7250 IXR generation one or generation two platforms. Please check Release Notes for an updated list of 7250 IXR unsupported features.

LFA policies provide the capability to influence LFA backup next-hop computations to include criteria such as admin-groups and SRLGs. LFA policies do not use admin-group or SRLG information that is advertised within the IGP. Instead LFA policies are local decisions based on constraints that are explicitly configured on local interfaces.

When TI-LFA is enabled, the LFA SPF is computed in the following order:

- Step 1: Compute a regular LFA that satisfies any applied LFA policy.
- Step 2: Compute a TI-LFA on the post-convergence path.
- Step 3: Compute a remote LFA for any prefixes that remain unprotected by steps 1 and 2.

When LFA policies are used, SR-OS runs the TI-LFA SPF and calculates a list of candidate post-convergence repair paths, the size of which is bounded by the setting of ECMP within the base routing instance (if ECMP is 1 then only one candidate post-convergence repair path will be computed). It then takes the set of candidate backup next-hops and applies the constraints of the LFA policy, which in the case of admin-groups or SRLGs involves pruning links from the computed paths. As a general rule, if no viable TI-LFA backup next-hop exists after this process, the system will revert back to step 1 above, or if no LFA backup was found in step 1, it will run step 3 and try to find a remote LFA backup. The steps are generalised here because the outcome of the protection-type (node-protect, link-protect) and backup mechanism (LFA, RLFA, TI-LFA) are dependent on what LFA options are configured.

You may wonder why SR-OS computes the candidate post-convergence repair paths and then prunes admin-groups and SRLGs instead of firstly pruning admin-groups/SRLGs and then computing the post-convergence repair path. This is done to minimise any risk of creating a loop. For example, consider an LFA policy with an SRLG constraint, where the primary path for destination D is declared down by a BFD timeout. In this scenario, any other members of the same SRLG will be unaffected and will remain in the IGP topology. However, if the SRLG member links were pruned before computing the post-convergence repair path the PLR will have a completely different topological view of the network to other routers in the domain. In this case it may start forwarding packets to a precomputed LFA backup in a manner that is not reflective of the actual network topology and potentially create a loop.

In addition to allowing for the use of admin-groups or SRLGs, the LFA policy allows for selection of node-protection or link-protection. However, the order of protection method (LFA, RLFA, TI-LFA) takes precedence over this. This means that for example, that a TI-LFA post-convergence repair path with link-protection would be preferred over a basic LFA or remote LFA repair path with node-protection.

To illustrate the use of LFA policies the network schematic from *Figure 9-8* is again used, but all link metrics are normalised to a cost of 100. TI-LFA and Remote LFA are configured as described in the previous section. The example will look at R2's primary and LFA backup paths to R5 and will place some links into a common SRLG to influence this LFA backup selection. Firstly, an **srlg-group** named 'red' is created as an **if-attribute** in **routing-options**. This **srlg-group** is then added as an **if-attribute** under the base routing interfaces to R1 and R5.

Output 9-10: SRLG Configuration

```
    router "Base" {
        interface "link-to-R1" {
            if-attribute {
                srlg-group "red" { }
            }
        }
        interface "link-to-R5" {
            if-attribute {
                srlg-group "red" { }
            }
        }
    routing-options {
        if-attribute {
            srlg-group "red" {
                value 10
            }
        }
    }
```

A **route-next-hop-policy** template is then created to allow for the LFA backup next-hop constraints to be specified. As previously described this template allows for the use of admin-groups (exclude-group, include-group), SRLGs, selection of the protection type (node-protect or link-protect), and to select whether the preferred next-hop should be IP or a tunnel. In the example of *Output 9-11* the **route-next-hop-policy template** 'srlg-red' is created and has the **srlg** constraint set to **true** with a **protection-type** of **link**. Link-protect is used in this example to provide protection to SR-TE LSPs with paths consisting of explicit Adj-SIDs.

Output 9-11: Route-Next-Hop-Policy Template

```
    routing-options {
        route-next-hop-policy {
            template "srlg-red" {
                srlg true
                protection-type link
            }
        }
    }
```

The final part of the configuration is to apply the **route-next-hop-policy** to the interface through which the primary path is routed. Prior to doing that though, the routing of R2's primary path and backup LFA path to R5 is captured. This is shown in *Output 9-12* which shows the following:

- The primary path contains a single label of {14005} representing R5's Node-SID and has a next-hop of 192.168.0.14 (R5).
- The backup path contains a label stack of {14004, 14005} from top to bottom, representing R4's Node-SID followed by R5's Node-SID. The next-hop is 192.168.0.1 (R1) and it will route along the path R2-R1-R4-R5.

Output 9-12: Primary/Backup Path Routing before LFA Policy Applied

```
A:admin@R2# show router fp-tunnel-table 1 192.0.2.5/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                             Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                 Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.5/32                            SR-ISIS-0            524296
  14005
    192.168.0.14                                             1/1/c3/1:100
  14005/14004
    192.168.0.1(B)                                           1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
```

```
--------------------------------------------------------------------------------
```

The **route-next-hop-policy** is then applied at R2 within the IS-IS/OSPF context on the interface through which the primary path is routed. In this case it is the interface towards R5.

Output 9-13: Applying the Route-Next-Hop Policy

```
    router "Base" {
        isis 0 {
            interface "link-to-R5" {
                loopfree-alternate {
                    policy-map {
                        route-nh-template "srlg-red"
                    }
                }
            }
        }
    }
```

Output 9-14 shows the primary and backup path routing after the LFA policy has been applied. The primary path remains the same as anticipated. The backup path however has changed and contains a label stack of {14006, 14005} from top to bottom, representing R6's Node-SID followed by R5's Node-SID. The next-hop is now 192.168.0.6 (R3) and it will now route along the path R2-R3-R6-R5. The primary and backup paths are now SRLG diverse due to the application of the LFA policy.

Output 9-14: Primary and Backup Path Routing After LFA Policy Applied

```
A:admin@R2# show router fp-tunnel-table 1 192.0.2.5/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                   Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                       Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.5/32                                  SR-ISIS-0            524296
  14005
    192.168.0.14                                                   1/1/c3/1:100
  14005/14006
    192.168.0.6(B)                                                 1/1/c4/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

It has previously been described that when LFA policies are used, SR-OS will run the TI-LFA SPF and calculate a list of post-convergence paths and will then apply the constraints of the LFA policy to those paths. The impact of this is important and needs to be understood. To illustrate it, I'll first step through the LFA SPF process carried out by R2 when it is able to successfully compute a repair path. The network topology will then be modified so that R2 fails to calculate a repair path, after which I will repeat the LFA SPF process carried out by R2 to identify why.

When R2 initially applies the LFA policy and successfully computes the repair path shown in *Output 9-14* the process of computing the repair path was as follows:

a. R2 runs the TI-LFA SPF to calculate a list of candidate post-convergence repair paths, the size of which is bounded by the setting of ECMP. In this case the configured ECMP is greater than two, and as R2 has two ECMP paths to R5 it computes two candidate post-convergence repair paths, one via R2-R1-R4-R5, and the other via R2-R3-R6-R5.
b. R2 then applies the constraints of the LFA policy, which dictates that links belonging to SRLG 'red' should be pruned. This means the removal of the link R2-R1 but leaves a single TI-LFA repair path available through R2-R3-R6-R5 and as a result this repair path was used.

The link metric of the link R3-R6 is now modified to a value of 301 and as a result this leaves prefix 192.0.2.5/32 (R5) with no backup repair path as shown in *Output 9-15*.

The process of attempting to compute the repair path at R2 would be as follows:

a. R2 runs the TI-LFA SPF to calculate a list of candidate post-convergence repair paths, the size of which is bounded by the setting of ECMP. In this case, R2 computes only a single path via R2-R1-R4-R5.
b. R2 then applies the constraints of the LFA policy, which dictates that links belonging to SRLG 'red' should be pruned. This means the removal of the link R2-R1, which in turn means the removal of the only candidate TI-LFA repair path.
c. R2 would then have attempted to fall back to a basic LFA, but no basic LFA can be computed.
d. R2 would then have attempted to compute a remote LFA, but again this would have failed as there is no PQ intersect on the path R2-R3-R6-R5.

Output 9-15: Failure to Compute LFA Backup

```
A:admin@R2# show router fp-tunnel-table 1 192.0.2.5/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                    Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.5/32                               SR-ISIS-0            524296
  14005
    192.168.0.14                                                1/1/c3/1:100
-------------------------------------------------------------------------------
Total Entries : 1
```

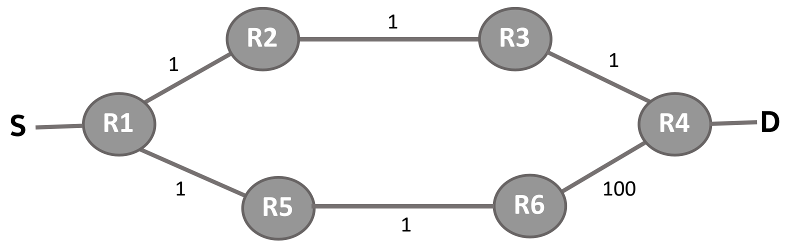
```
-------------------------------------------------------------------------------
```

Using LFA policies can be a useful mechanism for providing primary/backup SRLG diversity, or enforcing other constraints as required. Users should however be aware of the impact of applying them and take verification steps to ensure that the application of an LFA policy doesn't completely remove a backup path instead of increasing the backup path's availability.

Micro-Loop Avoidance

When there is topology change in a network all the nodes in the network need to reconverge on the change, and there will generally be some window of time when the forwarding states of each node are not completely synchronised. This can result in transient loops in the network, and because they are typically short in nature they are referred to as micro-loops. *Figure 9-9* illustrates the issue. In this topology all links have an IGP metric of 1, except link R6-R4 which has a metric of 100. When source S forwards traffic to destination D the normal path is S-R1-R2-R3-R4-D. When the link R3-R4 fails, traffic from S to D can be subject to transient forwarding loops while routers update their forwarding state for destination D. For example, if R1 updates its forwarding table before R5, packets destined for D may loop between R1 and R5. Equally, if R5 updates its FIB before R6 packets for D may loop between R5 and R6. The same transient micro-loops can occur when the R3-R4 link is restored. If R1 converges before R2 traffic for D will loop between R1 and R2. These differences in time to reconverge can be due to a number of things such as older routers with slower CPU, or perhaps SPF timers that are not equal across the network.

Figure 9-9: Micro-Loop Illustration



To avoid this, a headend can build an explicit path using an SR tunnel to a point in the network that ensures loop-free behaviour. For example, R1 could steer traffic to D using an SR path {Node-SID R6, Adj-SID R6-R4, D}. This ensures a loop-free path regardless of the forwarding state of R5 and R6. After some period of time allowing the network to reconverge, R1 can revert to normal forwarding without an SR path.

SR-OS supports micro-loop avoidance using SR [19] for a single event in the following cases:

- Link addition or restoration
- Link removal or failure
- Link metric change

If two or more link events are detected, the micro-loop avoidance procedure is aborted, and conventional forwarding behaviour is resumed. The basic premise of micro-loop avoidance is that a reconverging node does not trust the loop-freeness of its post-convergence path. Therefore, it applies a two-stage convergence process:

i. Step 1. After computing the new path to D, for a predetermined amount of time C, install a FIB entry for D that steers packets to D via a loop-free path. C should be greater than or equal to the worse-case convergence time of any node, network-wide.
ii. Step 2. After the period C expires, install the normal post-convergence FIB entry for D without any additional segments to ensure loop-freeness.

In the example of *Figure 9-9*, when the link R3-R4 fails, R1 forwards traffic for destination D over the SR path {Node-SID R6, Adj-SID R6-R4, D} for C milliseconds. During that time packets sent by R1 to D are guaranteed to be loop-free. When C expires, R1 now uses its normal post-convergence path to the destination, forwarding packets for D natively to R5.

In order to determine when and to where a repair tunnel should be constructed, R1 firstly needs to determine which routes are eligible for micro-loop avoidance. When link R3-R4 fails, LSA/LSP updates will be received from R3 and R4 that will trigger an SPF at R1. Routes not eligible for micro-loop avoidance include the following:

- Routes for which the SPF run results in no change to the next-hop router. At R1 this includes the route-table entries for R2, R3, R5, and R6.
- Routes for which the SPF run results in a change to another ECMP next-hop.

If the SPF run does result in a change in the single next-hop for a given prefix, then the route is marked as eligible for micro-loop avoidance and the repair tunnel is programmed as computed by micro-loop avoidance. In this example, R1 will change the single next-hop for prefixes at R4 and D.

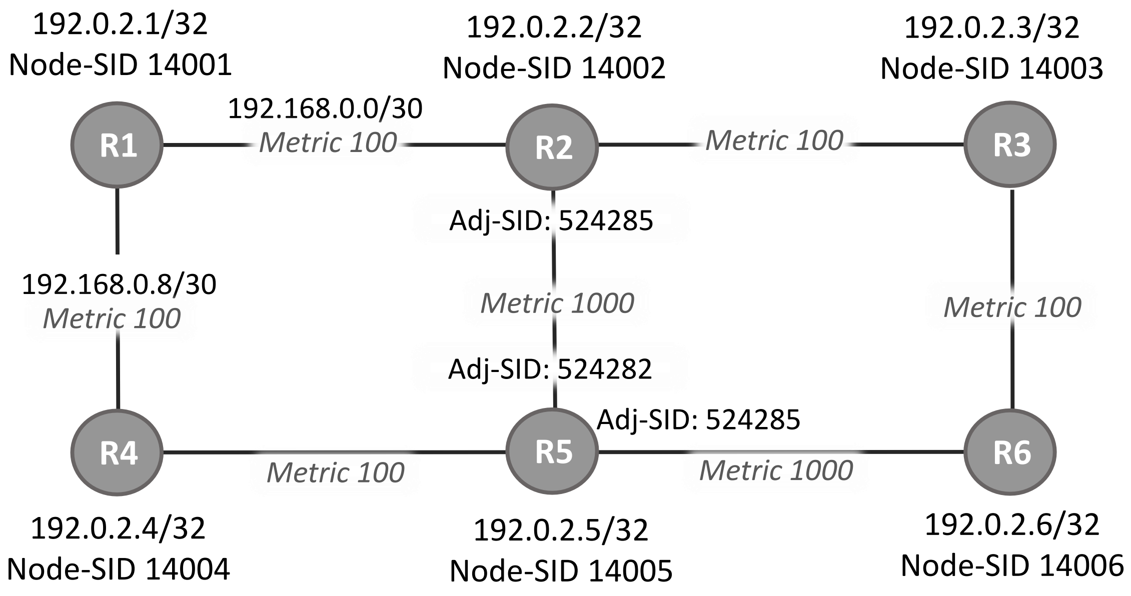
Micro-loop avoidance is enabled in SR-OS under the IS-IS **segment-routing** context. Micro-loop avoidance is not currently supported for OSPF. Within the **micro-loop-avoidance** context, the **fib-delay** command allows for configuration of a value between 1 and 300 in units of 100s of milliseconds to represent the timer 'C' . Once **micro-loop-avoidance** is enabled, the default **fib-delay** is 15 (1.5 seconds). When enabled it is applicable to shortest-path SR LSPs, and SR-TE LSPs/SR Policies that do not contain explicit Adj-SIDs in their segment lists. It should be noted that micro-loop avoidance is independent of TI-LFA; it can be used regardless of whether TI-LFA is enabled or not, but it does use TI-LFA procedures to compute the post-convergence path.

Output 9-16: Configuration for Micro-Loop Avoidance

```
        isis 0 {
            segment-routing {
                micro-loop-avoidance {
                    fib-delay 20
                }
            }
        }
```

To provide an illustration of micro-loop avoidance in action the generic topology is modified as shown in *Figure 9-10*. As usual all routers are in IS-IS level 2 and SR is enabled. Node-SIDs are shown together the relevant Adj-SIDs to aid understanding of the following outputs. All link metrics are 100, with the exception of links R2-R5 and R5-R6 which have a metric of 1000.

Figure 9-10: Topology for Micro-Loop Avoidance Example



In general, a reasonably short **fib-delay** timer would be used in most networks. To allow time to make the relevant captures router R1 is configured with an artificially high **fib-delay** timer of 200 (20 seconds). In a non-failure condition, R1’s primary path to R3 is via 192.168.0.2 (R2) and consists of a single label of 14003 (R3’s Node-SID). There is a computed backup path with label stack {14005, 524282, 14003} from top to bottom. This represents R5’s Node-SID 14005, R5’s Adj-SID for the link R5-R2 524282, and finally R3’s Node-SID 14003.

Output 9-17: R1's SR-ISIS LSP to R3 in Non-Failure Condition

```
A:R1# show router fp-tunnel-table 1 192.0.2.3/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                 Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                     Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.3/32                                SR-ISIS-0            524291
  14003
    192.168.0.2                                                  1/1/c1/1:100
  14003/524282/14005
    192.168.0.10(B)                                              1/1/c2/1:100
-------------------------------------------------------------------------------
Total Entries : 1
```

```
--------------------------------------------------------------------------------
```

The link R2-R3 is failed and as soon as router R1 completes its SPF the route-table entry for R3 is updated to reflect a new (higher) metric of 1300.

Output 9-18: R1's Post-Failure Route-Table Entry for R3

```
*A:R1# show router route-table 192.0.2.3/32

================================================================================
Route Table (Router: Base)
================================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
--------------------------------------------------------------------------------
192.0.2.3/32                                  Remote  ISIS      00h00m02s  18
       192.168.0.10                                                 1300
--------------------------------------------------------------------------------
No. of Routes: 1
```

Output 9-19 is then captured while the **fib-delay** timer is running. As can be observed, R1 has computed a loop-free path to R3 with the SR label stack {14005, 524285, 14003} from top to bottom. This represents R5’s Node-SID 14005, followed by R5’s Adj-SID for the R5-R6 link 524285, and finally R3’s Node-SID 14003.

Output 9-19: R1's SR-ISIS LSP with FIB-Delay Timer Running

```
A:R1# show router fp-tunnel-table 1 192.0.2.3/32

================================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
================================================================================
Destination                                  Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                      Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
--------------------------------------------------------------------------------
192.0.2.3/32                                 SR-ISIS-0            524291
  14003/524285/14005
    192.168.0.10                                                  1/1/c2/1:100
--------------------------------------------------------------------------------
Total Entries : 1
--------------------------------------------------------------------------------
```

After expiry of the **fib-delay** timer normal forwarding is resumed and the micro-loop avoidance SR tunnel is removed. With the link R2-R3 still failed, R1’s primary SR LSP to R3 has a next-hop of 192.168.0.10 (R4) and consists of a single label of 14003 (R3’s Node-SID). There is also a backup path with a next-hop of 192.168.0.2 (R2) with a label stack of {524285, 14003} from top to bottom. This represents R2’s Adj-SID for the link R2-R5 524285, followed by R3’s Node-SID 14003.

Output 9-20: FIB Table After Expiry of the FIB-Delay

```
A:admin@R1# show router fp-tunnel-table 1 192.0.2.3/32
```

```
===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                              Protocol          Tunnel-ID
  Lbl/SID
    NextHop                                                Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.3/32                             SR-ISIS-0          524291
  14003
    192.168.0.10                                            1/1/c2/1:100
  14003/524285
    192.168.0.2(B)                                          1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

LDP Fast Reroute over SR

The LFA backup mechanisms using SR described in this chapter can be extended to include support for LDP FECs. Using SR tunnels to provide backup to LDP FECs is an extension of the LDP to SR interworking functionality described in chapter 5. The exception being that using SR tunnels for LDP LFA backup can be considered a half-duplex function as it only needs to interwork LDP to SR, and not SR to LDP. Enabling LDP to use SR tunnels for fast reroute backup allows for increased LFA coverage for LDP without the requirement to use ad-hoc targeted LDP sessions that need to be setup and torn down with each topology change. It allows LDP backups to use the explicit post-convergence path to the destination.

Recall from chapter 5 that for basic interworking in the LDP to SR data-plane direction, LDP uses an **export-tunnel-table** command to reference an export policy. The LDP process monitors the tunnel-table and if there is a /32 SR tunnel of type SR-ISIS (or SR-OSPF) that matches a prefix in the export policy, LDP programs an LDP ILM entry and stitches it to the SR Node-SID tunnel endpoint. When LDP is configured to use SR tunnels as a fast-reroute backup, the process is slightly modified. LDP monitors the tunnel-table for SR tunnels to the same destination prefixes as that of the LDP FECs that it has resolved. If the system cannot find a local LFA next-hop but an RLFA/TI-LFA SR repair tunnel exists, LDP programs the primary next-hop using an LDP NHLFE, and a backup NHLFE with an implicit null label pointing to the SR tunnel. The fact that the backup NHLFE is programmed with an implicit null label is noteworthy; LDP packets are not tunnelled over the SR tunnel, but rather the LDP label is stitched to the SR label stack.

To illustrate the use of LDP fast reroute over SR the base topology depicted in *Figure 9-10* is re-used with a couple of modifications:

– All link metrics are normalised to a value of 100.

– Link-layer LDP is enabled on all routers (configuration not included in this section), and all routers have resolved LDP FECs to all others in the topology.

The configuration starts by creating a policy that LDP will use to monitor the tunnel-table for SR tunnels to the same destination as the LDP FECs it has resolved. As shown in *Output 9-21* a **prefix-list** "All-Loopbacks" is initially configured that encompasses all of the system addresses in the topology, limiting it to only 32-bit prefix lengths. The **policy-statement** "LDP-to-SRR-FRR" then references that **prefix-list** as **from** criteria in addition to specifying that that the prefix must also come from **protocol isis**. The **to** criteria simply allows those prefixes to be exported to **protocol ldp** with an action of **accept**.

Output 9-21: Policy Statement for LDP to SR Interworking

```
    policy-options {
        prefix-list "All-Loopbacks" {
            prefix 192.0.2.0/24 type range {
                start-length 32
                end-length 32
            }
        }
        policy-statement "LDP-to-SR-FRR" {
            entry 10 {
                from {
                    prefix-list ["All-Loopbacks"]
                    protocol {
                        name [isis]
                    }
                }
                to {
                    protocol {
                        name [ldp]
                    }
                }
                action {
                    action-type accept
                }
            }
        }
    }
```

Once the policy has been defined, it is referenced in the LDP context using the **export-tunnel-table** command. The **fast-reroute** context is also configured with **backup-sr-tunnel true** to allow LDP to program backup NHLFEs over SR tunnels.

Output 9-22: Enabling LDP Fast Reroute over SR

```
    router "Base" {
        ldp {
            export-tunnel-table ["LDP-to-SR-FRR"]
            fast-reroute {
                backup-sr-tunnel true
            }
        }
    }
```

Output 9-23 shows R1's tunnel-table entries for R3 (192.0.2.3/32) once the above configuration has been applied. The first entry for R3 belongs to LDP. It has a primary path with a label of 524279 and a next-hop of 192.168.0.2 (R2).It also has a backup entry with a label value of 3 (implicit null) with a next-hop of 192.0.2.3 (R3) that uses SR as a backup. This is the backup NHLFE pointing to an SR tunnel. The SR tunnel that it will use is dependent on the SR tunnel that is currently in use, and the second entry for R3 shows the available SR paths. The primary SR tunnel contains a single Node-SID of 14003 (R3) with a next-hop of 192.168.0.2 (R2). The backup SR tunnel is a remote LFA consisting of a label stack {14005, 14003} from top to bottom, representing R5's Node-SID followed by R3's Node-SID. It also has a next-hop of 192.168.0.10 (R4).

Output 9-23: R1's Tunnel-Table Entries for R3 with LDP FRR Enabled

```
A:admin@R1# show router fp-tunnel-table 1 192.0.2.3/32

===============================================================================
IPv4 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                              Protocol            Tunnel-ID
  Lbl/SID
    NextHop                                                  Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
192.0.2.3/32                             LDP                  -
  524279
    192.168.0.2                                               1/1/c1/1:100
  3
    192.0.2.3(B)                                              SR
192.0.2.3/32                             SR-ISIS-0            524301
  14003
    192.168.0.2                                               1/1/c1/1:100
  14003/14005
    192.168.0.10(B)                                           1/1/c2/1:100
-------------------------------------------------------------------------------
Total Entries : 2
-------------------------------------------------------------------------------
```

When **fast-reroute backup-sr-tunnel** is enabled under LDP, the preference order through which LDP will select repair tunnels is as follows:

i. TI-LFA/RLFA using SR tunnels.
ii. RLFA using LDP if **remote-lfa** and **augment-route-table** are enabled under within the IGP **loopfree-alternates** context.
iii. Basic LFA

The **augment-route-table** command attaches RLFA-specific information to route-table entries that are necessary for LDP to program repair tunnels towards a PQ node using a specific neighbour.

10 Flexible-Algorithm

By default, an IGP-computed path is based on the shortest IGP metric, but frequently these paths are accompanied by traffic engineered paths that are used either strategically or on an ad-hoc basis to meet the requirements of the network. These traffic engineered paths are facilitated by RSVP-TE or SR-TE, both of which perform source-routing based on a given set of metrics and constraints. In many networks this works well, but for some operators the overhead of traffic engineering in this manner is perceived as complex or costly.

This chapter provides an overview of the operation of Flex-Algorithm and how it is applicable to Segment Routing (SR). Worked examples use IS-IS and SR-MPLS but will indicate where the OSPF implementation deviates. Flex-Algorithm with SRv6 is discussed in chapter 12.

Overview

Flexible Algorithm (or Flex-Algorithm) [39] provides a mechanism for IGPs to compute constraint-based paths across a domain using extensions to IS-IS and OSPF to advertise TLVs containing one or more Flexible Algorithm Definitions (FAD). Each FAD is associated with a numeric identifier and identifies a set of metrics and constraints that should be used to calculate the best path along the constrained topology. When used with SR-MPLS, one or more Prefix Node-SIDs can be associated with a Flex-Algorithm identifier, thereby providing a level of traffic engineering without any associated control plane overhead or additional label stack imposition. The classic SPF technology used for shortest path calculation is referred to as algorithm 0. SR-OS supports Flex-Algorithm for IS-IS and OSPFv2 and can use either SR-MPLS or SRv6 as the forwarding plane. In addition to algorithm 0, up to seven additional new flexible algorithms can be supported.

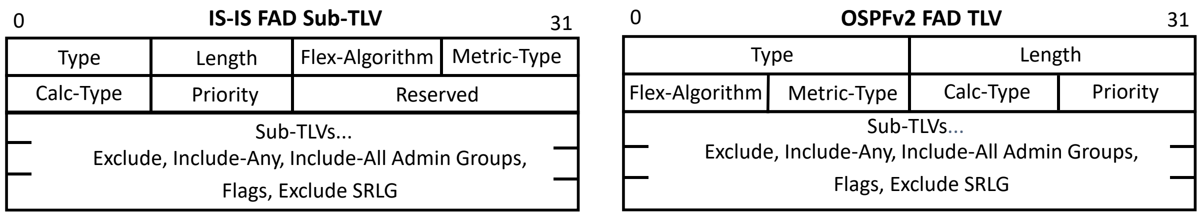
Flexible Algorithm Definition

The Flexible Algorithm Definition (FAD) is the construct that identifies how a path for a particular Flex-Algorithm should be computed, and consists of three components:

- A calculation-type
- A metric-type
- A set of constraints, such as include or exclude statements

To guarantee loop-free forwarding for paths computed with a given Flex-Algorithm, all routers that participate in that Flex-Algorithm must have a common understanding its definition and how it should work. In IS-IS the definition of the Flex-Algorithm is advertised using the Flexible Algorithm Definition sub-TLV, which is a sub-TLV of the Router Capability TLV. In OSPF The Flexible Algorithm Definition is a top-level TLV of the RI LSA. Both have area flooding scope. Each has a slightly different structure but carry fields containing the same information to define the FAD.

Figure 10-1: IS-IS and OSPF FAD Sub-TLVs



The Type and Length fields are self-explanatory. The Flex-Algorithm field contains a numeric identifier in the range 128-255 that is associated with the FAD through configuration. The metric-type may currently be one of IGP metric (0), Min Unidirectional Link Delay (1), or TE Default Metric (2). The Calc-Type field contains a value from 0-127 identifying the IGP algorithm type, such as shortest path (0). One or more sub-TLV fields may be present to specify 'colours' that are used to include or exclude links during the Flex-Algorithm path computation. These are encoded using Exclude Admin Group, Include-any Admin Group, Include-all Admin-Group, and Exclude SRLG sub-TLVs.

The sub-TLV field may also carry a Flags sub-TLV. Currently only the M-flag (Prefix Metric) is defined, and when set it indicates that the Flex-Algorithm Prefix Metric (FAPM) sub-TLV must be advertised with the prefix. The FAPM is not a sub-TLV of the FAD, but rather a sub-TLV of the IS-IS Extended IP Reachability TLV and OSPF Extended Prefix TLV and is intended to assist with inter-area and inter-domain Flex-Algorithm path calculations. Any IGP shortest-path tree calculation is limited to a single area, and the same applies to Flex-Algorithm. To allow for inter-area or inter-domain Flex-Algorithm calculations, the FAPM sub-TLV can be attached to IS-IS Extended IP Reachability TLVs or OSPF Extended Prefix TLVs that are advertised between areas or domains. The FAPM sub-TLV contains the metric equivalent to the metric of the redistributing router to reach the prefix. If the FAD Flags sub-TLV has the M-flag set, then the FAPM must be used when calculating prefix reachability for inter-area and inter-domain prefixes.

Only a subset of the routers participating in each Flex-Algorithm need to advertise the definition of the Flex-Algorithm. However, every router that is part of the intended Flex-Algorithm topology should express its willingness to participate in the Flex-Algorithm through configuration. If a router is not configured to participate in a particular Flex-Algorithm, it should ignore FAD TLV/Sub-TLV advertisements for that Flex-Algorithm.

Application-Specific Link Attributes

Advertisement of link attributes for the purpose of traffic engineering was initially introduced for IS-IS in [42] and for OSPF in [43]. Each included a number of sub-TLVs encoded within the IS-IS Extended IS Reachability TLV and OSPF Opaque LSA Link TLV, such as admin group, TE default metric, maximum link bandwidth, and unreserved bandwidth. They were both updated for the use of Extended Admin Groups [44] which increased the size of the admin group sub-TLV thereby allowing for advertisement of more than the standard 32 admin groups per link. Both were again further updated by the introduction of traffic engineering metric extensions [45] and [46]. These updates added additional sub-TLVs to the existing sub-TLVs such as unidirectional link delay, unidirectional link loss, and unidirectional available bandwidth. All of these traffic engineering extensions have been widely deployed for RSVP-TE purposes.

More recently, other applications that also make use of traffic engineering link attributes have been defined, such as SR, Loop Free Alternates (LFA), and Flex-Algorithm. If one or more of these applications co-exist it may be desirable to unambiguously indicate which traffic engineering attributes apply to which application. Their requirements may differ on a link-to-link basis, or two applications may not be fully congruent; for example, SR may not be fully deployed network-wide. For these reasons, Flex-Algorithm specifies the use of Application Specific Link Attributes (ASLAs). ASLAs allow for multiple sets of link attributes to be advertised for a given link, and also to independently associate a set of link attributes with one or more applications.

IS-IS extensions for ASLAs [40] define two new sub-TLVs of the Extended IS Reachability TLV, the ASLA sub-TLV and the Application-Specific Shared Risk Link Group (SRLG) sub-TLV:

- The ASLA sub-TLV specifies the applicable applications and associates the corresponding TE link attributes with those applications using the same standard format outlined above.
- The Application-Specific SRLG sub-TLV specifies the applications, encodes link identifier information such as interface address, neighbor address, and link local/remote identifiers and then associates SRLG information with them.

OSPF extensions for ASLAs [41] specify the ASLA sub-TLV which is carried in an Extended Link Opaque LSA as a sub-TLV of the of the OSPFv2 Extended Link TLV:

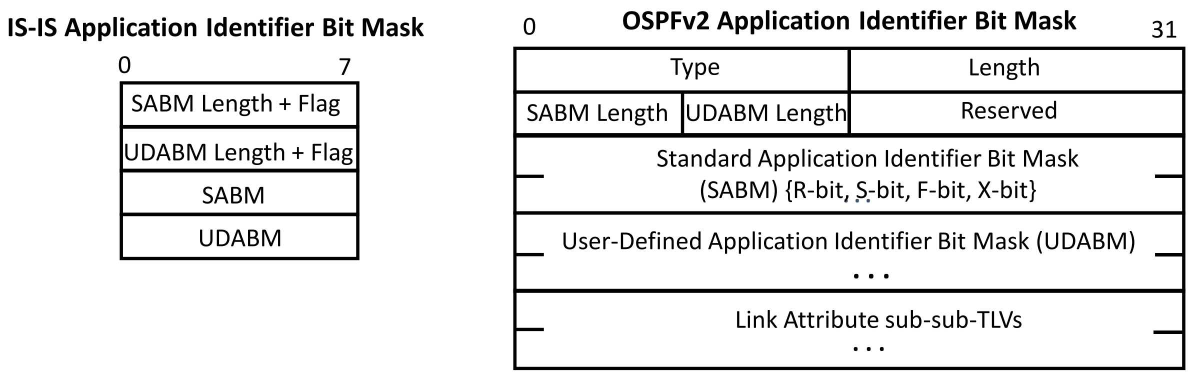
- Like it's IS-IS counterpart, the ASLA sub-TLV specifies the applicable applications and associates the corresponding TE link attributes with those applications, again using the same standard format outlined above.
- Unlike IS-IS however, OSPF does not require an additional Application-Specific SRLG sub-TLV as an SRLG sub-TLV is carried as a sub-TLV of the ASLA sub-TLV.

SR-OS will advertise Application-Specific SRLG information but does not currently use SRLG TLVs/sub-TLVs for computing SRLG-diverse paths. Support for LFA (primary/backup) SRLG diversity for Flex-Algorithm is provided using locally configured LFA policies as described in chapter 9.

Each of the IS-IS and OSPF ASLA sub-TLVs as well as the IS-IS Application-Specific SRLG TLV are advertised with an Application Identifier Bit Mask that identifies the application(s) associated with a particular advertisement. Two bit masks are available for use. One bit mask is used for standard applications (SABM), where the definition of each bit is controlled by IANA. A second bit mask is used for User Defined Applications (UDABM) and allows for future non-standard extensibility. The SABM field currently defines 4 bits to identify applications. The R-bit (bit 0) is used to specify RSVP-TE, the S-bit (bit 1) is used to specify SR, the F-Bit (bit 2) is used to specify LFA, and the X-bit (bit 3) is used to specify Flex-Algorithm.

The UDABM Length + Flag field contains a single R-flag, which is reserved for future use.

Figure 10-2: IS-IS and OSPF Application Identifier Bit Mask


IS-IS Application Identifier Bit Mask
0
7
SABM Length + Flag
UDABM Length + Flag
SABM
UDABM
OSPFv2 Application Identifier Bit Mask
0
31
Type
Length
SABM Length
UDABM Length
Reserved
Standard Application Identifier Bit Mask (SABM) {R-bit, S-bit, F-bit, X-bit}
User-Defined Application Identifier Bit Mask (UDABM)
Link Attribute sub-sub-TLVs

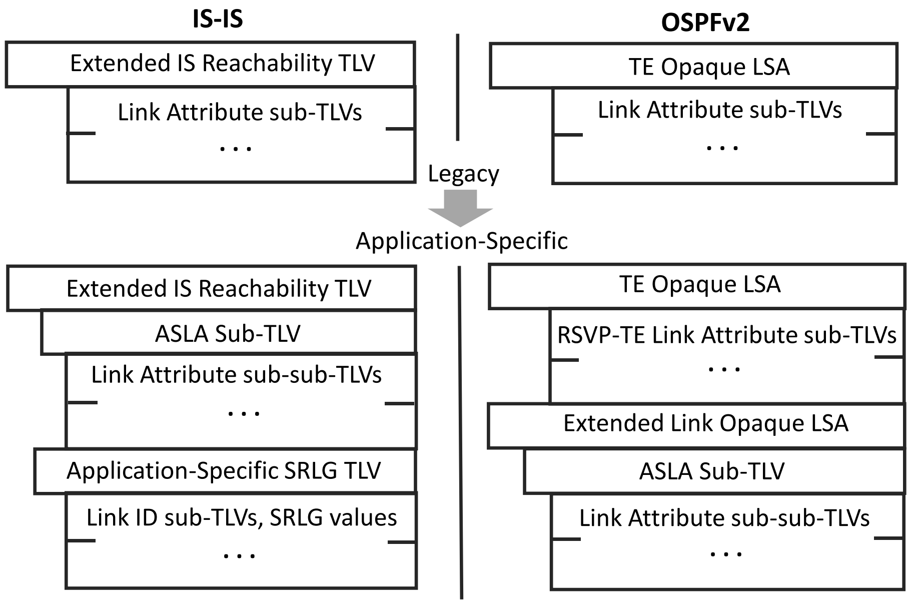
The ASLA extensions for OSPF support RSVP-TE as one of the supported applications, however, it recommends that all link attributes for RSVP-TE continue to be advertised in the existing TE Opaque LSA to maintain backwards compatibility. For IS-IS, the SABM Length + Flag field contains a single L-flag, known as the ‘Legacy’ flag. All IS-IS link attributes are carried as ASLA sub-TLVs or AS-specific SRLG sub-TLVs unless the L-flag is set in the Application Identifier Bit Mask. When set, all the applications specified in the bit mask must use the legacy traffic engineering advertisements for the corresponding link. That is, link attributes should be carried as sub-TLVs of the Extended IS Reachability TLV rather than sub-sub-TLVs of the ASLA sub-TLV.

In both cases, the use of legacy encoding of link attributes allows for a level of backward compatibility such that legacy advertisements may continue to be used as long as one of the following conditions exist:

- Only RSVP-TE is deployed, or
- Only SR /LFA is deployed, or
- A combination of RSVP-TE and SR/LFA is deployed but the set of links that each application uses are fully congruent.

Figure 10-3 summarises the transition from legacy encoding of traffic engineering link attributes into Application-Specific Link Attributes.

Figure 10-3: Transition from Legacy Link Attribute Encoding to ASLAs

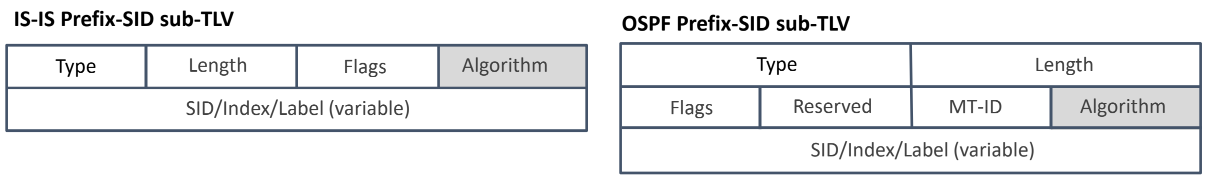

IS-IS
OSPFv2
Extended IS Reachability TLV
Link Attribute sub-TLVs
. . .
TE Opaque LSA
Link Attribute sub-TLVs
. . .
Legacy
Application-Specific
Extended IS Reachability TLV
ASLA Sub-TLV
Link Attribute sub-sub-TLVs
. . .
Application-Specific SRLG TLV
Link ID sub-TLVs, SRLG values
. . .
TE Opaque LSA
RSVP-TE Link Attribute sub-TLVs
. . .
Extended Link Opaque LSA
ASLA Sub-TLV
Link Attribute sub-sub-TLVs
. . .

Applicability of Flex-Algorithm to SR

A router may use various algorithms when calculating reachability to other nodes or prefixes attached to those nodes. To advertise the algorithms that the router can support IS-IS uses SR-Algorithm Sub-TLV carried as part of the Router Capabilities TLV while OSPF uses the SR-Algorithm TLV carried as part of the RI Opaque LSA. By default, an SR-capable router will signal support only for algorithm 0 (metric-based SPF). To advertise participation for a particular Flex-Algorithm for SR, the Flex-Algorithm value must also be advertised in the SR-Algorithm TLV or Sub-TLV (covered in chapters 3 and 4).

When an SR-capable router advertises a Prefix SID it includes an SR-algorithm, and as such it is possible to associate a given Prefix SID with a given algorithm.

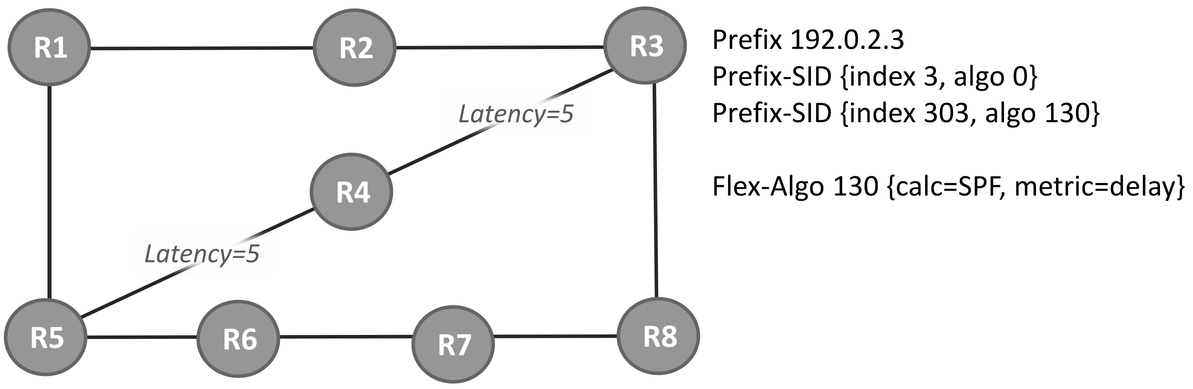
Figure 10-4: Prefix SID with Algorithm Field



For example, a router may advertise prefix P1 with Prefix SID {index=1, algorithm=0} and prefix P2 with Prefix SID {index=2, algorithm=128}. This would indicate to other SR routers that to reach prefix P1 the default metric-based SPF should be used to calculate the best path, and to reach prefix P2 Flex-Algorithm 128 and whatever that algorithm dictates should be used. Equally, in an SR-MPLS environment with an SR Global Block (SRGB) of {1000-1999}, a router may advertise prefix P1 with Prefix SID {index=1, algorithm=0} and also Prefix SID {index=101, algorithm=129}. This would indicate to other SR routers that when label 1001 is the active label to reach prefix P1, the default metric-based SPF should be used to calculate the best path, and when label 1101 is the active label Flex-Algorithm 129 should be used.

As an example, consider the topology in *Figure 10-5*. In this SR-MPLS domain all links have an IGP metric of 10 except for links R5-R4 and R4-R3 which both have an IGP metric of 30. Equally, all links have a unidirectional link latency of 10 milliseconds, except for links R5-R4 and R4-R3 which both have a unidirectional latency of 5 milliseconds. All routers use an SRGB of {1000-1999}.

Figure 10-5: Flex-Algorithm with SR


R1
R2
R3
R4
R5
R6
R7
R8
Latency=5
Latency=5
Prefix 192.0.2.3
Prefix-SID {index 3, algo 0}
Prefix-SID {index 303, algo 130}
Flex-Algo 130 {calc=SPF, metric=delay}

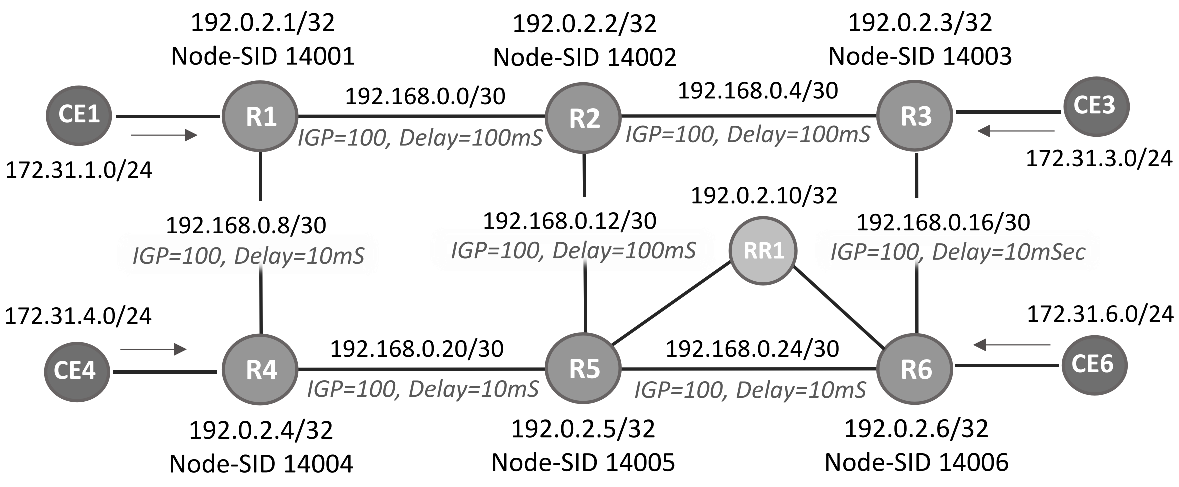
In addition to the default algorithm 0 (metric-based SPF), all routers participate in Flex-Algorithm 130 with FAD {calc-type=SPF, metric=delay, constraints=none}. Router R3 advertises prefix 192.0.2.3/32 with Prefix SID {index=3, algorithm=0} and Prefix SID {index=303, algorithm=130}. Router R5 has an SR-TE LSP provisioned with a destination of R3 (192.0.2.3) and a top (active) label of 1003. As a result, it is associated with algorithm 0 and uses the shortest path IGP metric R5-R1-R2-R3 to reach its destination. Router R5 is also provisioned with a second SR-TE LSP, again with a destination of R3 (192.0.2.3), but this time with a top (active) label of 1303. As such this second LSP is associated with Flex-algorithm 130 and uses the shortest path delay metric R5-R4-R3 to reach its destination.

Network Topology

To illustrate the use of Flex-Algorithm the network topology shown in *Figure 10-6* is used. Routers R1 to R6 and a Route-Reflector, RR1, are in a flat IS-IS Level 2 area. All routers belong to Autonomous System 64496 and Routers R1 to R6 peer in IBGP with RR1 as Route-Reflector clients to simplify BGP prefix advertisement. All IGP link metrics are 100. Unidirectional link delay is also configured, and all links have a delay of 10 milliseconds with the exception of links R1-R2, R2-R3, and R2-R5, which have a delay of 100 milliseconds. SR is enabled within the domain, and the associated Node-SIDs used as a baseline are shown. The SRGB in use is {12000-19999}.

There are a number of CE routers attached that advertise IPv4 prefixes to their adjacent routers which will be used to verify the allocation of Flex-Algorithms to services.

Figure 10-6: Test Topology for Flex-Algorithm



Advertising Link Delay

While illustrating the use of Flex-Algorithm the intention is to use shortest path delay as a metric. Before the delay metric for a given link can be advertised into the IGP a value for that metric needs to be derived. SR-OS provides the ability to populate the metric value either statically or dynamically. The static delay is a configured value for each interface, while the dynamic delay measurements use Simple Two-Way Active Measurement Protocol (STAMP) [49] test packets at interface level to gather results.

Output 10-1 shows the configuration of static delay values at R1 for the interfaces to R2 and R4. The **delay** is entered as an **if-attribute** under each interface as **static**, and has a value expressed in microseconds. As per the described network topology, the link R1-R2 has a link delay of 100 milliseconds, while the link R1-R4 has a link delay of 10 milliseconds.

Output 10-1: Configuration of Static Interface Delay

```
    router "Base" {
        interface "link-to-R2" {
            if-attribute {
                delay {
                    static 100000
                }
            }
        }
        interface "link-to-R4" {
            if-attribute {
                delay {
                    static 10000
                }
            }
        }
    }
```

The dynamic delay measurement approach requires significantly more configuration but has the obvious advantage of reacting dynamically to changes in transmission delay on a given link. The configuration parameters for delay measurement collection and reporting as well as the test criteria for the STAMP session are contained in a **measurement-template** as shown in *Output 10-2*.

The **unidirectional-measurement** command defines the method through which a unidirectional link delay measurement is calculated. If the adjacent nodes are synchronised to an extremely high level of accuracy (using the Precision Time Protocol (PTP) for example) the **actual** option can be used. This option calculates actual unidirectional delay using only the TWAMP/STAMP forward timestamps (T1 and T2). If the adjacent nodes do not have this level of synchronisation the **derived** option must be used. The **derived** option takes the round-tip delay and simply divides it by two. The **derived** option is the default setting hence this command would not be present in a configuration display but is included here as it is important to understand how unidirectional link delay is calculated.

The **delay** command allows the user to select either **minimum**, **maximum**, or **average** when comparing results against any configured thresholds. The **interval** command specifies the frequency of individual test probes and is a value of seconds with the default being one second. The **twamp-light** context is used to set the configuration parameters of the test probes, including **dscp** and **fc** as shown, but also parameters such as TTL, profile state, and destination UDP port, with port 862 being the default as defined for STAMP.

There are two measurement windows in a **measurement-template**, a **sample-window**, and an **aggregate-sample-window**. These two measurement windows have parameters that define the length of each measurement window and the thresholds that need to be crossed to trigger re-advertisement of IGP updates. Two measurement windows are provided to support a reporting hierarchy if required, and both windows include the same configuration options. The **sample-window** samples the results of individual test probes for a total length of the **interval** multiplied by its **multiplier** value. In the example below the **interval** is 2 and the **sample-window multiplier** is 5 hence the measurement window for sampling is 10 seconds. The **aggregate-sample-window** samples the results passed from the **sample-window** for a total length of its own configured **multiplier** value. In the example below a **multiplier** of 1 is used in the **aggregate-shaper-window** hence it will measure samples over the same interval as the **sample-window** (in other words, there is no real hierarchy or aggregation of results).

The **threshold** command compares the measurement window result to the last recorded delay measurement result and triggers an IGP update if it is crossed. The **threshold** can be configured as an **absolute** value expressed as microseconds, or a percentage increase using the relative command. The **window-integrity** command is used to specify a percentage of probe samples that must be present in the measurement interval for that window to be considered integral. If the number of probe samples falls below the configured value, the result is retained but not used for further processing.

Output 10-2: Measurement-Template for Dynamic Link Delay

```
    test-oam {
        link-measurement {
            measurement-template "direct-link" {
                admin-state enable
                description "STAMP probe for link delay"
                unidirectional-measurement derived
                delay avg
                interval 2
                twamp-light {
                    dscp af31
                    fc af
                }
                aggregate-sample-window {
                    multiplier 1
                    threshold {
                        relative 10
                    }
                }
                sample-window {
                    multiplier 5
                    window-integrity 80
                    threshold {
                        relative 10
                    }
                }
            }
        }
    }
```

Dynamic delay measurement is configured under an interface as an **if-attribute** in a similar manner to static delay. The **delay** context provides a sub-context for **dynamic** measurement and the **measurement-template** is applied here. There is also a context for configuring **twamp-light** source and destination addresses, which can be either **ipv4** or **ipv6**, but not both concurrently. When **ipv4** is used no source or destination address configuration is required if the interface primary address has a prefix length of 30 or 31. As the interfaces in the topology all use a prefix length of 30, all that is left is to put **twamp-light** into an **admin-state** of **enable**.

Note that the configuration shown in *Output 10-3* contains both a static and a dynamic delay. If required, both static and dynamic can be present on a given interface and the preferred delay selection mechanism can be configured, again at interface level, using the **delay-selection** command. The **delay-selection** command provides options for **static**, **dynamic**, **static-preferred**, or **dynamic-preferred**, with the latter being used here. Of course, if only static or dynamic delay exists on an interface, there is no requirement to configure a preferred mechanism.

Output 10-3: Applying the Link Measurement Template to an Interface

```
    router "Base" {
        interface "link-to-R2" {
            if-attribute {
                delay {
                    delay-selection dynamic-preferred
                    static 100000
                    dynamic {
                        measurement-template "direct-link"
                        twamp-light {
                            ipv4 {
                                admin-state enable
                            }
                        }
                    }
                }
            }
        }
    }
```

The final configuration step for enablement of dynamic delay measurement is to configure a session-reflector or responder for the TWAMP (or STAMP) probes. This is configured in the base routing instance under **twamp-light**. A **reflector** context contains parameters such as the **udp-port** on which it must listen (and which must align with the sender UDP port), and a **prefix** to constrain the list of IP addresses to which it will respond. Finally, the **reflector** must be put into an **admin-state** of **enable**.

Output 10-4: TWAMP/STAMP Reflector

```
    router "Base" {
        twamp-light {
            reflector {
                admin-state enable
                description "STAMP Link Measurement Reflector"
                udp-port 862
                prefix 192.168.0.0/24 {
                }
            }
        }
    }
```

The results of the link-measurement dynamic delay probes can be verified using the “show test-oam link-measurement interface <interface-name> detail” command. The objective here however is to flood the measurement results into the IGP for use with Flex-Algorithm and the process for doing that is covered in the next section.

Flex-Algorithm Configuration

There are a number of steps required to configure and enable the use of Flex-Algorithm:

a. Enable the use of Application-Specific Link Attributes (ASLAs)
b. Configure and advertise the Flex-Algorithm Definition
c. Configure Flex-Algorithm participation
d. Configure a Flex-Algorithm Prefix Node-SID
e. Traffic steering using Flex-Algorithm

These steps are covered in the following sub-sections.

Application-Specific Link Attributes

Flex-Algorithm specifies the use of Application-Specific Link Attributes (ASLAs) for the advertisement of traffic engineering information. Under the IS-IS instance the **traffic-engineering** command should be set to **true**, and within the **traffic-engineering-options** context **application-link-attributes** can be enabled. In IS-IS the **application-link-attributes** command has an optional **legacy** argument, which, when configured, allows link attributes to be encoded in the legacy manner as sub-TLVs of the Extended IS Reachability TLV rather than being encoded as (sub-)sub-TLVs of the ASLA sub-TLV.

Output 10-5: Enabling ASLA's

```
    router "Base" {
        isis 0 {
            traffic-engineering true
            traffic-engineering-options {
                application-link-attributes {
                }
            }
        }
    }
```

When enabling ASLAs for OSPF, the configuration parameters are largely the same as IS-IS, but as RSVP-TE and SR-TE may use different topologies, different configuration options exist for ASLA and legacy. When using OSPF the **traffic-engineering-options** context has an **sr-te** sub-context that has a **legacy** option or an **application-link-attributes** option. When **legacy** is selected, traffic engineering information for MPLS enabled SR links is advertised using Opaque LSAs. When the **application-link-attributes** option is selected traffic engineering information is advertised using the new ASLA sub-TLVs. If **sr-te** is not selected at all the conventional behaviour exists where RSVP-TE information is advertised in Opaque LSAs. *Table 10-1* summarises the traffic engineering flooding behaviour in OSPF with the various configuration parameters.

Table 10-1: TE Advertisement Options for OSPF

IGP Config	**ospf>traffic-eng**	**ospf>traffic-eng ospf>te-opts>no sr-te**	**ospf>traffic-eng ospf>te-opts>sr-te legacy**	**ospf>traffic-eng ospf>te-opts>sr-te application-link-attributes**
MPLS + RSVP	TE Opaque	TE Opaque	TE Opaque	TE Opaque
MPLS + SR	-	-	TE Opaque	ASLA (SR-TE)
MPLS + RSVP + SR	TE Opaque	TE Opaque	TE Opaque	TE Opaque (RSVP) and ASLA (SR-TE)

Output 10-6 provides an illustration of how ASLAs are advertised as sub-TLVs of the Extended IS Reachability TLV. The output is taken at R1 and is truncated to include only the IS neighbour R2. Within the Extended IS Reachability TLV there are three Application-Specific Link Attribute sub-TLVs. The first is non-legacy (L-bit is not set) and has a SABM field that has the R-bit and S-bit set to indicate these attributes are specific to RSVP-TE and SR respectively. The link attributes include Max Link Bandwidth and TE Metric. The second is non-legacy and has a SABM field with R-bit set, indicating that the intended application is RSVP-TE, and contains reservable and unreserved bandwidth link attributes. The final ASLA sub-TLV is again non-legacy and has the X-bit set to signify that this is specific only to Flex-Algorithm. This sub-TLV contains the Delay and TE Metric link attributes.

Note that SR-OS will not advertise an ASLA sub-TLV for Flex-Algorithm until it is configured and enabled. As that is covered in the following section this output can be considered somewhat out-of-sequence.

Output 10-6: Application-Specific Link Attribute Advertisement in IS-IS

```
A:admin@R1# show router isis database R1.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

Displaying Level 2 database
-------------------------------------------------------------------------------
LSP ID    : R1.00-00                                   Level     : L2
Sequence  : 0x4e                 Checksum  : 0x4b04    Lifetime  : 63801
Version   : 1                    Pkt Type  : 20        Pkt Ver   : 1
Attributes: L1L2                 Max Area  : 3         Alloc Len : 1492
SYS ID    : 1920.0000.2001       SysID Len : 6         Used Len  : 627
  ---[ snip ]---
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 143
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
    TE APP LINK ATTR    :
      SABML-flag:Non-Legacy SABM-flags:R S
        MaxLink BW: 10000000 kbps
        TE Metric : 100
    TE APP LINK ATTR    :
      SABML-flag:Non-Legacy SABM-flags:R
        Resvble BW: 10000000 kbps
        Unresvd BW:
            BW[0] : 10000000 kbps
            BW[1] : 10000000 kbps
            BW[2] : 10000000 kbps
            BW[3] : 10000000 kbps
            BW[4] : 10000000 kbps
            BW[5] : 10000000 kbps
            BW[6] : 10000000 kbps
            BW[7] : 10000000 kbps
    TE APP LINK ATTR    :
      SABML-flag:Non-Legacy SABM-flags:   X
        Delay Min : 100908 Max : 100908
        TE Metric : 100
    Adj-SID: Flags:v4BVL Weight:0 Label:524273
    Adj-SID: Flags:v6BVL Weight:0 Label:524284
  ---[ snip ]---
```

Flex-Algorithm Definition (FAD) and Participation

The first step is to define the Flex-Algorithm Definition (FAD) and advertise that FAD into the IGP, which in this example is IS-IS. This example of a FAD uses a metric-type of delay with no other constraints. As previously described, not all participating routers need to advertise the actual FAD; only their participation in it. Therefore, for this illustration routers R1 and R3 are used to advertise the FAD and the same configuration shown in *Output 10-7* is applied to both routers. Firstly, the FAD is created with a name under the **flexible-algorithms-definition** context. Once the **flex-algo** context has been created, **the metric-type** (IGP, TE metric, delay) and any other constraints (include-all, include-any, exclude) can be configured within it. It is also possible to configure a priority value for the **flex-algo** in the range 0-255.

In the event that multiple FAD advertisements are received the highest priority will be selected. If priorities are equal, then the FAD advertised by highest router ID is selected. The default priority is 100.

Output 10-7: FAD Definition

```
    routing-options {
        flexible-algorithm-definitions {
            flex-algo "Flex-Algo-128" {
                admin-state enable
                description "Flex-Algo for Delay Metric"
                metric-type delay
            }
        }
    }
```

Once the FAD has been defined it can be advertised into IS-IS. This is done under the **flexible-algorithms** context within the IS-IS instance. The Flex-Algorithm must initially be assigned a numeric identifier in the range 128-255, after which the **advertise** command is used to reference and advertise the previously configured FAD 'Flex-Algo-128'. The **participate** command is used to configure participation for the specific Flex-Algorithm and should be enabled on all routers that are part of this Flex-Algorithm topology. Within the Flex-Algorithm **loopfree-alternates** may be enabled, and if it is the LFA SPF will use the same Flex-Algorithm topology as that used to calculate the primary path. Also note that LFA settings (such as TI-LFA, Remote-LFA) within a Flex-Algorithm are inherited from the base algorithm 0 LFA configuration. Finally, for IS-IS **micro-loop-avoidance** may be enabled. Like LFA, any configuration parameters like the **fib-delay** timer are inherited from the base algorithm 0 micro-loop avoidance configuration. Flex-Algorithm aware micro-loop avoidance is not currently supported in OSPF.

Output 10-8: FAD Advertisement and Participation

```
    router "Base" {
        isis 0 {
            flexible-algorithms {
                admin-state enable
                flex-algo 128 {
                    participate true
                    advertise "Flex-Algo-128"
                    loopfree-alternate { }
                    micro-loop-avoidance { }
                }
            }
        }
    }
```

As R1 and R3 both advertise the FAD together with their willingness to participate in the Flex-Algorithm, what remains is to configure other routers in the domain to participate in the same Flex-Algorithm. For completeness, *Output 10-9* shows the configuration applied to routers R2, R4, R5, and R6 as they are intended to be part of the Flex-Algorithm 128 topology. The configuration is essentially the same as the configuration applied to R1 and R3, with the exception of the **advertisement** command which is not present. It is not applied to the Route-Reflector as this is a control-plane-only function which is not intended to be part of the data-plane and is therefore not part of the topology. Any router not indicating its willingness to participate in the algorithm is not considered part of that Flex-Algorithm topology.

Output 10-9: FAD Participation

```
    router "Base" {
        isis 0 {
            flexible-algorithms {
                admin-state enable
                flex-algo 128 {
                    participate true
                    loopfree-alternate { }
                    micro-loop-avoidance { }
                }
            }
        }
    }
```

Once the FAD with Flex-Algorithm identifier 128 has been advertised and all routers have signalled their willingness to participate in this Flex-Algorithm, it can be validated with router outputs. *Output 10-10* shows the pertinent parts of the IS-IS LSP advertised by R1. Within the Router Capabilities TLV, there are two notable additions. The SR-Algorithm Sub-TLV shows the default metric-based SPF (algorithm 0), but now additionally has algorithm 128, showing its willingness to participate in this algorithm. There is now also a FAD Sub-TLV containing the definition of Flex-Algorithm 128 with metric-type of delay. Note also that the FAD has a Flags sub-TLV with the M-flag set, indicating that the FAPM must be used when calculating prefix reachability for inter-area and inter-domain prefixes.

Output 10-10: IS-IS Advertisement

```
A:admin@R1# show router isis database R1.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

Displaying Level 2 database
-------------------------------------------------------------------------------
LSP ID    : R1.00-00                                Level     : L2
Sequence  : 0x4e                Checksum  : 0x4b04   Lifetime  : 63801
Version   : 1                   Pkt Type  : 20       Pkt Ver   : 1
Attributes: L1L2                Max Area  : 3        Alloc Len : 1492
SYS ID    : 1920.0000.2001      SysID Len : 6        Used Len  : 627

TLVs :
   ---[ snip ]---
  Router Cap : 192.0.2.1, D:0, S:0
    TE Node Cap : B E M  P
    SR Cap: IPv4 MPLS-IPv6
       SRGB Base:12000, Range:8000
    SR Alg: metric based SPF, 128
    Node MSD Cap: BMI : 12 ERLD : 15
    FAD Sub-Tlv:
        Flex-Algorithm   : 128
        Metric-Type      : delay
        Calculation-Type : 0
        Priority         : 100
        Flags: M
    S-BFD discr : 524288
   ---[ snip ]---
```

Assigning a Prefix Node-SID to a Flex-Algorithm

At each egress node a Prefix Node-SID must be assigned to each Flex-Algorithm in use. This will be advertised as a Prefix SID sub-TLV that will contain (among other things) the algorithm that should be used to reach the associated Prefix Node-SID. The Node-SID is taken from the generic SRGB; no special or dedicated label space is required. In the following example R3 is the egress router and the relevant configuration is shown in *Output 10-11.* Under the IS-IS system interface context the Node-SID label assigned to algorithm 0 is 14003 and is generic SR configuration. Within the same context a sub-context is created for **flex-algo 128** for which an **ipv4-node-sid label** of 17003 is assigned.

Output 10-11: Assigning a Prefix Node-SID to a Flex-Algorithm

```
    router "Base" {
        isis 0 {
            interface "system" {
                ipv4-node-sid {
                    label 14003
                }
                flex-algo 128 {
                    ipv4-node-sid {
                        label 17003
                    }
                }
            }
        }
    }
```

Once configured, the additional Prefix-SID advertisement can be viewed in R3's advertised IS-IS LSP.

Output 10-12: Prefix-SID with Flex-Algo 128

```
A:admin@R3# show router isis database R3.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

Displaying Level 2 database
-------------------------------------------------------------------------------
LSP ID    : R3.00-00                                    Level     : L2
Sequence  : 0x5110                 Checksum  : 0xa0bf   Lifetime  : 64813
Version   : 1                      Pkt Type  : 20       Pkt Ver   : 1
Attributes: L1L2                   Max Area  : 3        Alloc Len : 1492
SYS ID    : 1920.0000.2003         SysID Len : 6        Used Len  : 674

TLVs :
  ---[ snip ]---
  TE IP Reach   :
    Default Metric  : 0
    Control Info:   S, prefLen 32
    Prefix   : 192.0.2.3
    Sub TLV   :
      AttrFlags: NE
      Prefix-SID Index:2003, Algo:0, Flags:NnP
      Prefix-SID Index:5003, Algo:128, Flags:NnP
    Default Metric  : 100
  ---[ snip ]---
```

The flex-algorithm Prefix-SID also be viewed using the command in *Output 10-13* which includes the relevant flex-algo extension. Note that the index is 5003 and with an SRGB start-label of 12000, equates to the configured label value of 17003.

Output 10-13: Prefix-SID Entry for R3 using Flex-Algo 128

```
A:admin@R1# show router isis prefix-sids ip-prefix-prefix-length 192.0.2.3 algo 128

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                         SID        Lvl/Typ    SRMS   AdvRtr
                                Shared     Algo       MT     Flags
-------------------------------------------------------------------------------
192.0.2.3/32                   5003       2/Int.      N     R3
                                   N.A.       128        0      NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 1 (1 unique)
-------------------------------------------------------------------------------
```

The IS-IS route at R1 towards the prefix 192.0.2.3/32 (R3) for Flex-Algo 128 is shown in the following output, and two things are worthy of note. Firstly, the next-hop is shown as 192.168.0.10 (R4). This is because Flex-Algo 128 uses delay as a metric and the shortest path delay from R1 to R3 is using the path R1-R4-R5-R6-R3. Secondly, the metric is shown as 40000 which represents the value in microseconds of the afore-mentioned path.

Output 10-14: R1's IS-IS Route Towards R3 with Flex-Algo 128

```
A:admin@R1# show router isis routes ip-prefix-prefix-length 192.0.2.3/32 flex-algo flex-
algo-id 128

===============================================================================
Rtr Base ISIS Instance 0 Flex-Algo 128 Route Table
===============================================================================
Prefix[Flags]                  Metric     Lvl/Typ     Ver.  SysID/Hostname
  NextHop                                             MT     AdminTag/SID[F]
-------------------------------------------------------------------------------
192.0.2.3/32                   40000      2/Int.      163   R4
   192.168.0.10                                         0       0/5003[Nn*
-------------------------------------------------------------------------------
No. of Routes: 1 (1 path)
-------------------------------------------------------------------------------
```

The final step at R1 is to verify that the Prefix Node SID advertised by R3 with Algorithm 128 is present in the tunnel-table. *Output 10-15* shows the tunnel-table at R1 for R3 (192.0.2.3/32) and there are two entries. The first entry (tunnel ID 524300) is the default SR-ISIS tunnel calculated using algorithm 0. This has a next-hop of R2 (192.168.0.2) and metric of 200 representing the IGP cost of the path R1-R2-R3. The second entry (tunnel ID 524317) is calculated using Flex-Algorithm 128. It has a next-hop of R4 (192.168.0.10) and a metric of 40000 representing the accumulative delay metric (40msec) for the path R1-R4-R5-R6-R3. The presence of the [L] flag for both entries indicates that both tunnel-table entries have LFA backups available, noting that the backup path for Flex-Algorithm 128 would also have calculated using the Algorithm 128 topology.

Output 10-15: Tunnel-Table Entry for R3's Flex-Algo 128 Prefix-SID

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol isis

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32 [L]      isis (0)  MPLS  524300    11     192.168.0.2    200
192.0.2.3/32 [L]      isis (0)  MPLS  524317    11     192.168.0.10   40000
-------------------------------------------------------------------------------
```

Traffic Steering using Flex-Algorithm

The options available to steer traffic into one or more Flex-Algorithm SR LSPs differ between services that use SDP-based services and BGP/services that use the auto-bind-tunnel concept to resolve BGP next-hops.

Flex-Algorithm LSPs can be used for SDP-based services or BGP/services using auto-bind-tunnel wherever an SR-TE LSP or SR Policy path contains a single Prefix Node-SID assigned to a Flex-Algorithm identifier. *Output 10-16* provides an example of an SR-TE LSP configured at R1 towards R3. The LSP references a primary path named Flex-Algo-128, where that path has a single hop containing the label 17003. This is the label value that was previously allocated to Flex-Algorithm 128 at R3 (see *Output 10-11)*.

Output 10-16: SR-TE LSP Referencing a Flex-Algo Prefix-SID

```
    router "Base" {
        mpls {
            path "Flex-Algo-128" {
                admin-state enable
                hop 1 {
                    sid-label 17003
                }
            }
            lsp "R1-to-R3-FlexAlgo128" {
                type p2p-sr-te
                to 192.0.2.3
                max-sr-labels {
                    label-stack-size 1
                    additional-frr-labels 2
                }
                primary "Flex-Algo-128" {
                    admin-state enable
                }
            }
        }
    }
```

The SR-TE LSP is operationally up and present in the tunnel-table with tunnel ID 655369 and a has the maximum metric as is usual for explicit-routed SR-TE LSPs.

Output 10-17: Tunnel-Table at R1 for Flex-Algorithm SR-TE LSP

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol sr-te
```

```
===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          sr-te     MPLS  655369    8      192.0.2.3      16777215
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

The Flex-Algorithm aware SR-TE LSP can thereafter be referenced as an LSP within the SDP context, or it can be used by BGP/services that use auto-bind-tunnel with an appropriate **resolution-filter** configured to include SR-TE LSPs.

A second approach for traffic steering using the route-policy framework is available to BGP/services that use auto-bind-tunnel for resolution of BGP next-hops. To illustrate this approach a VPRN service is used, but the same methodology is applicable to BGP and other services that use auto-bind-tunnel for BGP next-hop resolution. The VPRN uses a **policy-statement** '*vrf-one-import*' as shown in *Output 10-18* to define the Route-Target communities that it will use for importing routes into the VRF. Within this **policy-statement** the **action** statement accepts routes with the **community name** '*vrf-one*' and also references a **flex-algo** identifier, which in this case is 128. In the example the flex-algo identifier is applied to all routes that are imported into the VRF, but the advantage of using the route-policy framework is that application of a **flex-algo** identifier can have per-Prefix granularity or can adopt the use of color communities to identify and steer traffic as described in chapter 7.

Output 10-18: Traffic Steering to Flex-Algo LSPs with Route-Policy

```
    policy-options {
        community "vrf-one" {
            member "target:64496:1" { }
        }
        policy-statement "vrf-one-import" {
            entry 10 {
                from {
                    community {
                        name "vrf-one"
                    }
                }
                action {
                    action-type accept
                    flex-algo 128
                }
            }
        }
    }
```

Within the VPRN service the **auto-bind-tunnel** context uses a **resolution-filter** that includes **sr-isis**. The auto-bind-tunnel context also contains the **allow-flex-algo-fallback true** command. When enabled, if a BGP next-hop cannot be resolved using the referenced Flex-Algorithm identifier, the system will fall back to an algorithm 0 topology to resolve the next-hop.

Output 10-19: Auto-Bind-Tunnel with Flex-Algo

```
    service {
        vprn "one" {
            bgp-ipvpn {
                mpls {
                    vrf-import {
                        policy ["vrf-one-import"]
                    }
                    auto-bind-tunnel {
                        resolution filter
                        allow-flex-algo-fallback true
                        resolution-filter {
                            sr-isis true
                        }
                    }
                }
            }
        }
    }
```

Using Flex-Algorithm with Admin Group Constraint

The configuration example used so far in this chapter has employed a metric-type of delay. For completeness, this section contains a second example using admin groups as a constraint.

When admin groups are used as a constraint the first step should be to apply the required admin group(s) to the relevant link(s). For the purpose of this example the link R1-R2 is associated with admin group “red”. Initially the admin group is configured as an **if-attribute** in the base router context and assigned an integer value in the range 0-31. The admin group is then assigned to each required interface as an **if-attribute** in the same manner that delay was previously configured. The following output is taken from R1 with a similar configuration applied at R2.

Output 10-20: Assigning Admin-Groups to an Interface

```
    routing-options {
        if-attribute {
            admin-group "red" {
                value 10
            }
        }
    }
    router "Base" {
        interface "link-to-R2" {
            if-attribute {
                admin-group ["red"]
            }
        }
```

Note: Flexible-Algorithm [39] permits the use of Admin Groups and Extended Admin Groups (EAGs) [44]. Admin Groups contain a 4-octet bit mask, where each set bit corresponds to a single admin group, thus allowing for support of 32 admin groups. EAGs remove this limit, and in fact have no fixed limit. SR-OS only supports Admin Groups, not EAGs. For backward compatibility, if EAGs are used by another vendor they must use only the first 32 colours in the EAG.

The next step is to configure the Flex-Algorithm Definition and participation. Firstly, the FAD is configured to **exclude** the **admin-group** "red" whilst the metric-type remains the default IGP metric. The following configuration is applied at R1 and R3.

Output 10-21: FAD to Exclude Admin-Group Red

```
    routing-options {
        flexible-algorithm-definitions {
            flex-algo "Flex-Algo-129" {
                admin-state enable
                description "Flex-Algo for excluding Admin-Group Blue"
                exclude {
                    admin-group "red" { }
                }
            }
        }
    }
```

In addition to the **exclude admin-group** constraint, there are options for **include-any** and **include-all** admin-groups. Include-any means that any link not configured with any of the specified admin-groups should be pruned. Include-all means that any link not configured with all of the specified admin-groups should be pruned.

The following step is to advertise the FAD and indicate willingness to participate in the Flex-Algorithm. Again, the following configuration is applied at R1 and R3, who both participate and advertise in Flex-Algorithm 129. Similar configuration is applied to R2, R4, R5, and R6, but without the **advertise** command since they only have a requirement to participate in the Flex-Algorithm and not advertise its definition.

Output 10-22: Advertise and Participate in Flex-Algo 129

```
    router "Base" {
        isis 0 {
            flexible-algorithms {
                admin-state enable
                flex-algo 129 {
                    participate true
                    advertise "Flex-Algo-129"
                    loopfree-alternate { }
                    micro-loop-avoidance { }
                }
            }
        }
    }
```

Finally, a Prefix Node SID is assigned to the Flex-Algorithm at the egress node(s). In the following example R3 is the egress router and label 18003 is assigned to Flex-Algorithm 129.

Output 10-23: Assign Prefix-SID to Flex-Algo 129

```
    router "Base" {
        isis 0 {
            interface "system" {
                flex-algo 129 {
                    ipv4-node-sid {
                        label 18003
                    }
                }
            }
        }
    }
```

The Prefix SID and associated Flex-Algorithm advertised by R3 is learnt at R1. As before, the SID is advertised as an index and with an SRGB of {12000-19999} represents label 18003.

Output 10-24: R3 Prefix-SID for Flex-Algo 129 Learnt at R1

```
A:admin@R1# show router isis prefix-sids ipv4-unicast ip-prefix-prefix-length 192.0.2.3/32
algo 129

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                            SID        Lvl/Typ    SRMS   AdvRtr
                                   Shared     Algo       MT     Flags
-------------------------------------------------------------------------------
192.0.2.3/32                      6003       2/Int.      N     R3
                                      N.A.       129        0      NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 1 (1 unique)
-------------------------------------------------------------------------------
```

The tunnel-table is also used to verify the Flex-Algorithm path towards the destination prefix. The following output shows the tunnel-table at R1 for R3 (192.0.2.3/32). In this output there are now three entries. The first entry (tunnel ID 524300) is the default SR-ISIS tunnel calculated using algorithm 0. This has a next-hop of R2 (192.168.0.2) and metric of 200 representing the IGP cost of the path R1-R2-R3. The second entry (tunnel ID 524320) is calculated using Flex-Algorithm 129. It has a next-hop of R4 (192.168.0.10) thus avoiding the R1-R2 link with admin-group red, and a metric of 400 representing the IGP metric for the path R1-R4-R5-R6-R3 (or R1-R4-R5-R2-R3 since R5 has ECMP to R3). Note that unlike entries one and three, entry two has no LFA backup denoted by the [L] flag. This is simply because from R1’s perspective if the link R1-R2 is pruned from the topology there is only one way out, which is through the link to R4. The final entry (tunnel ID 524317) is calculated using Flex-Algorithm 128. It has a next-hop of R4 (192.168.0.10) and a metric of 40000 representing the accumulative delay metric (40msec) for the path R1-R4-R5-R6-R3.

Output 10-25: Tunnel-Table at R1 With Flex-Algo 129 SR LSP

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol isis

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32 [L]      isis (0)  MPLS  524300    11     192.168.0.2    200
192.0.2.3/32          isis (0)  MPLS  524320    11     192.168.0.10   400
192.0.2.3/32 [L]      isis (0)  MPLS  524317    11     192.168.0.10   40000
-------------------------------------------------------------------------------
```

The Prefix Node-SID for Flex-Algorithm 129 is now available for carrying traffic. Methods for traffic steering into Flex-Algorithm LSPs have previously been described in this chapter and are therefore not repeated here.

Inter-Area Flex-Algorithm

To validate the use of Flex-Algorithm in an inter-area environment, the test topology illustrated in *Figure 10-6* is modified such that R1 and R4 are IS-IS Level-1 routers, while R2 and R5 are IS-IS Level-1/2 routers. Routers R3 and R6 remain Level-2 only routers.

```
R1--------R2--------R3
|         |         |
R4--------R5--------R6
<--- L1 ---><--- L2 --->
```

The previously configured Flex-Algorithm 128 (metric-type of delay) and Flex-Algorithm 129 (exclude admin-group red) are again used for the purpose of illustrating inter-area Flex-Algorithm path computations. Recall that R1 and R3 both advertise the FADs for these algorithms, and in this Level-1/2 inter-area scenario this is noteworthy simply because IS-IS FAD sub-TLVs are carried in Router Capability TLVs that only have area flooding scope; they are not redistributed between areas. (In other words, in this scenario R1 advertises the FAD within the Level-1 area, while R3 advertises the FAD within the Level-2 area.)

As previously described, when a FAD includes the M-flag (Prefix Metric), an L1/L2 router (or ASBR) must include the Flex-Algorithm Prefix Metric (FAPM) sub-TLV when advertising a prefix within an Extended IP Reachability TLV between areas, levels, or domains. The advertised metric should be equal to the metric to reach the prefix for a given Flex-Algorithm in the source area or domain. This allows a router in a different area/level/domain to include the FAPM when calculating prefix reachability for inter-area/domain prefixes and provides an optimal end-to-end path for a given Flex-Algorithm.

Note: SR-OS provides support for redistributing Prefix-SID information between IS-IS instances but does not currently support the redistribution of Prefix-SID information between OSPF instances.

When redistributing Prefix-SID information between IS-IS instances for Flex-Algorithm (algorithms 128-255), the source IS-IS (or redistributing) instance must be either configured as a level-1/2 router or have either the **iid-tlv** or the **standard-multi-instance** command set to **true**. The reason for this is because SR-OS uses the Route-Table Manager (RTM) to store information for exporting between IS-IS instances. By default, only information for algorithm 0 is stored in the RTM, hence redistribution of standard Prefix-SID information works as expected. This approach represents a reasonable compromise between availability of information and memory consumption. The configuration of one of the afore-mentioned commands will trigger SR-OS to store additional information into the RTM data structures to be used during redistribution of prefixes.

In the test topology, both R3 and R6 are assigned Node-SID labels for Flex-Algorithms 128 and 129. R3 is assigned label 17003 for algorithm 128 and 18003 for algorithm 129, whilst R6 is assigned label 17006 for algorithm 128 and 18006 for algorithm 129.

The first output shows R2's IS-IS Level-1 LSP as received by R1, truncated to show only the Extended IP Reachability TLV for R3 (192.0.2.3) and R6 (192.0.2.6). R3 has Prefix-SIDs associated with algorithm 0, algorithm 128, and algorithm 129, while R6 has Prefix-SIDs associated with algorithm 0 and algorithm 128. Flex-Algorithms 128 and 129 also have a FAPM sub-TLV (shown as `Prefix-Metric-FlexAlg`) containing the relevant metric for R2 to reach the destination prefix.

Output 10-26: Propagation of FAPM

```
A:admin@R1# show router isis database R2.00-00 level 1 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

Displaying Level 1 database
-------------------------------------------------------------------------------
LSP ID    : R2.00-00                                 Level     : L1
Sequence  : 0x5                  Checksum  : 0x4f2b  Lifetime  : 58125
Version   : 1                    Pkt Type  : 18      Pkt Ver   : 1
Attributes: L1L2                 Max Area  : 3       Alloc Len : 531
SYS ID    : 1920.0000.2002       SysID Len : 6       Used Len  : 531

TLVs :
  ---[ snip ]---
  TE IP Reach   :
    Default Metric  : 100
    Control Info: D S, prefLen 32
    Prefix   : 192.0.2.3
    Sub TLV   :
      AttrFlags: RNE
      Prefix-SID Index:2003, Algo:0, Flags:RNnP
      Prefix-SID Index:5003, Algo:128, Flags:RNnP
      Prefix-Metric-FlexAlg Algo:128, Metric:100000
      Prefix-SID Index:6003, Algo:129, Flags:RNnP
      Prefix-Metric-FlexAlg Algo:129, Metric:100
    Default Metric  : 200
    Control Info: D S, prefLen 32
    Prefix   : 192.0.2.6
    Sub TLV   :
      AttrFlags: RNE
      Prefix-SID Index:2006, Algo:0, Flags:RNnP
      Prefix-SID Index:5006, Algo:128, Flags:RNnP
      Prefix-Metric-FlexAlg Algo:128, Metric:110000
  ---[ snip ]---
```

The information from the FAPM sub-TLV advertised by R2 and R5 (Level-1/2 routers) allows the routers in the Level-1 area to construct optimal end-to-end paths with accumulative metrics. *Output 10-27* shows the tunnel-table at R1 for R3 (192.0.2.3). The first entry is the SR-ISIS LSP calculated with algorithm 0 and shows an IGP metric of 200 for the path R1-R2-R3. The second entry is the SR-ISIS LSP calculated with Flex-Algorithm 129, which uses a metric-type of IGP and excludes the R1-R2 link. It therefore has an IGP metric of 400 for the path R1-R4-R5-R6-R3 (or R1-R4-R5-R2-R3 as R5 has ECMP to R3). The final entry is the SR-ISIS LSP calculated with Flex-Algorithm 128 using a metric-type of delay. This entry has a metric of 40000 representing the delay metric for the path R1-R4-R5-R6-R3.

It can therefore be observed that optimal end-to-end paths can be calculated by redistributing prefixes with the FAPM sub-TLV and including that metric in the calculation for inter-area/domain prefixes.

Output 10-27: Tunnel-Table Entries for R3 at R1

```
A:admin@R1# show router tunnel-table 192.0.2.3/32 protocol isis

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.0.2.3/32          isis (0)  MPLS  524345    11     192.168.0.2    200
192.0.2.3/32          isis (0)  MPLS  524347    11     192.168.0.10   400
192.0.2.3/32          isis (0)  MPLS  524343    11     192.168.0.10   40000
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

By providing some well-structured extensions to IS-IS and OSPF, Flex-Algorithm provides a mechanism to achieve a level of traffic engineering without the requirement for a centralised controller. A traffic engineered path can be represented by a single Node-SID without the requirement to impose either a deep label stack for SR-MPLS, or numerous segments in the Segment Routing Header (SRH) for SRv6. Although the traffic engineering capabilities of Flex-Algorithm are limited compared to those available when using a centralised controller, it represents a reasonable trade-off between objective and complexity.

11 SR-MPLS with an IPv6 Control Plane

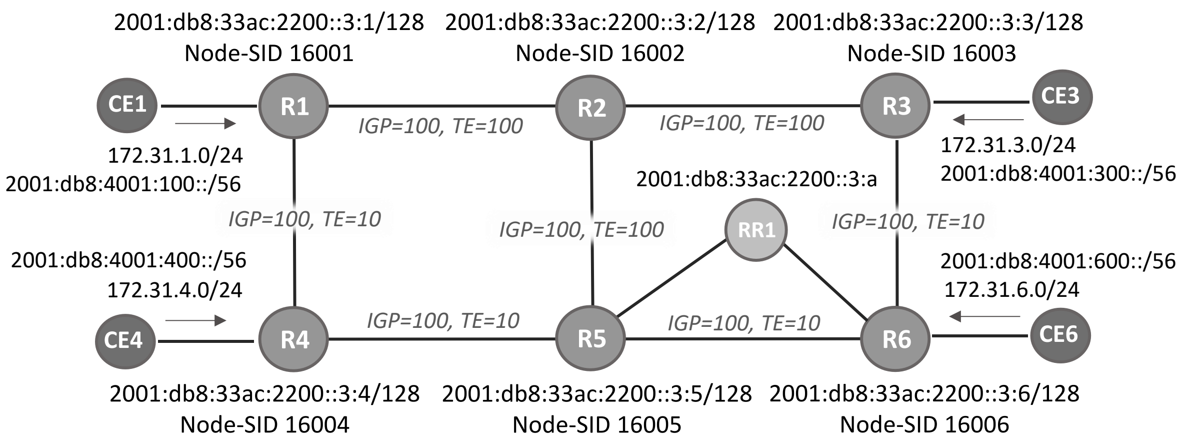
As outlined in the Segment Routing Architecture specification [12] SR can be instantiated on various data planes and introduces two, SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). With SR-MPLS, SR is directly applied to the MPLS architecture with no change to the forwarding plane. A segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack, and upon completion of a segment the related label is popped from the stack. This is of course known as SR-MPLS. With SRv6, SR is applied to the IPv6 architecture with a new type of routing header, known as the Segment Routing Header (SRH). A segment is encoded as an IPv6 address. An ordered list of segments is encoded as an ordered list of IPv6 addresses in the SRH. The active segment is indicated by the Destination Address of the packet. The next active segment is indicated by a pointer in the SRH. This is of course known as SRv6.

Having these two mechanisms to instantiate SR often leads to a misconception that in order to use SR with an IPv6 network it is necessary to adopt the use of SRv6. This is not the case. When the SR extensions were defined for IS-IS and OSPFv2/OSPFv3, they included provision for attaching Prefix-SIDs to IPv4 and IPv6 prefixes. Although Adjacency SIDs are not attached to a particular prefix but rather to an adjacency, the specifications gave provision for them to be identified as either IPv4 or IPv6 Adjacency-SID.

What this quite simply means is that it is perfectly valid to instantiate an SR-MPLS data plane over an IPv4 or an IPv6 network. To illustrate how this is achieved, the network topology in *Figure 11-1* is used. System interface addresses and network interfaces are all IPv4/IPv6 dual-stack, although the IPv4 addressing is not shown for the purpose of clarity. Dual-stack IPv4/IPv6 is not mandatory. SR-MPLS over a pure IPv6 network will work perfectly fine, although an IPv4 system address (Router-ID) is still a requirement for a number of protocols. Regardless, although IPv4 is present the illustration will use IPv6 addressing unless explicitly stated. Routers R1 to R6 are in a flat IS-IS Level 2 domain and the relevant IPv6 system addresses are shown. Link IGP metrics are 100 throughout, and TE metric is 10 with the exception of the R1-R2 and R2-R3 links where it is 100. SR is enabled with an SRGB of 12000-19999, and IPv6 Node-SIDs are shown in the figure. All routers reside in Autonomous System 64496 and routers R1 to R6 are Route-Reflector clients of RR1 for the VPN-IPv4, VPN-IPv6, EVPN, and BGP SR-Policy address families. All BGP sessions use IPv6 TCP transport.

There are a number of CE routers attached that advertise both IPv4 and IPv6 prefixes to their adjacent routers which will be used to validate the correct operation of the core topology in this configuration.

Figure 11-1: Test Topology for SR-MPLS with IPv6 Control Plane



IPv6 SR-MPLS Core

To enable shortest path SR-MPLS over an IPv6 network, the first requirement is to advertise IPv6 prefixes. For OSPF, this mandates the deployment of OSPFv3. For IS-IS that is used in this example, the **ipv6-routing native** command is added under IS-IS to enable IPv6 routing within Multi-Topology 0. This command allows advertisement of a given routers IPv6 interface addresses as IPv6 Reachability TLVs, together with any associated Prefix-SID sub-TLVs. It also enables additional IPv6 Adj-SID sub-TLVs to be attached to TE-IS Neighbour TLVs, hence one is present for IPv4, and another is present for IPv6. An IPv6 Node-SID is then assigned under the system interface using the **ipv6-node-sid label** in exactly the same manner as described for IPv4 in chapters 3 and 4. When configured on all routers in the domain it essentially enables the dissemination of IPv6 Node-SIDs and IPv6 Adj-SIDs. When **ipv6-routing native** is enabled, the IS-IS traffic-engineering extensions for IPv6 [37] are not present in advertised LSPs. The result of that means that there is no IPv6 TE Router-ID, and no IPv6 Interface Addresses carried in Extended IS Reachability TLVs. To enable these to be advertised the **traffic-engineering-options** context has the **ipv6** option set to **true**. Without the presence of this information, running a CSPF for an IPv6 destination would not be possible.

Output 11-1: IS-IS and Shortest Path SR Configuration for IPv6

```
        isis 0 {
            ipv6-routing native
            traffic-engineering-options {
                ipv6 true
            }
            interface "system" {
                ipv6-node-sid {
                    label 16001
                }
            }
        }
```

Output 11-2 shows the pertinent parts of the IS-IS LSP advertised by R1. The supported protocols TLV now show IPv4 and IPv6. There is an IPv6 TE Router-ID TLV showing R1’s IPv6 system address in addition to the IPv4 Router ID. The Extended IS Reachability TLV contains the IPv6 interface address and the IPv6 neighbour address. It also includes two Adj-SIDs and as described in chapter 3 the Address-Family Flag (F-Flag) in the Adj-SID sub-TLV indicates the address family in use. If unset the Adj-SID is used when forwarding IPv4-encapsulated traffic to the neighbour (shown as `Flags:v4`). If set, the Adj-SID is used when forwarding IPv6-encapsulated traffic to the neighbour (shown as `Flags:v6`). The IPv6 Reachability TLV contains all of the IPv6 interface addresses belonging to R1, notably the system address carries a Prefix-SID sub-TLV with its Node-SID advertised as an index of the SRLG start-label (12000 + 4001 = 16001).

Output 11-2: R1 IS-IS LSP with IPv6 Native Routing

```
A:admin@R1# show router isis database R1.00-00 level 2 detail

TLVs :
... [snip]...
  Supp Protocols:
    Protocols     : IPv4
    Protocols     : IPv6
  IS-Hostname   : R1
  Router ID   :
    Router ID   : 192.0.2.1
  TE Router ID v6  :
    Router ID   : 2001:db8:33ac:2200::3:1
...[snip]
  TE IS Nbrs   :
    Nbr   : R4.00
    IPv6 Addr : 2001:db8:33ac:2200::4:9
    Nbr IPv6  : 2001:db8:33ac:2200::4:a
... [snip] ...
    Adj-SID: Flags:v4BVL Weight:0 Label:524272
    Adj-SID: Flags:v6BVL Weight:0 Label:524275
  IPv6 Reach:
    Metric: ( IS) 0
    Prefix   : 2001:db8:33ac:2200::3:1/128
    Sub TLV   :
      AttrFlags: NE
      Prefix-SID Index:4001, Algo:0, Flags:NnP
    Metric: ( I ) 100
    Prefix   : 2001:db8:33ac:2200::4:0/126
    Metric: ( I ) 100
    Prefix   : 2001:db8:33ac:2200::4:8/126
... [snip] ...
```

With the above configuration in place IPv6 Node-SIDs are distributed across the SR domain providing a shortest path ECMP-aware IPv6 SR tunnel to every remote node.

Output 11-3: IPv6 Node-SIDs shown at R1

```
A:admin@R1# show router isis prefix-sids ipv6-unicast

===============================================================================
Rtr Base ISIS Instance 0 Prefix/SID Table
===============================================================================
Prefix                            SID        Lvl/Typ    SRMS   AdvRtr
                                   Shared                MT     Flags
-------------------------------------------------------------------------------
2001:db8:33ac:2200::3:1/128       4001       2/Int.      N     R1
                                       No                   0      NnP
2001:db8:33ac:2200::3:2/128       4002       2/Int.      N     R2
                                       N.A.                 0      NnP
```

```
2001:db8:33ac:2200::3:3/128       4003       2/Int.      N      R3
                                      N.A.                  0       NnP
2001:db8:33ac:2200::3:4/128       4004       2/Int.      N      R4
                                      N.A.                  0       NnP
2001:db8:33ac:2200::3:5/128       4005       2/Int.      N      R5
                                      N.A.                  0       NnP
2001:db8:33ac:2200::3:6/128       4006       2/Int.      N      R6
                                      N.A.                  0       NnP
-------------------------------------------------------------------------------
No. of Prefix/SIDs: 6 (6 unique)
-------------------------------------------------------------------------------
```

Some degree of traffic engineering is possible in SR-OS when using SR-MPLS with an IPv6 control plane although it is reasonable to say that feature coverage is not at parity with IPv4. Although PCEP sessions may use the IPv6 address family as TCP transport, there is currently no support for encoding of IPv6 objects within the PCEP protocol. Consequently, there is no support for PCC-initiated or PCE-initiated SR-TE LSPs when using an IPv6 control plane. It is however possible with IS-IS to use distributed SR-TE with either a local CSPF or explicit path definition. For example, *Output 11-4* shows the configuration of an SR-TE LSP from R1 to R6. The parameters of the SR-TE LSP are covered in chapter 6 and are not repeated here. The only notable exception is that the **to** command contains an IPv6 destination address. The example also uses a **metric-type** of **te** to validate that R1 does compute the path using a CSPF based on IPv6 traffic engineering attributes.

Output 11-4: R1-to-R7 SR-TE LSP with IPv6

```
        mpls {
            path "empty" {
                admin-state enable
            }
            lsp "r1-to-r6-mpls-ipv6" {
                admin-state enable
                type p2p-sr-te
                to 2001:db8:33ac:2200::3:6
                path-computation-method local-cspf
                metric-type te
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                primary "empty" {
                    admin-state enable
                }
            }
        }
```

The SR-TE LSP is operationally up and although the actual hops are shown as IPv6 interface addresses not highlighted in *Figure 11-1*, the system address for each hop are also shown in brackets. Consequently, it can be seen that the LSP routes via R1-R4-R5-R6-R3 which aligns with a path computed on TE metric.

Output 11-5: R!-to-R6 SR-TE LSP Path

```
A:admin@R1# show router mpls sr-te-lsp "r1-to-r6-mpls-ipv6" path detail | match "Actual
Hops" post-lines 10
Actual Hops      :
    2001:db8:33ac:2200::4:a
    (2001:db8:33ac:2200::3:4)(A-SID)
    Record Label        : 524275
```

```
   2001:db8:33ac:2200::4:16
-> (2001:db8:33ac:2200::3:5)(A-SID)
   Record Label        : 524280
   2001:db8:33ac:2200::4:1a
-> (2001:db8:33ac:2200::3:6)(A-SID)
   Record Label        : 524285
```

Although computing the path of an SR-TE LSP with a local CSPF is supported in IS-IS, it is not supported for OSPFv3. SR-OS does not currently support OSPFv3 TE link parameter advertisement and will not populate the traffic engineering database with them if received from other vendors. (This is true for both SR-TE and RSVP-TE.) Without that information it is not possible to compute a path with a local CSPF.

Services over an IPv6 MPLS Core

When deploying services using SR-MPLS with an IPv6 core there are some nuances to be aware of. Deploying services using SR-MPLS in general is discussed in detail in chapter 8 and therefore will not be covered again here. Rather this section will cover the notable changes and additions needed when using an IPv6 core.

SDP-Based Services

For Layer-2 for services that use the Service Distribution Point (SDP) construct, the adoption of an IPv6 control plane is a fairly simple change. As one would anticipate, the **far-end ip-address** command in the SDP context uses an IPv6 address as opposed to an IPv4 address. The option then exists to use **sr-isis**, **sr-ospf**, or explicitly reference an LSP if an IPv6 SR-TE LSP should be used. Following this, services reference the SDP in the conventional manner with the **spoke-sdp** or **mesh-sdp** command followed by **<sdp>:<vc-id>**.

Output 11-6: SDP Construct with an IPv6 Control Plane

```
    service {
        sdp 2003 {
            admin-state enable
            delivery-type mpls
            sr-isis true
            far-end {
                ip-address 2001:db8:33ac:2200::3:3
            }
        }
    }
```

Advertising IPv4 and VPN-IPv4 NLRI

When using BGP and services with an IPv6 control plane in the core, there are some additional considerations with regard to the setting of next-hop self when a BGP speaker advertises IPv4 NLRI or VPN-IPv4 NLRI with Multiprotocol BGP.

The transport of IPv6 prefixes using Multiprotocol BGP over an IPv4 SR-MPLS backbone is already commonplace in provider networks. When advertising VPN-IPv6 routes to BGP speakers over an IPv4 network, or when interconnecting IPv6 islands over an IPv4 MPLS network using the 6PE approach, the next-hop is encoded as an IPv4-mapped IPv6 address containing the IPv4 address of the advertising BGP speaker. In both instances the IPv4 address can thereafter be extracted by an upstream router wishing to send traffic to the advertised prefix, and an appropriate IPv4 LSP to that next-hop address can be used. This method of advertising IPv6 prefixes over Multiprotocol BGP aligns with the base Address Family Indicator (AFI)/Subsequent Address Family Indicator (SAFI) definitions that specify that NLRI of a given address family should be advertised with a next-hop of the same address family.

Where Multiprotocol BGP is used to advertise reachability of IPv4 prefixes over an IPv6 core, it translates into the requirement for BGP to advertise NLRI of a given address family with a next-hop of a different address family (IPv4 with an IPv6 next-hop). However, this requirement does not align with the AFI/SAFI definitions specifying that the advertisement of IPv4 NLRI or VPN-IPv4 NLRI contain a next-hop address belonging to the IPv4 address family type. Extensions were therefore introduced in order to permit this to happen [38]. The extensions simply allowed for the advertisement of the IPv4 NLRI and VPN-IPv4 NLRI to contain either an IPv4 address or an IPv6 address. However, to do so meant that BGP speakers would need to know whether a peer is capable of supporting this new next-hop encoding. Hence a new capability was introduced known as the Extended Next-Hop Encoding capability that allowed a peer to indicate which set of AFI/SAFI could be advertised with a given next-hop AFI. Before a BGP speaker could advertise an IPv4 NLRI or VPN-IPv4 NLRI to a peer with an IPv6 next-hop, it must receive the relevant capability value from that peer.

To advertise and receive IPv4 routes with IPv6 next-hop addresses two additional commands are required in SR-OS which can be enabled at BGP, group, or neighbour level. The **advertise-ipv6-next-hops** command instructs the system to advertise IPv4 NLRI with an IPv6 next-hop address and is followed by one or more address families. To receive IPv4 NLRI with an IPv6 next-hop the **extended-nh-encoding** command must be enabled, again followed by one or more address families.

Output 11-7: Extended Next-Hop Encoding

```
        bgp {
            extended-nh-encoding {
                vpn-ipv4 true
                ipv4 true
            }
            advertise-ipv6-next-hops {
                evpn true
                vpn-ipv4 true
                ipv4 true
            }
        }
```

The above commands are enabled on all routers in the test topology of *Figure 11-1*, noting that this includes the Route-Reflector (the Route-Reflector must advertise this capability before IBGP clients can send IPv4 prefixes to it with an IPv6 next-hop). Enabling the **extended-nh-encoding** command will cause the router to advertise this capability to the peers to which it is applied, and since the capability negotiation is part of the Open exchange will cause the BGP session to cycle through a Notification and Open exchange.

Output 11-8: Extended Next-Hop Encoding Capability

```
A:admin@R1# show router bgp neighbor "2001:db8:33ac:2200::3:a" | match Capability post-
lines 1
Local Capability      : RtRefresh MPBGP 4byte ASN
                      : EXT_NH_ENCODING
Remote Capability     : RtRefresh MPBGP 4byte ASN
                      : EXT_NH_ENCODING
```

A VPRN service is extended between R1 and R3 and configured to use **auto-bind-tunnel** with a **resolution-filter** set to **sr-isis** only. CE3 advertises IPv4 prefix 172.31.3.0/24 to R3, which in turn advertises that prefix as a VPN-IPv4 route towards RR1. *Output 11-9* shows the prefix as received at R1, and it can be seen that the next-hop for this prefix is the IPv6 address 2001:db8:33ac:2200::3:3 (R3).

Output 11-9: VPN-IPv4 NLRI with an IPv6 Next-Hop

```
A:admin@R1#  show  router  bgp  routes  172.31.3.0/24  vpn-ipv4  detail  |  match  "Original
Attributes" post-lines 26
Original Attributes

Network         : 172.31.3.0/24
Nexthop         : 2001:db8:33ac:2200::3:3
Route Dist.     : 64496:1                VPN Label      : 524278
Path Id         : None
From            : 2001:db8:33ac:2200::3:a
Res. Nexthop    : n/a
Local Pref.     : 100                    Interface Name : link-to-R2
Aggregator AS   : None                   Aggregator     : None
Atomic Aggr.    : Not Atomic             MED            : None
AIGP Metric     : None                   IGP Cost       : 200
Connector       : None
Community       : target:64496:1
Cluster         : 192.0.2.10
Originator Id   : 192.0.2.3              Peer Router Id : 192.0.2.10
Fwd Class       : None                   Priority       : None
Flags           : Used Valid Best IGP Group-Best
Route Source    : Internal
AS-Path         : 64512
Route Tag       : 0
Neighbor-AS     : 64512
Orig Validation: N/A
Source Class    : 0                      Dest Class     : 0
Add Paths Send : Default
Last Modified   : 03d00h39m
VPRN Imported   :  1
```

As the **resolution-filter** filters tunnel table entries only to **sr-isis** for next-hop resolution, the IPv6 next-hop resolves to the shortest path IPv6 SR LSP advertised by R3.

Output 11-10: FIB Entry for VPN-IPv4 Prefix

```
*A:R1# show router 1 fib 1 172.31.3.0/24

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
172.31.3.0/24                                               BGP_VPN
  2001:db8:33ac:2200::3:3 (VPRN Label:524278 Transport:SR-ISIS:524310)
```

```
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
===============================================================================
```

Ethernet VPN

By default, EVPN routes use the IPv4 system address as the next-hop for a given EVPN MPLS service. EVPN MPLS services can however be deployed over an IPv6 core with IPv6 addresses used as next-hops of advertised EVPN routes, whereafter EVPN routes received with IPv6 next-hops are resolved to tunnels in the IPv6 tunnel table. This is enabled at service level with the **route-next-hop** command set to **system-ipv6** within the **bgp-evpn mpls** context. The configured IPv6 address is used as a next-hop for the advertised MAC Advertisement routes, Inclusive Multicast Ethernet Tag routes, and (per-EVI) Auto-Discovery routes.

To demonstrate its use an EVPN MPLS service is extended between R1, R3, and R4 with the **route-next-hop** command set to **system-ipv6** and an **auto-bind-tunnel resolution-filter** set to only **sr-isis**. *Output 11-11* shows the pertinent parts of the configuration applied for that service.

Output 11-11: IPv6 Tunnel Resolution for EVPN

```
    service {
        vpls "three" {
            bgp-evpn {
                mpls 1 {
                    auto-bind-tunnel {
                        resolution filter
                        resolution-filter {
                            sr-isis true
                        }
                    }
                    route-next-hop {
                        system-ipv6
                    }
                }
            }
        }
    }
```

Output 11-12 shows an EVPN MAC route advertised by R1 as received at R3 for the EVPN service with the next-hop attribute set to the IPv6 address 2001:db8:33ac:2200::3:1 (R1).

Output 11-12: EVPN MAC Advertisement from R1

```
*A:R3# show router bgp routes evpn mac mac-address 52:54:00:b0:00:01 detail | match
expression "Nexthop|Res. Nexthop|Mac Address|MPLS"
Nexthop          : 2001:db8:33ac:2200::3:1
Res. Nexthop     : fe80::5054:ff:fef3:4b76
Community        : target:64496:3 bgp-tunnel-encap:MPLS
Mac Address      : 52:54:00:b0:00:01
MPLS Label1      : LABEL 524269           MPLS Label2     : n/a
```

The MAC address is installed in the Forwarding Database (FDB) of the EVPN service at R3. The source identifier for the MAC is again R1's IPv6 system address, and the transport is shown as SR-ISIS (`isis`) with a tunnel table ID corresponding to the shortest path SR-ISIS LSP to R1.

Output 11-13: FDB Table Entry for Advertised MAC Route

```
A:admin@R3# show service id 3 fdb mac 52:54:00:b0:00:01

===============================================================================
Forwarding Database, Service 3
===============================================================================
ServId     MAC               Source-Identifier       Type     Last Change
            Transport:Tnl-Id                         Age
-------------------------------------------------------------------------------
3          52:54:00:b0:00:01 mpls-1:                 Evpn     11/22/21 17:04:46
                             2001:db8:33ac:2200::3:1:524269
           isis:524307
-------------------------------------------------------------------------------
Legend:  L=Learned O=Oam P=Protected-MAC C=Conditional S=Static Lf=Leaf
```

12 SRv6

Whilst the majority of this document has looked at SR-MPLS, this chapter focuses solely on SRv6. The chapter starts with an overview of SRv6 and how services are overlaid onto an SRv6 domain, and then goes on to cover the SR-OS implementation and configuration requirements.

Overview

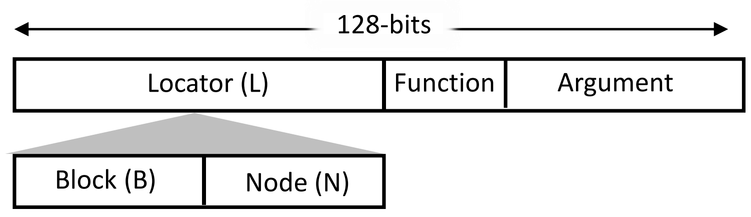
SR can be applied to the IPv6 data plane with a new type of routing header called the SR Header (SRH). An SRv6 segment, or SRv6 SID, is encoded as an IPv6 address and a segment list can be formed of a single segment or an ordered list of segments.

In shortest path routing the source node originates an IPv6 packet with a single segment encoded in the destination address of the IPv6 header and there no explicit requirement to include an SRH. A source-routed path contains an ordered list of segments encoded as IPv6 addresses in the SRH. The active segment is indicated by the destination address of the packet. The next active segment is indicated by a pointer in the SRH known as the *Segments Left* (SL) field. When an SRv6 segment or SID is completed, the segment endpoint copies the next segment into the destination address and decrements the SL by 1. Only routers whose address matches the address in the destination address of the IPv6 header must examine the SRH. Any transit nodes where the destination address is not any locally configured segment or local interface do not process a segment or SRH, but simply do a longest prefix lookup and forward the packet towards the destination.

SID Format

The SID format and processing behaviour for SRv6 SIDs is defined in [55]. The 128-bit SRv6 SID consists of a Locator, a Function, and an Argument. The Locator is encoded in the most significant bits of the SID, and the length of it is user-defined. The Function and Argument lengths may also be variable as long as the total of (Locator + Function + Argument) is <=128-bits. The Locator may be decomposed into a Block and a Node, where the Block is the IPv6 block/prefix range allocated to SRv6 SIDs by the operator, and the Node is the identifier of the parent node instantiating the SID. When the Locator part of the SRv6 SID is routable, it leads to the node that instantiated the SID. It is synonymous to a summary IPv6 address for a set of SIDs instantiated on an SRv6-capable router.

Figure 12-1: SRv6 SID Encoding



For example, assuming a /64 Locator part of the SID, an operator might allocate /48 prefix 2001:db8:aaaa/48 to all routers in an SRv6 routing domain, and this forms the common block B part of the Locator. Each router in the domain is then allocated a /16 node identifier allocated, and this is the node block N. Examples of Locator prefixes in the SRv6 domain would thereafter be R1=2001:db8:aaaa:1::/64, R2=2001:db8:aaaa:2::/64, and R3=2001:db8:aaaa:3::/64.

The Function is an opaque identification of a local behaviour bound to the SID. *Table 12-1* lists some well-known behaviours that can be associated with an SRv6 SID, but the list is not exhaustive. When an SRv6-capable node receives an IPv6 packet whose destination address matches a FIB entry representing a locally instantiated SRv6 SID, the function determines the endpoint behaviour and how the SRH and upper-layer header are processed. In SR-OS the function field has a configurable length range of 16-bits or between 20-96 bits with the default value being 20-bits and is used to assign End and End.X SIDs corresponding to Node-SIDs and Adj-SIDs respectively. The End function is statically configured, while the End.X function can be statically configured or dynamically assigned by the system.

Table 12-1: SRv6 Endpoint Behaviours

Behaviour	**Description**
End	Endpoint. The SRv6 instantiation of a Prefix-SID
End.X	L3 cross-connect. SRv6 equivalent of an Adjacency-SID
End.DX4	Decapsulation and IPv4 cross-connect (IPv4-L3VPN)
End.DX6	Decapsulation and IPv4 cross-connect (IPv6-L3VPN)
End.DT4	Decapsulation and specific IPv4 table lookup (IPv4-L3VPN)
End.DT6	Decapsulation and specific IPv6 table lookup (IPv4-L3VPN)
End.DT2U	Decapsulation and unicast MAC L2 table lookup (EVPN bridging unicast)
End.DX2	Decapsulation and L2 cross-connect (L2VPN)
End.DT2M	Decapsulation and L2 table flooding (EVPN BUM with ESI filtering)
H.Encaps	SR Headend with Encapsulation in an SR Policy

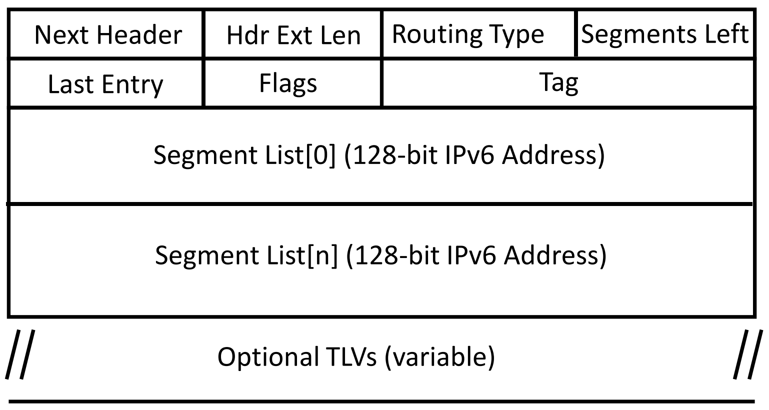
The Argument encodes any additional information that an SRv6 endpoint may need to process the packet and is currently set to all zeros in SR-OS with a non-configurable length that is determined by the length of the Locator and Function fields. For example, if the length of the Locator field is 64-bits and the length of the Function field is 20 bits, then the Argument field is 44-bits in length (64+20+44=128) which are all set to zero. If necessary, the Locator and Function fields may consume all 128-bits of the SID in which case the Argument field has a length of 0-bits.

Segment Routing Header (SRH)

The IPv6 SRH [54] is derived from the IPv6 Routing Header and its format is shown in *Figure 12-2*. The Next Header identifies the type of header immediately following the SRH, while the Hdr Ext Len specifies the length of the SRH in 8-octet units excluding the first 8 octets. The Routing Type identifies the type of Routing Header and the allocated value for SRH is

Routing Type 4. The Segments Left field contains the number of explicitly listed segments that still need to be visited before reaching the final destination. The Last Entry contains an index in the Segment List to identify the last element of the Segment List. There are currently no Flags specified, and the Tag is used to tag a packet as part of a class or group of packets. The Segment List [0..n] contains a list of 128-bit IPv6 addresses representing each segment in the Segment List. The Segment List is encoded starting with the last segment of the path. That is, the first element of the Segment List (Segment List[0]) contains the last segment of the path, the second element contains the penultimate segment of the path, and so on. There are some optional TLVs such as the Padding TLV used to meet the alignment requirement for subsequent TLVs, and the Hashed Message Authentication Code (HMAC) TLV used to provide authenticity to the SRH, but these are not discussed here.

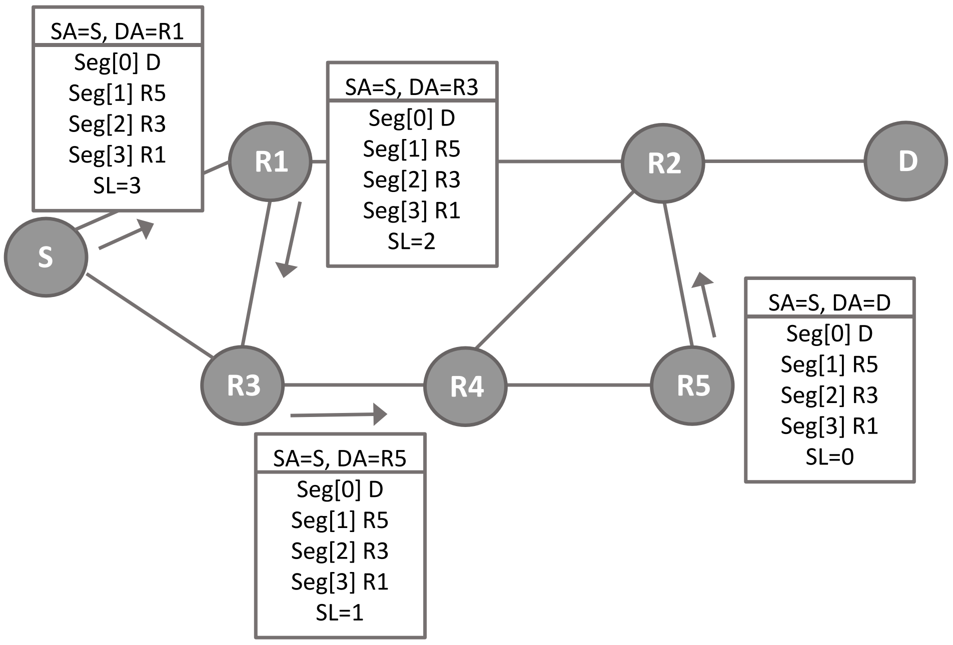
Figure 12-2: SRH Format



A source node is responsible for steering a packet into the SRv6 path. If the SRv6 path has a Segment List containing a single segment, and there is no requirement to add additional TLVs, the destination address is set to the single Segment List entry and the SRH can be omitted. If the Segment List contains multiple Segments, the SRH is required. The first segment is encoded in the IPv6 destination address. The first element of the SRH Segment List is the ultimate segment, the second element is the penultimate segment and so forth until all the segments of the path are specified in the Segment List. The Segments Left field and the Last Entry field are set to n-1, where n is the highest segment number defined in the Segment List. 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 path since it is already encoded in the destination address of the IPv6 header.

Figure 12-3 provides an example of packet forwarding for an SRv6 encapsulated packet from source S to destination D that needs to route along the path S-R1-R3-R5-D. At source S the IPv6 destination address for the packet is R1, and the segment list contains four segments encoded with destination D as the top segment [0] in the list with the SL field set to 3. Source S forwards the packet to R1, and when it arrives R1 interrogates the SRH, decrements the SL field by 1 leaving the SL with a value of 2. The SID contained in segment 2 (R3) is thereafter copied into the IPv6 destination address and the packet is forwarded towards R3. When the packet arrives at R3, it once again interrogates the SRH, decrements the SL field by 1 leaving the SL with a value of 1. The SID contained in segment 1 (R5) is therefore copied into the IPv6 destination address and the packet is forwarded towards R5 via R4. As R4 is not the destination address of the IPv6 packet, it is a transit node and does not interrogate the SRH but simply forwards the packet towards R5. When the packet arrives at R5, the same behaviour carried out at R1 and R3 is actioned, with SL now being 0 and the destination address set to D. When the packet arrives at the destination D, the SRH is interrogated and since the SL is set to 0, D removes the SRH outer IPv6 header and sends the packet for further processing.

Figure 12-3: SRv6 Packet Forwarding with SRH



Penultimate and Ultimate Segment Popping and Decapsulation

In the example of *Figure 12-3* the destination node D removed the SRH, however, it is possible for the penultimate hop to remove the SRH. Indeed, three variants of the End, End.X, and End.T behaviours are defined for removal of the SRH; the Penultimate Segment Pop (PSP), Ultimate Segment Pop (USP), and Ultimate Segment Decapsulation (USD). When using PSP, the penultimate SR segment endpoint (R5 in the above example) copies the last SID from the SRH into the IPv6 destination address and removes the SRH from the IPv6 extension header chain, updating the Next Header field in the preceding header to the Next Header value from the SRH.

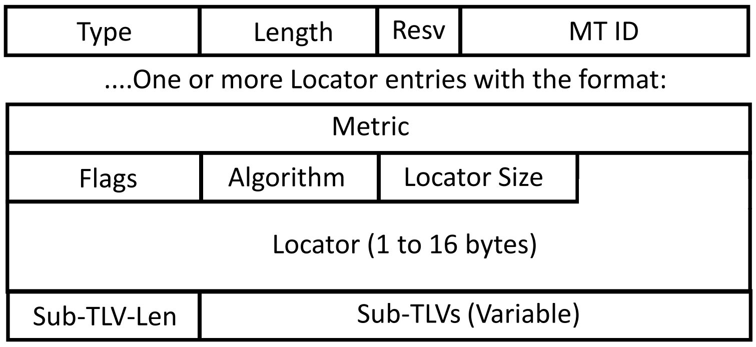
The USP variant involves removal of the SRH at the ultimate segment (where SL is 0) by some other entity than the host actually processing the IPv6 packet. This might be applicable to a server removing the SRH before passing the IPv6 packet to a virtual machine or container, or an application on a host with a smartNICs where the SRH is removed by the smartNIC before sending the packet to the host.

The USD variant involves not only removal of the SRH, but also decapsulation of the entire outer IPv6 header and extension headers by the ultimate segment endpoint before processing the inner IPv6 packet (which in itself may be an SRv6 packet). One application for this would be when a Point of Local Repair (PLR) programs a TI-LFA repair path for an IPv6 destination. When that PLR forwards IPv6 or SRv6 packets into the IPv6 repair path it requires an SRH to encode the segment or segments towards the tunnel endpoint. There are two potential ways to do this. Firstly, the PLR could insert an(other) SRH into the existing IPv6 packet, or secondly the PLR could completely encapsulate the existing IPv6 packet into an outer IPv6 header and SRH. SRH insertion is the most efficient way to do this from the perspective of additional overhead, but the insertion of an SRH at any node other than the originating node or destination node is yet to be fully standardised [56]. Some implementations may therefore fully encapsulate IPv6 packets in an outer SRv6 header at a PLR to fulfil a TI-LFA repair path, in which case the outer IPv6 header and any extension headers need to be completely decapsulated by the repair tunnel endpoint such that it can process the inner IPv6 packet. This is the function of USD.

Advertising Locators and Endpoints

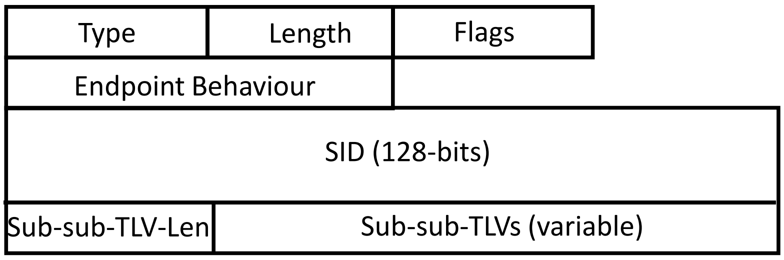
To make the Locator reachable to other nodes in the domain it is advertised in IS-IS and OSPFv3 using an SRv6 Locator TLV. SR-OS does not currently support SRv6 for OSPFv3, hence the focus in this section will be IS-IS. In IS-IS the SRv6 Locator TLV is a top level TLV and shares the sub-TLV space defined for Extended IP Reachability and Multi-topology IP Reachability TLVs. The IS-IS Locator TLV is used not only to advertise SRv6 Locators but also any End SIDs associated with each Locator. It takes the format shown in *Figure 12-4* where the Locator TLV has a base header and allows for insertion of one or more Locator entries within that header. The Locator entry encodes the Locator entry and size and allows the Locator to be associated with an algorithm, which is key to enabling the use of Flex-Algorithm with SRv6. Locators associated with algorithm 0 and 1 should be advertised in a Prefix Reachability TLV so that legacy SRv6-incapable routers can install a forwarding entry for algorithm 0 and 1 SRv6 traffic. The Flags field currently specifies a single flag, the D flag, which is the Up/Down bit. It is set to 1 as a loop prevention mechanism when the Locator is advertised from IS-IS Level 2 to IS-IS Level 1 to avoid it being re-advertised from Level 1 back into Level 2. The Locator TLV then allows for End SIDs associated with each Locator to be advertised as Sub-TLVs.

Figure 12-4: SRv6 Locator TLV



The SRv6 End SID sub-TLV is used to advertise SRv6 SIDs with endpoint behaviours that do not have an association with a particular neighbour or adjacency in order to be correctly programmed. The End endpoint behaviour is the most basic behaviour as it is the SRv6 instantiation of a Prefix-SID. That is, a packet received at router R with an IPv6 destination address as a local End SID of R will process the SRH and Next-Header. The endpoint behaviours encoded in the Endpoint Behaviour field of the sub-TLV are IANA allocated (iana.org/assignments/segment-routing) and SR-OS currently supports the End with PSP (Type 2) and End with USP (Type 3) variants. There are currently no flags defined for the SRv6 End SID sub-TLV.

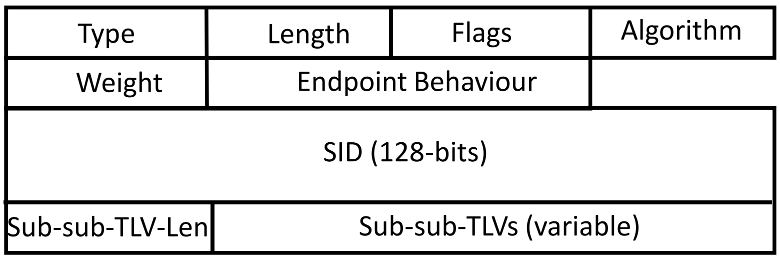
Figure 12-5: SRv6 End SID Sub-TLV



The advertisement of SRv6 Adjacency SIDs is clearly something that is associated with a particular neighbour or adjacency, and to that end two new two new sub-TLVs of the Extended IS Reachability TLV are used. The SRv6 LAN End.X SID sub-TLV is used to advertise SRv6 SIDs associated with a LAN, while the SRv6 End.X sub-TLV is used to advertise one or more SRv6 SIDs associated with a point-to-point adjacency. As the format of the SRv6 LAN End.X SID sub-TLV is largely the same with the exception of an additional Neighbor system-ID field, only the SRv6 End.X sub-TLV is described here, and its format is shown in *Figure 12-6*.

The Flags field defines three flags. The B-Flag is set if the SID is eligible for protection. When configuring End.X SIDs dynamically or statically in SR-OS a configuration option exists to advertise the SID as protected or unprotected. The former sets the B-flag while the latter leaves the flag unset. The S-Flag is used to indicate that the SID is part of an Adjacency Set, and the P-flag is set if the SID is persistently allocated. In SR-OS SRv6 SIDs can be allocated dynamically in which case the P-flag is set to zero, or they can be created as a static SID in which case the P-flag is set to one. The algorithm field is used to indicate which algorithm should be used to compute a path towards the advertised SID. All SIDs advertised in End.X SIDs must be a subnet of a Locator with a matching algorithm advertised by the same node in an SRv6 Locator TLV. If they do not meet these requirements the End.X SID sub-TLV must be ignored. This is enforced to ensure that the node advertising the SID is also advertising a corresponding Locator with the algorithm that will be used for computing a path towards that SID. The weight field is used for load-balancing in the presence of multiple parallel adjacencies. Weighted load-balancing in this manner is not currently supported by SR-OS. The endpoint behaviours encoded in the Endpoint Behaviour field of the sub-TLV are IANA allocated, and SR-OS currently supports the End.X with PSP (Type 6) and End.X with USP (Type 7) variants.

Figure 12-6: SRv6 End.X SID Sub-TLV



Both the End SID sub-TLV and the End.X SID described above can carry an optional SRv6 SID Structure sub-sub-TLV containing the Locator Block, Locator Node, Function, and Argument length. This optional sub-sub-TLV is intended for informational use. For example, a controller may learn the advertised SIDs together with their structure to ensure that the selected SRv6 SID format/structure is being adhered to. It is not currently signalled in SR-OS.

Service Overlay

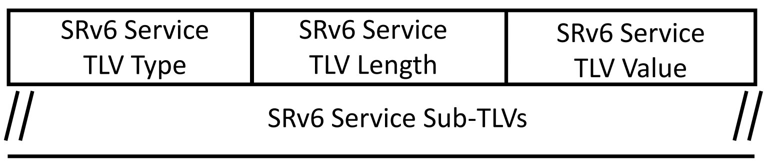
When using an MPLS (or SR-MPLS) core, BGP-based services such as global IPv4/IPv6, IPv4/IPv6-VPN, and EVPN advertise their relevant reachability and associate an MPLS label with those NLRI. This enables Layer 2 and Layer 3 overlay services with BGP as a control plane and MPLS as a data plane. When using an SRv6 domain, BGP-based services are extended to advertise SRv6 Service SIDs [57] in order to provide Layer 2 and Layer 3 overlay services with a BGP control plane and an SRv6 data plane. SRv6 Service SIDs refer to an SRv6 SID associated with one of the service-specific endpoint behaviours on the advertising PE router, such as End.DT4 (IPv4-L3VPN) or End.DT2U (EVPN Bridging Unicast).

A BGP Prefix Segment is a BGP prefix with a Prefix-SID [58] attached and is carried in Multi-Protocol BGP. An optional transitive BGP attribute known as the BGP Prefix-SID attribute can be associated with a BGP Prefix-SID, and it is this attribute that is extended to carry SRv6 Service TLVs. There are two new TLVs of the BGP Prefix-SID attribute used to signal SRv6 SIDs for Layer 2 and Layer 3 overlay services:

- The SRv6 L3 Service TLV. This TLV carries Service SID information for SRv6-based Layer 3 services and corresponds to the equivalent of an MPLS label in carried in VPN-IPv4/VPN-IPv6 service routes. Example endpoint behaviours include but are not limited to End.DX4, End.DT4, End.DX6, and End.DT6.
- The SRv6 L2 Service TLV. This TLV carries Service SID information for SRv6-based Layer 2 services and corresponds to the equivalent of MPLS Label1 for EVPN service routes. Example endpoint behaviours include but are not limited to End.DX2, End.DX2V, and End.DT2U.

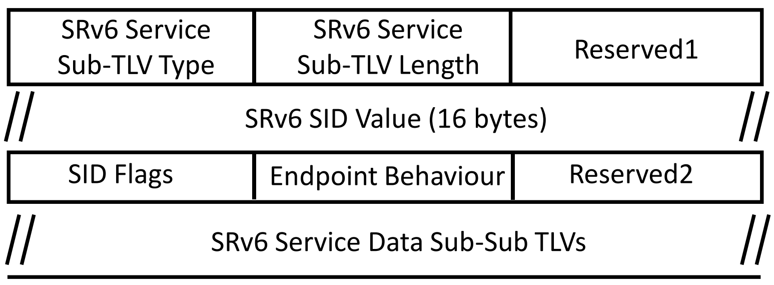
SRv6 service related information is structured in a hierarchical way consisting of SRv6 Service TLVs, which carry SRv6 Service sub-TLVs, which in turn carry SRv6 Service Data sub-sub-TLVs. Starting from the top of the hierarchy, the structure of the SRv6 Service TLV is shown in *Figure 12-7* and is encoded in the BGP Prefix-SID attribute. It is a simple TLV format that acts as a holding or umbrella structure, with the option to carry SRv6 Service sub-TLVs with variable length.

Figure 12-7: SRv6 Service TLV



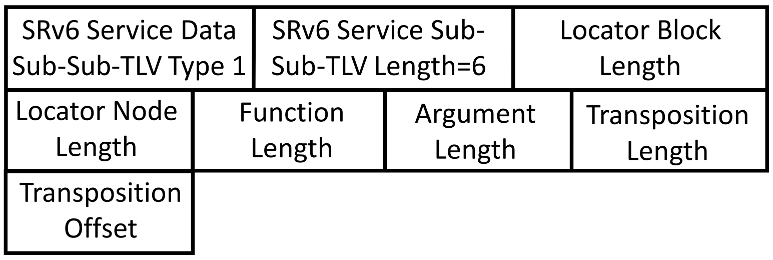
The format of the SRv6 Service sub-TLVs carried in the SRv6 Service TLV is again a simple TLV structure, and SRv6 Service Sub-TLV Type 1 is assigned to the SRv6 SID Information sub-TLV. Its encoding is shown in *Figure 12-8*, and it is used to signal the SRv6 SID value assigned to the NLRI together with the endpoint behaviour associated with that SID value. It also has a variable field for encoding the next level of SRv6 service information, the SRv6 Service Data sub-sub-TLVs

Figure 12-8: SRv6 SID Information Sub-TLV


SRv6 Service Sub-TLV Type
SRv6 Service Sub-TLV Length
Reserved1
SRv6 SID Value (16 bytes)
SID Flags
Endpoint Behaviour
Reserved2
SRv6 Service Data Sub-Sub TLVs

The SRv6 Service sub-TLV above carries SRv6 Service Data sub-sub-TLVs, and sub-sub-TLV Type 1 is assigned to the SRv6 SID Structure sub-sub TLV. Its encoding is shown in *Figure 12-9* and as the name suggests it is used to provide information about the SID encoding. The Locator Block Length, Locator Node Length, Function Length, and Argument Length fields together provide information as to exactly how the 128-bit SID is structured. The Transposition Length and Transposition Offset fields require some explanation. Some types of BGP services such as IP-VPN use a per-SID allocation (analogous to label-per-VRF) where a single SID is shared across multiple NLRIs, and this provides efficient packing of NLRI into BGP updates . Other types of BGP services like EVPN-VPWS use a per-pseudowire allocation where each NLRI has its own unique SID, resulting in inefficient NLRI packing. To allow for efficient NLRI packing, the SRv6 SID value contained in the SRv6 SID Information Sub-TLV may contain just the common part of the SRv6 SID (the Locator) while the variable parts of the SRv6 SID (Function and Argument) are encoded into the existing MPLS label fields. Because the rest of the SID is common for routes of the same type in the service, multiple routes can be packed into the same BGP Update. This type of encoding is referred to as *Transposition*, and the SRv6 SID Structure sub-sub-TLV describes the sizes of the Locator, Function, and Length, as well as indicating the offset of the variable part and its length.

Figure 12-9: SRv6 SID Structure Sub-Sub-TLV



For service routes, SR-OS transposes up to 20-bits of the SRv6 SID Function value into the label field of the NLRI, which is bounded by the size of the MPLS label field. As the function length is configurable as either 16-bits or between 20-96 bits, it follows that the Transposition Length is therefore either 16-bits or 20-bits and the Transposition Offset identifies the point in the SID where the transposed value starts. For example, if the Locator {Block + Node} length is 64-bits, and the Function length is 20-bits, then the transposition offset is 64-bits. If the Locator {Block + Node} length is 64-bits and the Function Length is 32-bits, only 20-bits of the Function are transposed as the right-most bits of the Label field, and the transposition offset is 76-bits. This offset represents the 64-bit Locator and 12-bits of the Function field that are not transposed.

To help clarify this transposition behaviour, *Output 12-1* shows a VPN-IPv4 service route that is part of a VPRN overlay service on an SRv6 core. The output is truncated to contain only the relevant parts, but the SRv6 Service TLV, SRv6 SID Information sub-TLV, and SRv6 SID Structure sub-sub-TLV are shown. The `Sid` field in the SRv6 SID Information shows the advertised Locator prefix 2001:db8:4497:5::, which according to the SRv6 SID Structure sub-sub-TLV is formed of a 48-bit Locator Block (`Loc-Block-Len`) and 16-bit Locator Node (`Loc-Node-Len`). The Function follows the Locator, and as shown in the `Func-Len` field it is 20-bits in length. This 20-bits is shown in the `Full Sid` field as being hex 7ffd:7. This is the value that is transposed into the NLRI MPLS label and is shown in the `VPN Label` as the decimal equivalent 524247. The resulting transposition offset (the point that marks the beginning of the transposed value) is shown in the `Tpose-offset` field as 64-bits. To be clear, the `Full Sid` field is not an advertised value because the Function part is transposed. It is provided in the CLI output purely for operator ease to show a concatenated view of the entire SID constructed of Locator, Function, and Argument.

Output 12-1: Service Route Transposition

```
*A:SRv6-DUT# show router bgp routes 10.148.5.0/24 vpn-ipv4 detail
Network         : 10.148.5.0/24
Nexthop         : 2001:db8:33ac:2200::3:5
Route Dist.     : 64496:800              VPN Label      : 524247
--- [snip] ---
Originator Id   : 192.0.2.5              Peer Router Id : 192.0.2.10
Fwd Class       : None                   Priority       : None
Flags           : Used Valid Best IGP Group-Best
Route Source    : Internal
--- [snip] ---
SRv6 TLV Type   : SRv6 L3 Service TLV (5)
SRv6 SubTLV     : SRv6 SID Information (1)
Sid             : 2001:db8:4497:5::
Full Sid        : 2001:db8:4497:5:7ffd:7000::
```

```
Behavior        : End.DT4 (19)
SRv6 SubSubTLV : SRv6 SID Structure (1)
Loc-Block-Len   : 48                          Loc-Node-Len    : 16
Func-Len        : 20                          Arg-Len         : 0
Tpose-Len       : 20                          Tpose-offset    : 64
```

Configuration

At the time of writing SRv6 is supported only on FP-based platforms with network interfaces on FP4 ports. It is not currently supported on 7250 IXR generation 1 platforms but is supported on IXR generation 2/2c platforms. Additionally, SR-OS does not support SRv6 for OSPFv3 with current support only for IS-IS in both Multi-Topology (MT) 0 (standard) and MT2 (IPv6).

Please check Release Notes for an updated list of 7250 IXR unsupported features.

This section looks at the configuration and validation aspects of enabling SRv6 in SR-OS. Before diving straight into the configuration, a few words on caveats. SRv6 is supported in IS-IS for shortest path tunnels only. There is no current support for encoding additional segments of a source-routed path into the SRH, and the rationale for this will be discussed later in this chapter in the section ‘The Future of SRv6’. There is however full support for Flex-Algorithm with SRv6 to enable a degree of traffic engineering, and the applicability of Flex-Algorithm to SRv6 will covered throughout this chapter. The current release makes very little use of the SRH and the only scenario that mandates the insertion of an SRH in the SRv6 header is when programming an LFA backup path that requires the imposition of a SID. In the case of a PQ intersect, this may be an End SID, and in the case where the P and Q nodes are adjacent by a single hop it will be an End.X SID. As the segment (IPv6 Adj-SID) is globally unique it only requires a single SID as opposed to the Node-SID and Adj-SID required with SR-MPLS. Finally, SRv6 is supported in SR-OS on FP-based platforms with network interfaces on FP4 ports.

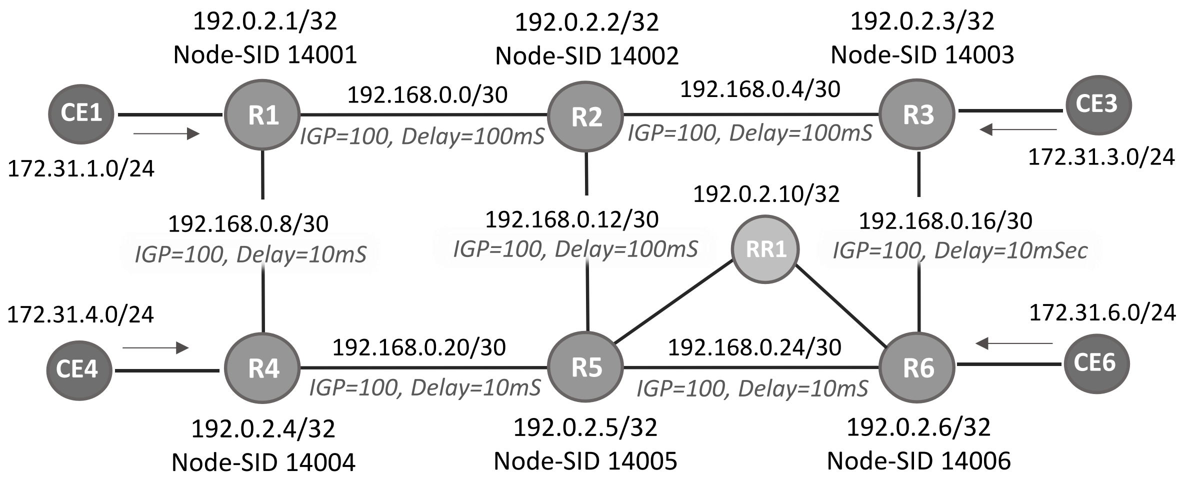
Test Topology

To illustrate the use of SRv6 the network topology shown in *Figure 12-10* is used. Routers R1 to R6 and a Route-Reflector, RR1, are in a flat IS-IS Level 2 area. All routers belong to Autonomous System 64496 and Routers R1 to R6 peer in IBGP with RR1 as Route-Reflector clients for advertisement of service routes. The network is dual-stack IPv4/IPv6 and all BGP peering uses IPv6 addressing. Note that IPv6 peering is not a requirement for SRv6 simply because SRv6 does not use BGP next-hop addresses to resolve BGP prefixes to SRv6 tunnels – that is the function of the Locator. All IGP link metrics are 100. Unidirectional link delay is also configured, and all links have a delay of 10 milliseconds with the exception of links R1-R2, R2-R3, and R2-R5, which have a delay of 100 milliseconds. Flex-Algorithm is enabled and algorithm identifier 128 is configured to use latency as a metric. Two /48 locator blocks are used for SRv6, both of which have node lengths of 16-bits to yield a total 64-bit locator length. Locator block 2001:db8:1::/48 is assigned to the base algorithm (algorithm 0) while

Locator block 2001:db8:128::/48 is assigned to Algorithm 128. Node numbers are 1 to 6 for routers R1-R6 respectively. For example, R1's advertised locators are 2001:db8:1:1::/64 and 2001:db8:128:1::/64, while R2's locators are 2001:db8:1:2::/64 and 2001:db8:128:2::/64.

A number of CE routers are attached to the topology that advertise IPv4 and IPv6 prefixes to their adjacent routers which will be used to verify that SRv6 is functioning as anticipated.

Figure 12-10: Topology for SRv6



Forwarding Path Extension (FPE)

As a general rule whenever IP is encapsulated in IP on a Nokia FP-based platform, there is a requirement to use a logical loopback function to allow for multiple passes through the forwarding plane. As SRv6 is essentially IP encapsulated in IP it is no exception and requires the use of a Forwarding Path Extension (FPE) at any router that originates or terminates SRv6-based services. It is not required on any SRv6 transit routers, including any router that may need to process an SRH. The FPE is an extension of the Port Cross-Connect (PXC) feature, which allows for one or more ports, or an E4 MAC chip, to be logically looped to itself allowing for two passes through the forwarding plane.

Note: Configuration of an FPE is only required on FP-based platforms that originate or terminate SRv6-based services. It is not required on 7250 IXR Gen 2/2c platforms supporting SRv6.

The FPE is dedicated to SRv6, and its function differs at the ingress PE and egress PE.

- At the ingress PE the FPE egress data path pushes on the SRv6 encapsulation header (and if necessary, an SRH). The FPE ingress data path does the IPv6 route lookup and forwards the packet towards the appropriate network interface.
- At the egress PE, a longest prefix lookup checks if the destination address matches a local locator, and if it does, the packet is forwarded to the FPE. The FPE egress data path processes the specific SID function and removes the SRv6 encapsulation headers, including the SRH if any are present. It then inserts a service label with the value derived from the Function field into the inner service packet. The FPE ingress

data path does an ILM lookup on the service label and forwards the packet to the relevant service context for further processing.

Output 12-2 shows the necessary configuration to implement a logical loopback port on an E4 MAC chip. The **xconnect** context under the relevant card and MDA allows for selection of the appropriate MAC chip if multiple exist, and in this example, **mac 1** is selected. Within the **mac** context, up to two loopback functions can be implemented with the **loopback** command. Creating a loopback port on an E4 MAC chip does not stop the MAC chip from being used for generic data path forwarding. Rather, the use of a loopback function on a MAC chip is intended to use spare capacity that is not used by faceplate ports. Indeed, if required, within the **loopback** context the user can choose to restrict throughput of the logical loopback function using an optional **bandwidth** command.

Output 12-2: E4 MAC Chip Logical Loopback

```
    card 1 {
        mda 1 {
            xconnect {
                mac 1 {
                    loopback 1 {
                    }
                }
            }
        }
    }
```

The next step is to configure the PXC which will then be assigned to the FPE. This is done in the **port-xc** context and can be numbered from 1 to 64. A port must be assigned to the PXC before it is put into an **admin-state** of **enable**, and for standardisation of configuration the MAC chip loopback also uses port nomenclature where **port-id 1/1/m1/1** represents the card/MDA/MAC chip/loopback identifier. SRv6 requires an FPE for service origination and another for service termination, and since the requirement is to do both, two PXCs are created which will be assigned to two different FPEs. These two PXC instances can use the same or different logical loopback functions, and in this example the same MAC loopback is used for both PXC instances.

Output 12-3: PXC Configuration

```
    port-xc {
        pxc 1 {
            admin-state enable
            port-id 1/1/m1/1
        }
        pxc 2 {
            admin-state enable
            port-id 1/1/m1/1
        }
    }
```

Once the PXC has been placed into an **admin-state** of **enable**, two PXC sub-ports are automatically created by the system and are identified by *.a* and *.b* suffixes of the parent PXC. Following system creation the PXC sub-port CLI configuration is automatically generated and can be accessed in the same way as a conventional physical port using the syntax “port pxc-*n.l*” where “*n*” represents the assigned PXC number and “*l*” represents the sub-port letter (a or b). The sub-ports are created in a disabled **admin-state**, so both the PXC sub-ports and the MAC loopback port need to be put into an **admin-state** of **enable**.

Output 12-4: Enabling the PXC Sub-Ports and MAC Loopback Port

```
    port pxc-1.a {
        admin-state enable
    }
    port pxc-1.b {
        admin-state enable
    }
    port pxc-2.a {
        admin-state enable
    }
    port pxc-2.b {
        admin-state enable
    }
    port 1/1/m1/1 {
        admin-state enable
    }
```

The final configuration step is the creation of the FPEs, which are created in the **fwd-path-ext** context as shown in *Output 12-5*. The FPEs are assigned a number from 1 to 64, and within each FPE the **path** context allows for the assignment of the associated PXC. In this example, **fpe 1** is assigned to **pxc 1,** and **fpe 2** is assigned to **pxc 2**, and in turn both PXCs are assigned to the same MAC loopback port. FPEs can be used for multiple applications, hence the **application** is explicitly configured to be **srv6** and the FPE cannot be used concurrently for any other application. Within the **srv6** sub-context, the **type** command is used to select whether the FPE is for **origination** or **termination**, and in this case FPE 1 is dedicated to origination while FPE 2 is dedicated to termination.

Output 12-5: FPE Configuration

```
    fwd-path-ext {
        fpe 1 {
            path {
                pxc 1
            }
            application {
                srv6 {
                    type origination
                }
            }
        }
        fpe 2 {
            path {
                pxc 2
            }
            application {
                srv6 {
                    type termination
                }
            }
        }
    }
```

The operational state of each FPE and the applications assigned to it are shown in *Output 12-6*. As the PXC sub-ports use conventional port nomenclature, their operational state can also be displayed if required using conventional 'show port' commands.

Output 12-6: Operational State of FPE

```
A:admin@R1# show fwd-path-ext fpe 1

===============================================================================
FPE Id: 1
===============================================================================
Description          : (Not Specified)
Multi-Path           : Disabled
Path                 : pxc 1
Pw Port              : Disabled                  Oper     : down
Sub Mgmt Extension   : Disabled                  Oper     : N/A
Vxlan Termination    : Disabled                  Oper     : down
Segment-Routing V6   : Enabled                   Oper     : up
SRv6 Type            : origination
If-A Qos Policy      : default
If-B MTU             : 9786 bytes                Oper MTU : 9190 bytes
If-B Qos Policy      : default
```

Locators and Function Endpoint Behaviour

Note: The default Function length for FP-based platforms is 20-bits. Broadcom Jericho platforms such as the 7250 IXR Gen2/2c are constrained to the use of a 16-bit Function length, hence this may be required for interop. When using 16-bit function lengths on FP-based platforms there are some notable differences in both configuration and router behaviour when compared to 20-bit or larger function lengths. As opposed to trying to identify and call out those differences within this section, the delta configuration requirements when using 16-bit function lengths with FP-based platforms are documented in Appendix B.

Two locators are configured on each of the routers R1 to R6 representing one Locator that will use the base routing algorithm (algorithm 0) and one Locator that will use Flex-Algorithm 128. One Locator is always required for algorithm 0. As shown in *Output 12-7* the **segment-routing-v6** context is used for configuration of these Locators and their associated attributes, together with the behaviour for End and End.X Functions in the global routing table or base routing instance. The **origination-fpe** command configures a single originating FPE for all SRv6-based services, and in this example FPE 1 is used. The **source-address** is used as the source address of data packets originated for SRv6-based services, and if necessary, this can be overridden at service level using the **segment-routing-v6 source-address** command. There are two **locator** instances configured. The second entry is configured to use **algorithm** 128, which is the delay-based algorithm. The first entry has no explicit algorithm configured and will therefore use the default algorithm 0. This algorithm identifier will be included in the SRv6 Locator TLV when advertising the Locator into IS-IS. The locator **block-length** is 48-bits, which together with a 16-bit node identifier yields a total of 64-bits Locator length. The **function-length** is also configurable within this context, although it is not shown in the output because the default 20-bit value is used. The **termination-fpe** specifies the FPE instance that will be used for SRv6 service termination. Unlike the **origination-fpe**, the **termination-fpe** is configurable on a per-locator basis, and both Locators are configured to use FPE 2. To reiterate for clarity, only routers that originate and terminate services need to specify an **origination-fpe** and **termination-fpe**, transit routers do not require the use of an FPE. In the test topology routers R2 and R5 will not be terminating services, therefore they do not require the configuration of an originating or terminating FPE. They do however require the configuration and advertisement of locators and SID functions, to allow them for example to function as TI-LFA repair path endpoints. The **prefix** sub-context allows for configuration of the SRv6 Locator prefix using the **ip-prefix** command and the example specifies a /64 prefix for each Locator. Finally, both locator instances are put into an **admin-state** of **enable**.

Although not shown in *Output 12-7*, a **static-function** context exists within each **locator** instance. Within the **static-function** context there is a **max-entries** command that defines the maximum number of values from the Function field that must be reserved for static End, End.X and service SID function assignment. It has a default value of 1, if static End, End.X, or service SIDs are used and more than one is required, this value must be increased accordingly.

The **base-routing-instance** context provides the context to configure the endpoint behaviour for End, End.X, and service SIDs for IPv4/IPv6 prefixes within the global routing table. The **locator** command is used to reference the Locator configured to use algorithm 0. Within the following **function** context, the **end** command is followed by an integer value in the range 1-1048575 that is used to statically define the End Function value. In this example the **end 1** command will mean the insertion of the value 1 into the 20-bit Function field. The **srh-mode** then defines the method of processing the SRH and can be either **usp** or **psp**, with the default being **psp**. End.X Function values can be allocated statically or dynamically within the **function** context. Dynamic allocation uses the **end-x-auto-allocate** command and allocates a Function value from the range [{max-entries+1}..{2^{20}-1}]. It is followed by the SRH mode which again can be **usp** or **psp**, and a **protection** command that specifies whether or not the link is eligible for TI-LFA protection (which in turn determines if the B-flag set or unset in the SRv6 End.X SID sub-TLV). If a **protection** type of **protected** is selected, TI-LFA must be enabled in order to generate End.X SIDs. Static End.X Functions are configured using the **end-x** command followed by an integer value in the range 1-1048575. If multiple **end-x** instances are required, which will very likely be the case as one will be required for every adjacency, the relevant Locator **static-function max-entries** value must be increased from the default value of 1. Each **end-x** context provides for definition of the **srh-mode**, the **protection** mode, and an **interface-name** to which the Function will be assigned. Statically configured End-X Functions are persistent in nature. Dynamically allocated entries are not persistent.

Output 12-7: Locator and Function Endpoint Behaviour

```
    router "Base" {
        segment-routing {
            segment-routing-v6 {
                origination-fpe [1]
                source-address 2001:db8:1:1::1
                locator "Alg0-Locator" {
                    admin-state enable
                    block-length 48
                    termination-fpe [2]
                    prefix {
                        ip-prefix 2001:db8:1:1::/64
                    }
                }
                locator "Alg128-Locator" {
                    admin-state enable
                    block-length 48
                    termination-fpe [2]
                    algorithm 128
                    prefix {
                        ip-prefix 2001:db8:128:1::/64
                    }
                }
                base-routing-instance {
                    locator "Alg0-Locator" {
                        function {
                            end 1 {
                                srh-mode usp
                            }
                            end-x-auto-allocate usp protection protected { }
                        }
                    }
                }
            }
        }
    }
```

Output 12-8 shows the local SIDs instantiated at R1 as a result of the above configuration. The End SID has a Function value of 1 which is appended after the Locator value to derive a SID value of 2001:db8:1:1:**0:1**000. The End.X SIDs have Function values of 2 and 3, which are derived from the formula [max-entries+1], and hence increment from the value 2.

Output 12-8: Router R1's Local-SIDs

```
A:admin@R1# show router segment-routing-v6 local-sid

===============================================================================
Segment Routing v6 Local SIDs
===============================================================================
SID                                               Type           Function
  Locator
  Context
-------------------------------------------------------------------------------
2001:db8:1:1:0:1000::                             End            1
  Alg0-Locator
  Base
2001:db8:1:1:0:2000::                             End.X          2
  Alg0-Locator
    None
2001:db8:1:1:0:3000::                             End.X          3
  Alg0-Locator
    None
-------------------------------------------------------------------------------
SIDs : 3
-------------------------------------------------------------------------------
```

Once the Locators and SID Functions are configured the next step is to advertise them into IS-IS. One or more Locators can be enabled in a given IS-IS instance or multiple instances, and each Locator can be enabled in a single topology of that instance, either MT0 (standard topology) or MT2 (IPv6). By default, Locators are added in the standard topology, MT0. Using export policies, remote Locators can also be redistributed between MT0 and MT2 topologies of different IS-IS instances.

Within the **segment-routing-v6** context of the relevant **isis** instance, each **locator** is configured and assigned a **level-capability** and an optional **metric** that will be advertised with the SRv6 Locator TLV. Each locator instance must also be put into an **enable admin-state**.

Output 12-9: Enabling IS-IS for SRv6

```
    router "Base" {
        isis 0 {
            segment-routing-v6 {
                admin-state enable
                locator "Alg0-Locator" {
                    level-capability 2
                    level 2 {
                        metric 1
                    }
                }
                locator "Alg128-Locator" {
                    level-capability 2
                    level 2 {
                        metric 1
                    }
                }
            }
        }
```

Note: IS-IS narrow metrics are 6-bits long and have a maximum value of 63. As the Locator TLV metric is 32-bits and has a maximum value of 16,777,215 they are not advertised in IS-IS unless **wide-metrics** are enabled.

Output 12-10 shows router R1's advertised IS-IS LSP with SRv6 enabled, truncated to show only the parts relevant to SRv6. The Router Capability TLV is extended to carry an SRv6 Capabilities sub-TLV, and the Node MSD Advertisement sub-TLV discussed in chapters 3 and 4 is extended to include SRv6 MSDs, which are described in *Table 12-2* below the output. The Extended IS Reachability TLV shows the End.X SID sub-TLV for the adjacency to R2 with its associated SID value and endpoint behaviour. The Locator TLV shows the two Locator prefixes of 2001:db8:1:1::/64 and 2001:db8:128:1::/64 for algorithm 0 and 128 respectively. Although not shown in the output, the prefix 2001:db8:1:1::/64 is also advertised in an IPv6 Prefix Reachability TLV so that SRv6-incapable routers can install a forwarding entry for algorithm 0 SRv6 traffic.

Output 12-10: R1's IS-IS LSP with SRv6 Enabled

```
A:admin@R1# show router isis database R1.00-00 level 2 detail

 --- [snip] ---
  Router Cap : 192.0.2.1, D:0, S:0
    Node MSD Cap: BMI : 12 ERLD : 15 SRH-MAX-SL : 10 SRH-MAX-END-POP : 9 SRH-MAX-H-ENCAPS
: 1 SRH-MAX-END-D : 9
    SRv6 Cap: 0x0000
 --- [snip] ---
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 173
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
 --- [snip] ---
    End.X-SID: 2001:db8:1:1:0:2000:: flags:B algo:0 weight:0 endpoint:End.X-USP
  SRv6 Locator  :
    MT ID : 0
    Metric: ( ) 1 Algo:0
    Prefix   : 2001:db8:1:1::/64
    Sub TLV   :
      AttrFlags: N
      End-SID   : 2001:db8:1:1:0:1000::, flags:0x0, endpoint:End-USP
    Metric: ( ) 1 Algo:128
    Prefix   : 2001:db8:128:1::/64
    Sub TLV   :
      AttrFlags: N
 --- [snip] ---
```

Table 12-2: Node MSD Extensions for SRv6

Behaviour	**Description**	**SR-OS Value**
Max Segments Left	The maximum value of the Segments Left field in the SRH of a received packet before applying the Endpoint behaviour associated with the SID.	10
Max End Pop	The maximum number of SIDs in an SRH to which the router can apply PSP or USP behaviour.	9
Max H.Encaps	Maximum number of SIDs in an SRH that can be pushed for SR Policy	1
Maximum End D	Maximum number of SIDs in an SRH when performing decapsulation for permitted SID types	9

All SRv6-enabled routers advertise a Locator prefix and each SR-OS router in the SRv6 domain installs resolved Locator prefixes from received SRv6 Locator TLVs in the route-table, FIB, and tunnel-table. *Output 12-11* shows R1's route-table entry for R3's algorithm 128 Locator prefix 2001:db8:128:3::/64. It is programmed as a direct tunnel next-hop over SRv6 IS-IS and has a metric of 40001 representing the shortest path latency path via R1-R4-R5-R6-R3 with the addition of the metric 1 that was advertised with the Locator TLV.

Output 12-11: R1's Route-Table Entry for R3's Algorithm 128 Locator Prefix

```
A:admin@R1# show router route-table ipv6 2001:db8:128:3::/64

===============================================================================
IPv6 Route Table (Router: Base)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
2001:db8:128:3::/64                           Remote  ISIS      02d22h47m  18
       2001:db8:128:3::/64 (tunneled:SRV6-ISIS)                     40001
```

An entry for the Locator prefix is also added as added to the tunnel-table with a protocol owner of `srv6-isis`. The next-hop address is the IPv6 link local address of the neighbour R4, and the `[L]` flag indicates that an LFA backup is programmed. The preference for the entry is 0 simply because there is no requirement to have a comparative preference metric

– it is the only tunnel-table protocol that can be used to forward SRv6 traffic. The tunnel ID shown in the output is 524319 and this can be reconciled from the FIB output shown in *Output 12-13*.

Output 12-12: R1's Tunnel-Table Entry for R3's Algorithm 128 Locator Prefix

```
A:admin@R1# show router tunnel-table ipv6 2001:db8:128:3::/64

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                  Owner     Encap TunnelId  Pref
Nexthop                                      Color           Metric
-------------------------------------------------------------------------------
2001:db8:128:3::/64 [L]                      srv6-isis SRV6  524319    0
  fe80::5054:ff:fec0:8c57-"link-to-R4"                         40001
-------------------------------------------------------------------------------
```

Output 12-13: R1's FIB Entry for R3's Algorithm 128 SRv6 Locator

```
A:admin@R1# show router fib 1 ipv6 ip-prefix-prefix-length 2001:db8:128:3::/64

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                            Protocol
  NextHop
-------------------------------------------------------------------------------
2001:db8:128:3::/64                                       ISIS
  2001:db8:128:3::/64 (Transport:SRV6:524319)
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Before concluding on the advertisement of Locator prefixes it is worth mentioning that when Locator prefixes are advertised between levels, they can be summarised using the existing **summary-address** command within IS-IS. In other words, the existing **summary-address** command is locator prefix and algorithm aware. This can be useful in large networks where CIDR is essential to reduce the size of FIB tables.

LFA Support

LFA support is provided for SRv6 Locator prefixes for the base routing instance and any Flex-Algorithm instances and is enabled for the former as described in chapter 9 and the latter as described in chapter 10.

When SR-MPLS is enabled and the **isis segment-routing** context is in an **admin-state** of **enable**, the system will use the value configured for **max-sr-frr-labels** (default value of two) for the SRv6 LFA. If SR-MPLS is disabled, the system will set **max-sr-frr-labels** to a value of one and use this. Using this derived value, the system does a single set of base LFA, RLFA, and TI-LFA computations for both SR-MPLS and SRv6 when both are enabled. If SR-MPLS computes a TI-LFA repair path consisting of two labels SR-OS will try to compress the SID to one End.X SID for SRv6. If the repair path cannot be compressed to a single SID, the SRv6 backup path is not programmed. In other words, SR-OS will only ever use a single SID in an SRv6 LFA repair path.

Output 12-14 shows R1's alternative route-table entry for R3's algorithm 128 Locator prefix 2001:db8:128:3::/64. The primary next-hop is the link local address of R4, while the backup next-hop is indicated with the `(LFA)` flag and is the link local address of R2.

Output 12-14: Alternative Route-Table Entry for R3's Algorithm 128 Locator Prefix

```
A:admin@R1# show router tunnel-table ipv6 2001:db8:128:3::/64 alternative

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                   Owner     Encap TunnelId  Pref
Nexthop                                       Color           Metric
-------------------------------------------------------------------------------
2001:db8:128:3::/64                           srv6-isis SRV6  524319    0
  fe80::5054:ff:fec0:8c57-"link-to-R4"                          40001
2001:db8:128:3::/64 (LFA)                     srv6-isis SRV6  524319    0
  fe80::265:ffff:fe00:0-"link-to-R2"                             -
-------------------------------------------------------------------------------
```

The programmed backup path can also be seen in the FP tunnel-table shown in *Output 12-15*. For Locator prefix 2001:db8:128:3::/64 which uses the algorithm 128 delay metric, R1 is able to compute a basic LFA. The primary path uses the link local address of R4 with an egress interface of 1/1/c2/1 while the backup path, denoted by the `(B)` flag uses the link local address of R2 with an egress interface of 1/1/c1/1.

Output 12-15: FP-Tunnel-Table Entry for R3's Algorithm 128 Locator Prefix

```
A:admin@R1# show router fp-tunnel-table 1 ipv6 2001:db8:128:3::/64

===============================================================================
IPv6 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                Protocol         Tunnel-ID
  Lbl/SID
    NextHop                                                 Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
2001:db8:128:3::/64                        SRV6              524319
  -
    fe80::5054:ff:fec0:8c57-"link-to-R4"                     1/1/c2/1:100
  -
    fe80::265:ffff:fe00:0-"link-to-R2"(B)                    1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

When computing a TI-LFA backup path that requires a repair tunnel, SR-OS inserts a SID into the SRH. For example, *Output 12-16* shows R1's FP tunnel-table entry for R3's algorithm 0 Locator prefix 2001:db8:1:3::/64. As the path is computed using the base shortest path first algorithm, the primary path is via R1-R2-R3 with an egress interface of 1/1/c1/1, but it is not possible to compute a basic LFA backup. The LFA backup therefore uses an RLFA repair tunnel to 2001:db8:1:5:0:1000:: which represents the End function for router R5 with an egress interface of 1/1/c2/1 towards router R4. This SID 2001:db8:1:5:0:1000 will be used in the IPv6 destination address, while any destination SIDs associated with the Locator prefix 2001:db8:1:3::/64 will be carried in the SRH.

Output 12-16: FP-Tunnel-Table for R3's Algorithm 0 Locator Prefix

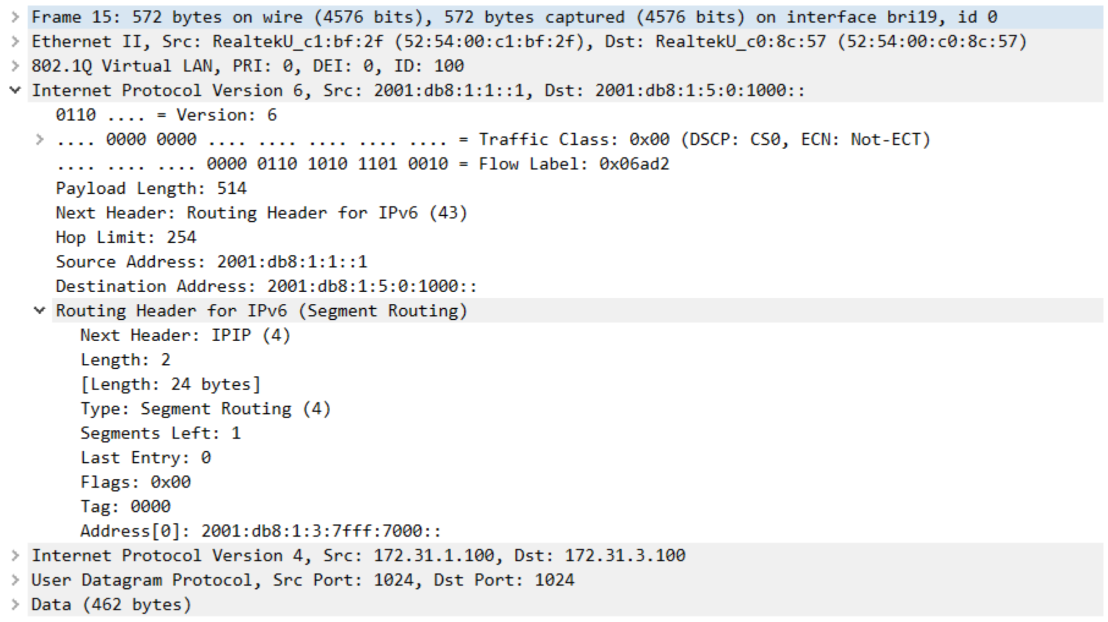
```
A:admin@R1# show router fp-tunnel-table 1 ipv6 2001:db8:1:3::/64

===============================================================================
IPv6 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                                 Protocol         Tunnel-ID
  Lbl/SID
    NextHop                                                  Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
2001:db8:1:3::/64                           SRV6              524318
  -
    fe80::265:ffff:fe00:0-"link-to-R2"                        1/1/c1/1:100
  2001:db8:1:5:0:1000::
    fe80::5054:ff:fec0:8c57-"link-to-R4"(B)                   1/1/c2/1:100
-------------------------------------------------------------------------------
Total Entries : 1
```

To validate the LFA repair path with an SRH shown in *Output 12-16* traffic is forwarded from CE1 (172.31.1.100) towards CE3 (172.31.3.100) using an End.DT4 function of R3's algorithm 0 Locator prefix. The link 1/1/c1/1 on the primary path R1-R2 is then failed and a packet capture taken on the R1-R4 link. *Figure 12-11* shows the first packet of the traffic towards CE3 captured on the R1-R4 link with the LFA repair path active. The outer IPv6 header has a source address of R1 (2001:db8:1:1::1) and a destination address of R5's End Function 2001:db8:1:5:0:1000:: with a Next Header field indicating an IPv6 Routing Header. The Routing Header itself is a Type 4 header (SRH) with the Segments Left field set to 1, and a single segment of R3's End.DT4 SID (2001:db8:1:3:7fff:7000::). When the packet reaches R5 it will copy this segment into the IPv6 destination address and decrement the Segments Left field to 0 before forwarding the packet towards R3.

Figure 12-11: Packet Capture of LFA Repair Path with SRH



```
> Frame 15: 572 bytes on wire (4576 bits), 572 bytes captured (4576 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_c1:bf:2f (52:54:00:c1:bf:2f), Dst: RealtekU_c0:8c:57 (52:54:00:c0:8c:57)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
v Internet Protocol Version 6, Src: 2001:db8:1:1::1, Dst: 2001:db8:1:5:0:1000::
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0110 1010 1101 0010 = Flow Label: 0x06ad2
    Payload Length: 514
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 254
    Source Address: 2001:db8:1:1::1
    Destination Address: 2001:db8:1:5:0:1000::
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPIP (4)
      Length: 2
      [Length: 24 bytes]
      Type: Segment Routing (4)
      Segments Left: 1
      Last Entry: 0
      Flags: 0x00
      Tag: 0000
      Address[0]: 2001:db8:1:3:7fff:7000::
> Internet Protocol Version 4, Src: 172.31.1.100, Dst: 172.31.3.100
> User Datagram Protocol, Src Port: 1024, Dst Port: 1024
> Data (462 bytes)
```

As the last two sub-sections have covered the advertisement of Locators and support of LFA for SRv6, this would appear to be the appropriate time to highlight a potential issue when advertising Locators between IS-IS levels where LFA is in use.

When an End SID is redistributed between levels and RLFA/TI-LFA it is necessary to know if an End SID was redistributed to avoid the incorrect calculation of a repair path. In SR-OS, IS-IS will not use a redistributed End SID for calculating RLFA/TI-LFA repair paths. The SRv6 Locator TLV has a D-bit in the Flags field which when set to 1 indicates that the Locator has been redistributed from Level 2 to Level 1. Hence such an End SID would not be used for RLFA/TI-LFA as it is recognisable as having been redistributed. However, when a Locator TLV is advertised in the opposite direction from Level 1 to Level 2 there is no such indication. As a result, if the advertising Level-1/2 router does not advertise any End SIDs of its own, a router in the level receiving the prefix could potentially calculate an incorrect repair path using the redistributed End SID. For example, consider the following simple topology where R1 is a Level 1 router, R2 is a Level 1/2 router, and R3, R4, and R5 are all Level 2 only.

```
R1--------R2--------R4
           |          |
          R3--------R5
<--- L1 ---><--- L2 --->
```

R1 advertises a Locator TLV with End SID ‘End SID.R1’. Router R2 redistributes End SID.R1 into Level 2 as its own LSP with no indication that it is a redistributed prefix/End SID. This could potentially lead to a situation where R5 computes an RLFA repair tunnel to protect link R5-R4 and uses End SID.R1 as the repair tunnel endpoint. If the link R5-R4 were to fail, R5 would forward traffic on the repair path with an IPv6 destination address of End SID.R1. When R2 receives this traffic, it will simply forward the traffic to R1 and things are clearly broken.

To avoid this potential situation, the recommendation is to always include the Prefix Attributes sub-TLV discussed in chapter 3 using the command **prefix-attributes-tlv true** within the IS-IS context. The Prefix Attributes sub-TLV contains an R-flag that is set when the prefix has been leaked from one level to another (either upwards or downwards). As a result, redistributed prefixes containing the Prefix-Attributes sub-TLV will not be used for RLFA/TI-LFA purposes. Note that this is not an issue that is attributable to SR-OS. It is an IETF standards track issue that is recognised and will need to be addressed.

Note: This situation does not arise when using SR-MPLS. The Prefix-SID sub-TLV flags field contains the R-bit (Readvertisement flag) which is set when the attached Prefix-SID has been propagated by the router from another IS-IS level (upwards or downwards).

Services

SR-OS currently provides SRv6 support for VPRN (End.DT4, End.DT6), VPRN services with EVPN-IFL (again End.DT4, End.DT6), EVPN-VPLS (End.DT2U, End.DT2M), EVPN-VPWS (End.DX2), and IPv4/IPv6 BGP routes (End.DT4, End.DT6) in the base routing instance. This section will provide an example of an SRv6 enabled native VPRN service and also SRv6 applied to BGP routes in the base routing instance.

VPRN Service

A VPRN service is created between R1 and R3 and configured to use SRv6. Within the base **vprn** context a **segment-routing-v6** is configured followed by an integer that can either be 1 or 2. SR-OS allows for up to two instances to allow for a scenario where an SRv6 router is straddling two SRv6 domains and interworking between them. Within the **segment-routing-v6** context a **locator** is configured, which must reference a previously configured **locator** instance. Within the **locator** context the relevant Endpoint behaviours are enabled under the **function** context. In this example, both **end-dt4** and **end-dt6** Endpoint behaviours are enabled to support a VPRN that is running both IPv4 and IPv6. Within the **bgp-ipvpn** context the same **segment-routing-v6** instance is referenced, under which the conventional VPRN service parameters such as **route-distinguisher** and **vrf-import/export** policies are configured. An **srv6** context then allows for the assignment of a **default-locator** to each configured SRv6 instance. The use of the **default-locator** means that in the current SR-OS release the association of prefixes to a particular Flex-Algorithm identifier has per-VPRN granularity. Future releases will allow for overriding the Locator in the received VPN-IPv4/VPN-IPv6 prefix.

Output 12-17: VPRN Service with SRv6

```
    service {
        vprn "one" {
            segment-routing-v6 1 {
                locator "Alg128-Locator" {
                    function {
                        end-dt4 {
                        }
                        end-dt6 {
                        }
                    }
                }
            }
            bgp-ipvpn {
                segment-routing-v6 1 {
                    admin-state enable
                    route-distinguisher "64496:1"
                    vrf-import {
                        policy ["vrf-one-import"]
                    }
                    vrf-export {
                        policy ["vrf-one-export"]
                    }
                    srv6 {
                        instance 1
                        default-locator "Alg128-Locator"
                    }
                }
            }
        }
    }
```

CE3 advertises IPv4 prefix 172.31.3.0/24 to R3, which in turn advertises that prefix as a VPN-IPv4 route towards RR1. Recall that the core network topology uses IPv6 for BGP peering, so in order to advertise and receive IPv4 routes with IPv6 next-hop addresses the commands **advertise-ipv6-next-hops** and **extended-nh-encoding** described in chapter 11 need to be configured at BGP, group, or neighbour level. *Output 12-18* shows the VPN-IPv4 prefix received at R1, truncated to show only the relevant parts. The next-hop for the received prefix is R3's IPv6 system address 2001:db8:33ac:2200::3:3. An SRv6 node advertising a BGP prefix can set the next-hop to any of its IPv6 addresses, and that next-hop address may or may not be covered by a Locator. If the next-hop is not covered by the SRv6 Locator a receiving router will resolve next-hop reachability in the IGP in order for the prefix to be considered valid, but it is not used to resolve the prefix to a tunnel. The job of resolving the prefix to a tunnel is solely down to the advertised SID and the Locator block derived from that SID, and the router will attempt to resolve that Locator prefix independently. The `Sid` field in the SRv6 SID Information shows the advertised Locator prefix 2001:db8:128:3::, which is formed of a 48-bit Locator Block (`Loc-Block-Len`) and 16-bit Locator Node (`Loc-Node-Len`). The Function follows the Locator, and as shown in the `Func-Len` field it is the default 20-bits in length. This 20-bits is shown in the `Full Sid` field as being hex 7fff:2 and is transposed into the NLRI MPLS label, shown in the `VPN Label` as the decimal equivalent 524274. The resulting transposition offset is shown in the `Tpose-offset` field as 64-bits. As previously described, the `Full Sid` field is not an advertised value because the Function part is transposed. It is provided in the CLI output purely for operator ease to show a concatenated view of the entire SID constructed of Locator, Function, and Argument.

Output 12-18: VPN-IPv4 Prefix Received at R1

```
A:admin@R1# show router bgp routes 172.31.3.0/24 vpn-ipv4 detail
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP VPN-IPv4 Routes
===============================================================================
Original Attributes

Network         : 172.31.3.0/24
Nexthop         : 2001:db8:33ac:2200::3:3
Route Dist.     : 64496:1                VPN Label      : 524274
Path Id         : None
From            : 2001:db8:33ac:2200::3:a
 --- [snip] ---
SRv6 TLV Type   : SRv6 L3 Service TLV (5)
SRv6 SubTLV     : SRv6 SID Information (1)
Sid             : 2001:db8:128:3::
Full Sid        : 2001:db8:128:3:7fff:2000::
Behavior        : End.DT4 (19)
SRv6 SubSubTLV  : SRv6 SID Structure (1)
Loc-Block-Len   : 48                     Loc-Node-Len   : 16
Func-Len        : 20                     Arg-Len        : 0
Tpose-Len       : 20                     Tpose-offset   : 64
VPRN Imported   :  1
```

The received VPN-IPv4 prefix can thereafter be seen in the VPRN route-table at R1 with its associated SRv6 SID and an indication that traffic towards that destination is tunnelled with SRv6.

Output 12-19: R1's Route-Table Entry for VPN-IPv4 Prefix

```
A:admin@R1# show router 1 route-table 172.31.3.0/24

===============================================================================
Route Table (Service: 1)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
172.31.3.0/24                                 Remote  BGP VPN   00h00m07s  170
       2001:db8:128:3:7fff:2000:: (tunneled:SRV6)                   40001
```

BGP in the Base Routing Instance

An IES service is created on R4 and R6 with IPv4 interfaces to CE4 and CE6 respectively. IPv4 BGP sessions are established between CE4 and R4, and between CE6 and R6. CE4 advertises prefix 172.31.4.0/24 while CE6 advertises prefix 172.31.6.0/24 to their respective BGP peers, which are then advertised through the SRv6 core with IPv6 next-hop addresses and as with the VPRN service example, **advertise-ipv6-next-hops** and **extended-nh-encoding** are enabled under BGP. The configuration of the IES and the BGP peering to each of the CE routers are not described here as this is generic router and service configuration. The parts of the configuration necessary to enable SRv6 for IPv4 BGP prefixes is shown in *Output 12-20* and is applied at R4 and R6. Within the base router **bgp** context a **segment-routing-v6 context** is enabled. Within that context the **family** command allows for selection of either **ipv4** or **ipv6** address families, and in this example is **ipv4** only. Under each **family** the **add-**

srv6-tlvs command instructs the router to add the SRv6 Service TLVs for IPv4 prefixes advertised in BGP and references an existing Locator with the **locator-name** command. The **ignore-received-srv6-tlvs** is set to **false** in which case the system will process received SRv6 Service TLVs and use the Locator for next-hop programming of resolved prefixes. If set to **true**, the system will ignore any received SRv6 Service TLVs and use the received next-hop to resolve the prefix. The referenced **locator** 'Alg0-Locator' is also included in the output within the **base-routing-instance** context, mainly to highlight the need to configure the **end-dt4 function** in order to enable that Endpoint behaviour for this locator. When enabling the **end-dt4** function a sub-context is created under which a static Function **value** can be assigned in the range 1-1048575. If no explicit **value** is specified, a dynamic value is assigned.

Output 12-20: SRv6 Configuration for Base Router BGP

```
    router "Base" {
        bgp {
            segment-routing-v6 {
                family ipv4 {
                    ignore-received-srv6-tlvs false
                    add-srv6-tlvs {
                        locator-name "Alg0-Locator"
                    }
                }
            }
        }
        segment-routing {
            segment-routing-v6 {
                base-routing-instance {
                    locator "Alg0-Locator" {
                        function {
                            end-dt4 {
                            }
                        }
                    }
                }
            }
        }
    }
```

Although not shown in *Output 12-20*, the option exists under the **segment-routing-v6** context at both BGP **group** and **neighbour** level to advertise routes to certain peers with the SRv6 TLVs removed using the command **route-advertisement family strip srv6-tlvs true**. Equally, if required, it is possible to drop routes from certain peers that contain SRv6 TLVs using the command **route-advertisement drop-routes-with-srv6-tlvs true.** The intent of these commands is to allow for routers in the network that may not be able to properly process BGP updates containing SRv6 information.

Output 12-21 shows prefix 172.31.6.0/24 received in IPv4 BGP at R4, truncated to show only the relevant parts. The next-hop for the received prefix is R6's IPv6 system address 2001:db8:33ac:2200::3:6. The `Sid` field in the SRv6 SID Information shows the advertised SID, and since transposition cannot be used for unlabelled BGP prefixes the complete SID is shown as advertised with an endpoint behaviour of End.DT4. The advertised SID uses the algorithm 0 Locator prefix of 2001:db8:1::/48 and the prefix will therefore be encapsulated in SRv6 and routed in the base shortest path first algorithm. The Locator block length (`Loc-Block-Len`) is 48-bits, which together with a 16-bit Node length (`Loc-Node-Len`) yields a locator prefix of 64-bits, while the Function length (`Func-Len`) is the SR-OS default of 20-bits.

Output 12-21: IPv4 BGP Prefix Received at R4

```
A:admin@R4# show router bgp routes 172.31.6.0/24 detail
===============================================================================
 BGP Router ID:192.0.2.4        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP IPv4 Routes
===============================================================================
Original Attributes

Network         : 172.31.6.0/24
Nexthop         : 2001:db8:33ac:2200::3:6
Path Id         : None
From            : 2001:db8:33ac:2200::3:a
 --- [snip] ---
SRv6 TLV Type   : SRv6 L3 Service TLV (5)
SRv6 SubTLV     : SRv6 SID Information (1)
Sid             : 2001:db8:1:6:7fff:d000::
Behavior        : End.DT4 (19)
SRv6 SubSubTLV  : SRv6 SID Structure (1)
Loc-Block-Len   : 48                    Loc-Node-Len   : 16
Func-Len        : 20                    Arg-Len        : 0
Tpose-Len       : 0                     Tpose-offset   : 0
```

The received prefix can thereafter be seen in the base routing instance with its associated SRv6 SID and an indication that traffic towards that destination is tunnelled over SRv6.

Output 12-22: Route-Table Entry for Received IPv4 Prefix

```
A:admin@R4# show router route-table 172.31.6.0/24

===============================================================================
Route Table (Router: Base)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
172.31.6.0/24                                 Remote  BGP       20h03m58s  170
       2001:db8:1:6:7fff:d000:: (tunneled:SRV6)                       201
```

SRv6 OAM

Unlike SR-MPLS services no extensions are required in order to deliver OAM capabilities for SRv6. It uses native IP and therefore the classic tools such as ping and traceroute provide the troubleshooting and connectivity verification tools. SR-OS will respond to ping and traceroute targeted at any locally instantiated Endpoint. For example, R6's End Function has the SID 2001:db8:1:6:0:1000::

Output 12-23: R6's End Function SID

```
A:admin@R6# show router segment-routing-v6 local-sid locator Alg0-Locator end

===============================================================================
Segment Routing v6 Local SIDs
===============================================================================
SID                                                 Type           Function
  Locator
  Context
-------------------------------------------------------------------------------
2001:db8:1:6:0:1000::                               End            1
  Alg0-Locator
  Base
-------------------------------------------------------------------------------
SIDs : 1
-------------------------------------------------------------------------------
```

From router R1, initiate a ping to R6's End SID.

Output 12-24: Ping from R1 to R6 End Function SID

```
A:admin@R1# ping 2001:db8:1:6:0:1000::
PING 2001:db8:1:6:0:1000:: 56 data bytes
64 bytes from 2001:db8:1:6:0:1000:: icmp_seq=1 hlim=62 time=3.77ms.
64 bytes from 2001:db8:1:6:0:1000:: icmp_seq=2 hlim=62 time=3.74ms.
64 bytes from 2001:db8:1:6:0:1000:: icmp_seq=3 hlim=62 time=3.29ms.
64 bytes from 2001:db8:1:6:0:1000:: icmp_seq=4 hlim=62 time=3.43ms.
64 bytes from 2001:db8:1:6:0:1000:: icmp_seq=5 hlim=62 time=2.97ms.

---- 2001:db8:1:6:0:1000:: PING Statistics ----
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min = 2.97ms, avg = 3.44ms, max = 3.77ms, stddev = 0.297ms
```

Similarly, a traceroute can be targeted to R6's End SID.

Output 12-25: Traceroute from R1 to R6 End Function SID

```
A:admin@R1#  traceroute 2001:db8:1:6:0:1000::
traceroute to 2001:db8:1:6:0:1000::, 30 hops max, 60 byte packets
  1  2001:db8:33ac:2200::4:a (2001:db8:33ac:2200::4:a)    1.73 ms  2.00 ms  2.10 ms
  2  2001:db8:33ac:2200::4:16 (2001:db8:33ac:2200::4:16)    2.89 ms  2.71 ms  2.91 ms
  3  2001:db8:33ac:2200::3:6 (2001:db8:33ac:2200::3:6)    3.95 ms  4.74 ms  4.12 ms
```

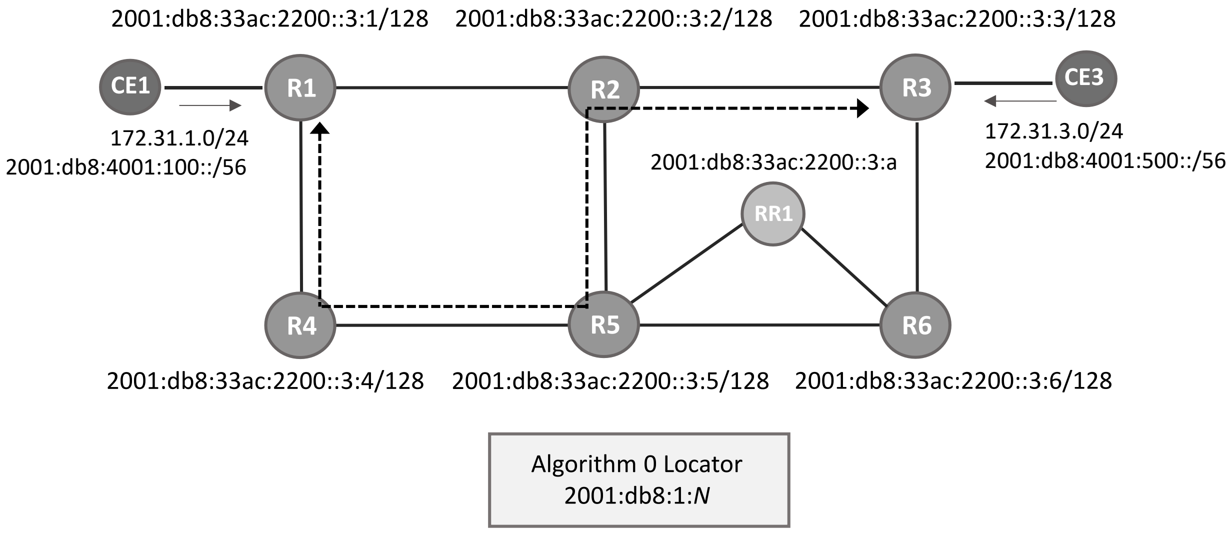
SRv6 Policies

This chapter has so far discussed the use of SRv6 paths consisting of a single SRv6 SID, where traffic engineering can only be achieved using Flex-Algorithm. In some scenarios however, there may be a requirement for more defined source-routed paths consisting of a number of SRv6 SIDs. In SR-OS this capability is supported using SR policies as described in Chapter 7 but consisting of 128-bit SRv6 SIDs instead of MPLS labels. Like their SR-MPLS counterparts, these SRv6 Policies can be configured statically, or they can be learned through BGP. The supported SR Policy headend behaviours include H.Encaps.Red and H.Encaps.L2.Red. The '.Red' suffix of both behaviours indicates support for an optimised encapsulation that reduces the length of the SRH. This is achieved by excluding the first SID from the SRH of the pushed IPv6 header, and instead placing it only in the destination address field of the IPv6 header (hence the '.Red' suffix indicates a reduced SRH). The H.Encaps.Red behaviour is valid for layer 3 services, and provides support for IPv4 and IPv6 VPRNs, EVPN Interface-Less (IFL), and BGP shortcuts. The H.Encaps.L2.Red behaviour is applicable to Layer 2 Ethernet frames and provides support for EVPN-VPLS and EVPN-VPWS. There is additional support for End.B6.Encaps.Red at an intermediate router functioning as a BSID anchor to encapsulate SRv6 Policy traffic into an SRv6 Policy towards a downstream endpoint.

To illustrate the use of SRv6 Policies, the network topology shown in *Figure 12-12* is used. Routers R1 to R6 and a Route-Reflector, RR1, are in a flat IS-IS Level 2 area, and all IGP link metrics are 100. All routers belong to Autonomous System 64496 and Routers R1 to R6 peer in IBGP with RR1 as Route-Reflector clients for advertisement of service routes. The network is dual-stack IPv4/IPv6 and all BGP peering uses IPv6 addressing. SRv6 is enabled throughout the domain as previously described within this chapter, with the base algorithm 0 configured to use the locator block 2001:db8:1::/48 and a 16-bit node length to yield a total 64-bit locator length. Node numbers are 1 to 6 for routers R1-R6 respectively. For example, R1's advertised algorithm 0 locator is 2001:db8:1:1::/64, while R2's algorithm 0 locator is 2001:db8:1:2::/64.

A static SRv6 Policy is initially extended between R1 and R3 along the path R1-R4-R5-R2-R3 and vice versa before moving on to look at the use of BGP to advertise SRv6 Policies. CE routers are attached to R1 and R3 and belong to a VPRN service which will use the afore-mentioned SRv6 Policies. Both of these CE routers advertise IPv4 and IPv6 prefixes to their adjacent routers which will be used to verify that SRv6 Policies are functioning as anticipated.

Figure 12-12: Network Topology for SRv6 Policy


2001:db8:33ac:2200::3:1/128
2001:db8:33ac:2200::3:2/128
2001:db8:33ac:2200::3:3/128
CE1
R1
R2
R3
CE3
172.31.1.0/24
2001:db8:4001:100::/56
2001:db8:33ac:2200::3:a
172.31.3.0/24
2001:db8:4001:500::/56
RR1
R4
R5
R6
2001:db8:33ac:2200::3:4/128
2001:db8:33ac:2200::3:5/128
2001:db8:33ac:2200::3:6/128
Algorithm 0 Locator
2001:db8:1:*N*

Static SRv6 Policies

Output 12-26 shows the configuration of the static SRv6 Policy at router R1, with a similarly configured SRv6 Policy at R3. The **static-policy** is configured under the **segment-routing sr-policies** context in exactly the same way as MPLS-based SR Policies, with the **type** command providing the option to configure **srv6** to distinguish the SR Policy from the default type of **mpls**. The SR Policy parameters such as **headend**, **color**, and **distinguisher** are described in Chapter 7 and are therefore not repeated here. The **endpoint** is an IPv6 address and is the system address of router R3. This is the same address that R3 uses as a next-hop when advertising VPN-IPv4 and VPN-IPv6 prefixes, hence the SR Policy will need to resolve to that next-hop address. The **segment-routing-v6** context provides the context to configure the **binding-sid**. The **binding-sid** parameter is followed by an index, although currently SR-OS only supports a single binding SID per SRv6 Policy. The **binding-sid** can be entered in one of two formats, using either **locator** or **ip-address**, depending on the value of the **headend** command:

- If the headend is **local**, the **binding-sid** is entered as a **locator** that is configured on the system, and requires a **locator-name** and a **function**, which can currently only be **end-b6-encaps-red**. The **locator** context has an optional **function-value** parameter where the function value can be statically defined. If no **function-value** is configured, the system automatically allocates one.
- If the headend is a non-local IP address, the **binding-sid** must be entered as an **ip-address**, where that address is a 128-bit IPv6 address.

The **segment-list** context provides the ability to construct a path consisting of 128-bit SIDs, which can correspond to a 128-bit SID of any function type (node-SID, Adj-SID, binding-SID). A maximum of three segments can be configured and supported within each segment-list. The first of these segments will be placed into the IPv6 destination address of the SRH header as a result of the (H.Encaps.Red/H.Encaps.L2.Red) reduced SRH behaviour. The remaining two segments, together with the service SID represent the current SR-OS limit of three SRv6 SIDs carried in an SRH. Multiple segment-lists can be configured for the purpose of ECMP or UCMP (defined by the **weight** parameter). The example in *Output 12-26* contains three segments, each containing an SRv6 SID. The first SID is 2001:db8:1:4:7fff:d000:: which is R4's End.X SID for the link towards R5. The second SID is 2001:db8:1:5:0:5000:: which is R5's End.X SID for the link towards R2. The third and final SID is 2001:db8:1:2:0:6000:: which is R2's End.X SID for the link towards R3. Hence, the SRv6 Policy defines a path R1-R4-R5-R2-R3 consisting of End.X SIDs.

Output 12-26: Static SRv6 Policy at R1 (to R3)

```
    router "Base" {
        segment-routing {
            sr-policies {
                static-policy "R1-to-R3-SRv6" {
                    admin-state enable
                    color 100
                    endpoint 2001:db8:33ac:2200::3:3
                    head-end local
                    distinguisher 12345
                    type srv6
                    segment-routing-v6 {
                        binding-sid 1 {
                            locator {
                                locator-name "Alg0-Locator"
                                function end-b6-encaps-red
                            }
                        }
                    }
                    segment-list 1 {
                        admin-state enable
                        segment 1 {
                            srv6-sid 2001:db8:1:4:7fff:d000::
                        }
                        segment 2 {
                            srv6-sid 2001:db8:1:5:0:5000::
                        }
                        segment 3 {
                            srv6-sid 2001:db8:1:2:0:6000::
                        }
                    }
                }
            }
        }
    }
```

Finally, the **segment-list** and **static-policy** are placed into an **admin-state** of **enable**. *Output 12-27* shows the operational state of the static SRv6 Policy. The `active` field shows whether this candidate path is the selected path in the presence of multiple candidate paths. The SRv6 Policy segment list is considered valid if the headend is able to perform path resolution for the first SID in the segment list into one or more outgoing interfaces and next-hops. The segment 1 SRv6 SID is 2001:db8:1:4:7fff:d000:: and the state is shown as `resolved-up`, indicating that this is a valid segment list.

Output 12-27: Operational State of SRv6 Policy at R1

```
A:admin@R1# show router segment-routing sr-policies static color 100 end-point
2001:db8:33ac:2200::3:3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Type            : srv6
Active          : Yes                   Owner           : static
Color           : 100
Head            : 0.0.0.0               Endpoint Addr   : 2001:db8:33ac:2200::*
RD              : 12345                 Preference      : 100
SRv6 BSID 1     : 2001:db8:1:1:7fff:f000::
TunnelId        : 917509                Age             : 5759
Origin ASN      : 0                     Origin          : 0.0.0.0
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 08/24/2022 13:52:36
Maintenance Policy: N/A

Path Segment Lists:
Segment-List    : 1                     Weight          : 1
S-BFD State     : Down                  S-BFD Transitio*: 0
Num Segments    : 3                     Last Change     : 08/22/2022 14:31:14
  Seg 1 SID   : 2001:db8:1:4:7fff:d000::                 State : resolved-up
  Seg 2 SID   : 2001:db8:1:5:0:5000::                    State : N/A
  Seg 3 SID   : 2001:db8:1:2:0:6000::                    State : N/A
```

If the policy is considered valid, it is populated in the IPv6 tunnel-table with an owner of `srv6-policy`. The entry indicates the destination and color and always has a metric value of 0 regardless of how the SRv6 Policy is instantiated. The 0 metric is used simply because there is no effective way for the headend to determine a more reflective value for an SRv6 Policy when learnt through BGP or statically configured. The preference value assigned to SRv6 Policy is 14 and is currently non-configurable.

Output 12-28: IPv6 Tunnel-Table Entry for SRv6 Policy

```
A:admin@R1# show router tunnel-table ipv6 protocol srv6-policy

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                     Owner     Encap TunnelId  Pref
Nexthop                                         Color           Metric
-------------------------------------------------------------------------------
2001:db8:33ac:2200::3:3/128                     srv6-pol  SRV6  917509    14
  fpe_1.a                                         100             0
-------------------------------------------------------------------------------
```

Advertising SRv6 Policies in BGP

SRv6 policies can be advertised in BGP using the 'SR TE Policy' (SAFI 73) in largely the same manner as described for SR-MPLS Policies in chapter 7 with a few notable exceptions:

- SRv6 policies are advertised with AFI 2 for IPv6 (as opposed to AFI 1 for IPv4).
- SRv6 policies are distinguished from SR (MPLS) policies by encoding an SRv6 Binding SID Sub-TLV within the Tunnel Encapsulation Attribute instead of a Binding SID Sub-TLV.
- The specification for 'SR TE Policy' [35] defines a number of segment types for segments contained within an SRv6 Policy Segment-List. SRv6 policies use a Type B segment type, which defines a SID only in the form of an IPv6 address.

The other parameters contained within an SRv6 Policy are the same as MPLS-based SR policies, although the manner in which they are structured and encoded may differ. To illustrate how SRv6 policies are advertised in BGP, an SRv6 Policy is advertised by RR1 to headend R1 with an endpoint of R3 and a segment-list that defines the path R1-R4-R5-R2-R3 as shown in *Figure 12-12*. When an SR-OS router advertises SRv6 policies into BGP they must first be statically configured in order to provide the relevant information to populate the BGP attributes, and this static policy is shown in *Output 12-29*. The policy has a **headend** of 192.0.2.1 (R1) which must always be an IPv4 address to match the head-end BGP Router ID, and an **endpoint** of R3's IPv6 system address 2001:db8:33ac:2200:3::3. This is the IPv6 address that R3 peers in BGP with and will be the next-hop of advertised prefixes, hence this is the address that the SRv6 Policy must resolve. The **color** is 100 and the **distinguisher** is 23456. If a second candidate path were to be advertised for this SRv6 Policy, a different distinguisher would be required to create a unique NLRI and avoid the route being suppressed. The **type** is set to **srv6** to distinguish it from an MPLS-based SR Policy and means that the binding SID will be carried in an SRv6 Binding SID Sub-TLV. As previously described in this section, the **binding-sid** can be entered as either **locator** or **ip-address** depending on whether the **headend** is a local or non-local IP address. In this example, the **headend** is a non-local IP address, so the **binding-sid** must be entered as an **ip-address**. The configured **ip-address** must match a locator prefix on the headend router and must carry a function length of 20-bits followed by argument bits that are set to zero. The **ip-address** shown in this example is 2001:db8:1:1:0:a000:: and is constructed as follows:

- 2001:db8:1:1 represents the **ip-prefix** assigned to the algorithm 0 SRv6 Locator at R1.
- The 20-bit function length following the locator prefix and shown as 0:a is a decimal function of 10. This function must fall within the range of the algorithm 0 locator **static-function max-entries** and must not be a previously allocated static function value. In this example, R1 is configured with a locator **static-function max-entries** value of 100, so the function of 10 falls within this range (and is not previously allocated).
- The argument bits following the function are all set to zero.

The **segment-list** contains three **segments** and essentially replicates the path used previously in this section for the static SRv6 Policy at R1. The first SID is 2001:db8:1:4:7fff:d000:: which is R4's End.X SID for the link towards R5. The second SID is 2001:db8:1:5:0:5000:: which is R5's End.X SID for the link towards R2. The third and final SID is 2001:db8:1:2:0:6000:: which is R2's End.X SID for the link towards R3. Hence, the SRv6 Policy defines a path R1-R4-R5-R2-R3 consisting of End.X SIDs. Finally, the **static-policy** is placed into an **admin-state** of **enable**.

Output 12-29: Static SRv6 Policy at RR1 for Advertisement into BGP

```
    router "Base" {
        segment-routing {
            sr-policies {
                static-policy "R1-to-R3-SRv6" {
                    admin-state enable
                    color 100
                    endpoint 2001:db8:33ac:2200::3:3
                    head-end 192.0.2.1
                    distinguisher 23456
                    type srv6
                    segment-routing-v6 {
                        binding-sid 1 {
                            ip-address 2001:db8:1:1:0:a000::
                        }
                    }
                    segment-list 1 {
                        admin-state enable
                        segment 1 {
                            srv6-sid 2001:db8:1:4:7fff:d000::
                        }
                        segment 2 {
                            srv6-sid 2001:db8:1:5:0:2000::
                        }
                        segment 3 {
                            srv6-sid 2001:db8:1:2:0:4000::
                        }
                    }
                }
            }
        }
    }
```

To advertise the static policy into BGP, two steps are required at RR1. Firstly, the command **sr-policy-import true** must be configured under the global BGP context. This command instructs BGP to import all statically configured non-local SR Policies from the SR database into the BGP RIB such that they can be advertised towards BGP peers supporting the SR TE

Policy Address Family. Secondly, the BGP peering with R1 (2001:db8:33ac:2200::3:1) must have the **sr-policy-ipv6 true** command enabled within the **family** context to be able to support the SR TE Address Family for IPv6.

Output 12-30: BGP Configuration for Advertising SRv6 Policy

```
        bgp {
            sr-policy-import true
            group "IPv6-CLIENTS" {
                family {
                    sr-policy-ipv6 true
                }
            }
            neighbor "2001:db8:33ac:2200::3:1" {
                group "IPv6-CLIENTS"
            }
        }
```

Debug 12-1 shows the BGP Update for the SRv6 Policy advertised from RR1 (2001:db8:33ac:2200::3:a) as received at R1. The address family is SR_POLICY_IPv6 (SAFI 73, AFI 2) and the NLRI contains the distinguisher (shown as `RD`), color, and endpoint. Note the presence of an IPv4 address specific Route-Target Extended Community encoding the headend R1's address (192.0.2.1), which allows for potential constraining of route propagation as required. The SRv6 Policy attributes such as Preference, SRv6 BSID, and Segment List are encoded as sub-TLVs of the Tunnel-Encapsulation Attribute.

Debug 12-1: BGP Update at R1 Showing SRv6 Policy

```
16 2022/08/29 12:59:20.567 BST minor: DEBUG #2001 Base Peer 1: 2001:db8:33ac:2200::3:a
Peer 1: 2001:db8:33ac:2200::3:a: UPDATE
Peer 1: 2001:db8:33ac:2200::3:a - Received BGP UPDATE:
    Withdrawn Length = 0
    Total Path Attr Length = 190
    Flag: 0x90 Type: 14 Len: 46 Multiprotocol Reachable NLRI:
        Address Family SR_POLICY_IPV6
        NextHop len 16 Global NextHop 2001:db8:33ac:2200::3:a
        [Len 192] RD: 23456 Color: 100 Endpoint: 2001:db8:33ac:2200::3:3
    Flag: 0x40 Type: 1 Len: 1 Origin: 0
    Flag: 0x40 Type: 2 Len: 0 AS Path:
    Flag: 0x40 Type: 5 Len: 4 Local Preference: 100
    Flag: 0xc0 Type: 16 Len: 8 Extended Community:
        target:192.0.2.1:0
    Flag: 0xc0 Type: 23 Len: 112 Tunnel-Encapsulation-attr:
      Tunnel-Encap-Type:- sr-policy
      Preference SubTlv:-
         flags: 0x0  value: 100
      Binding Sid SubTlv:-
         len: 6  flags: 0x0  Label: 0
      SRv6 Binding Sid SubTlv:-
          len: 18  flags: 0x0  SRv6 Sid: 2001:db8:1:1:0:a000::
      Segment List 0 SubTlv:- len: 69 Weight: 1
          Segment 0 -Type 13 Flags 0x0 SID: 2001:db8:1:4:7fff:d000::
          Segment 1 -Type 13 Flags 0x0 SID: 2001:db8:1:5:0:2000::
          Segment 2 -Type 13 Flags 0x0 SID: 2001:db8:1:2:0:4000::
```

The following output is taken at R1 and shows that the SRv6 Policy is active as the first SRv6 SID in the segment list has been correctly resolved (shown as `resolved-up`). The other attributes of the SRv6 Policy are shown together with a tunnel ID. The owner of the policy is BGP.

Output 12-31: Operational State of BGP SRv6 Policy at R1

```
A:admin@R1# show router segment-routing sr-policies bgp color 100 end-point
2001:db8:33ac:2200::3:3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Type            : srv6
Active          : Yes                  Owner           : bgp
Color           : 100
Head            : 0.0.0.0              Endpoint Addr   : 2001:db8:33ac:2200::*
RD              : 23456                Preference      : 100
SRv6 BSID 1     : 2001:db8:1:1:0:a000::
TunnelId        : 917511               Age             : 5409
Origin ASN      : 64496                Origin          : 2001:db8:33ac:2200::*
NumReEval       : 0                    ReEvalReason    : none
NumActPathChange: 0                    Last Change     : 08/29/2022 12:59:21
Maintenance Policy: N/A

Path Segment Lists:
Segment-List    : 1                    Weight          : 1
S-BFD State     : Down                 S-BFD Transitio*: 0
Num Segments    : 3                    Last Change     : 08/25/2022 10:36:53
  Seg 1 SID   : 2001:db8:1:4:7fff:d000::                State : resolved-up
  Seg 2 SID   : 2001:db8:1:5:0:2000::                   State : N/A
  Seg 3 SID   : 2001:db8:1:2:0:4000::                   State : N/A
```

Output 12-32 shows that the SRv6 Policy is correctly populated in the IPv6 tunnel-table. The destination and color are shown. The non-configurable preference is 14, and the metric is shown as 0 for reasons that have previously been described.

Output 12-32: IPv6 Tunnel-Table Entry for SRv6 Policy from R1 to R3

```
A:admin@R1# show router tunnel-table ipv6 protocol srv6-policy

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                    Owner     Encap TunnelId  Pref
Nexthop                                        Color           Metric
-------------------------------------------------------------------------------
2001:db8:33ac:2200::3:3/128                    srv6-pol  SRV6  917511    14
  fpe_1.a                                        100             0
-------------------------------------------------------------------------------
Flags: B = BGP or MPLS backup hop available
       L = Loop-Free Alternate (LFA) hop available
       E = Inactive best-external BGP route
       k = RIB-API or Forwarding Policy backup hop
```

Finally, it's worth highlighting that the SRv6 binding SID from the SRv6 Policy instantiates a local SID with a behaviour of End.B6.Encaps.Red. This can be used to encapsulate upstream SRv6 Policy traffic into another SRv6 Policy towards a downstream endpoint when the router is functioning as a BSID anchor.

Output 12-33: Instantiation of Local-SID With End.B6.Encaps.Red Behaviour

```
A:admin@R1# show router segment-routing-v6 local-sid 2001:db8:1:1:0:a000::

===============================================================================
Segment Routing v6 Local SIDs
```

```
===============================================================================
SID                                                 Type           Function
  Locator
  Context
-------------------------------------------------------------------------------
2001:db8:1:1:0:a000::                               End.b6.encaps* 10
  Alg0-Locator
    None
-------------------------------------------------------------------------------
SIDs : 1
```

Using SRv6 Policies in Services

Regardless of whether SRv6 Policies are statically configured or learned in BGP, once they are programmed in the IPv6 tunnel-table they can be used by VPRN, EVPN-VPLS, EVPN-VPWS, and EVPN-IFL services, as well as BGP shortcuts. The example in this section uses a VPRN service to illustrate how SRv6 Policies are used for services, but the procedure is fundamentally the same or similar for the other listed services.

To enable the VPRN service between R1 and R3 to use the SRv6 Policies as transport, the VPRN-specific SRv6 configuration previously described in this chapter (*Output 12-17*) is applied, with a **default-locator** being referenced in the **bgp-ipvpn segment-routing-v6 context**. However, as shown in *Output 12-34* an additional **resolution** command is introduced that allows the user select whether the service uses SRv6 or SRv6 Policy to resolve BGP service routes. The **tunnel-table** option means that the system will try to resolve BGP routes containing an SRv6 service SID to an SRv6 Policy with a matching color and an endpoint that corresponds to the received BGP next-hop. The **route-table** option means that the system will try to resolve the received SRv6 service SID in the BGP route to an SRv6 locator in the route-table (the fact that SRv6 locators are also installed in the tunnel-table does make this option slightly misleading). The **fallback-tunnel-to-route-table** option does what it says and means that the system will first try resolving the route to an SRv6 Policy in the tunnel-table, and if none is found it will fall back to resolving the route to an SRv6 locator in the route-table. It’s worth emphasising that when selecting one of the **tunnel-table/route-table** options that the system is not just switching between SRv6 or SRv6 Policy transport, it is also switching the way in which a received route is resolved. SRv6 uses the received SRv6 service SID and looks in the route-table for a matching locator. SRv6 Policy uses the BGP next-hop and tries to find a policy with a matching endpoint and color.

Output 12-34: VPRN Configuration for SRv6 Policy

```
    service {
        vprn "one" {
            segment-routing-v6 1 {
                locator "Alg0-Locator" {
                    function {
                        end-dt4 {
                        }
                        end-dt6 {
                        }
                    }
                }
            }
            bgp-ipvpn {
                segment-routing-v6 1 {
                    admin-state enable
                    route-distinguisher "64496:1"
                    resolution tunnel-table
                    vrf-import {
                        policy ["vrf-one-import"]
                    }
                    vrf-export {
                        policy ["vrf-one-export"]
                    }
                    srv6 {
                        instance 1
                        default-locator "Alg0-Locator"
                    }
                }
            }
        }
    }
```

As the example configuration at R1 uses a **resolution** of **tunnel-table** the system will look in the tunnel-table for a matching SRv6 Policy. VPN-IPv4 and VPN-IPv6 routes advertised between R1 and R3 are appended with the Extended Community Color of 100. *Output 12-35* shows R1's IPv4 and IPv6 VPRN route-table containing the prefixes advertised by CE3 (and subsequently R3). Both prefixes are known through BGP VPN, and both have a next-hop of R3's system address 2001:db8:33ac:2200::3:3. As both prefixes also have a color of 100, they match the SRv6 Policy to R3 and are shown as `tunneled:SRv6-Policy:917509`, where 917509 correlates with the tunnel-ID for the SRv6 Policy.

Output 12-35: R1's VPRN Route-Table with IPv4 and IPv6 Prefixes Advertised by R3

```
A:admin@R1# show router 1 route-table 172.31.3.0/24

===============================================================================
Route Table (Service: 1)
===============================================================================
Dest Prefix[Flags]                              Type    Proto     Age        Pref
      Next Hop[Interface Name]                                      Metric
-------------------------------------------------------------------------------
172.31.3.0/24                                   Remote  BGP VPN   02h06m15s  170
       2001:db8:33ac:2200::3:3 (tunneled:SRV6-Policy:917509)        0
-------------------------------------------------------------------------------
No. of Routes: 1

A:admin@R1# show router 1 route-table ipv6 2001:db8:4001:500::/56

===============================================================================
IPv6 Route Table (Service: 1)
===============================================================================
Dest Prefix[Flags]                              Type    Proto     Age        Pref
      Next Hop[Interface Name]                                      Metric
-------------------------------------------------------------------------------
2001:db8:4001:500::/56                          Remote  BGP VPN   02h10m24s  170
       2001:db8:33ac:2200::3:3 (tunneled:SRV6-Policy:917509)        0
-------------------------------------------------------------------------------
No. of Routes: 1
```

To validate the end-to-end data-path through the SRv6 Policy, an IPv4 ping is initiated from CE1 (172.31.1.1) towards CE3 (172.31.3.1).

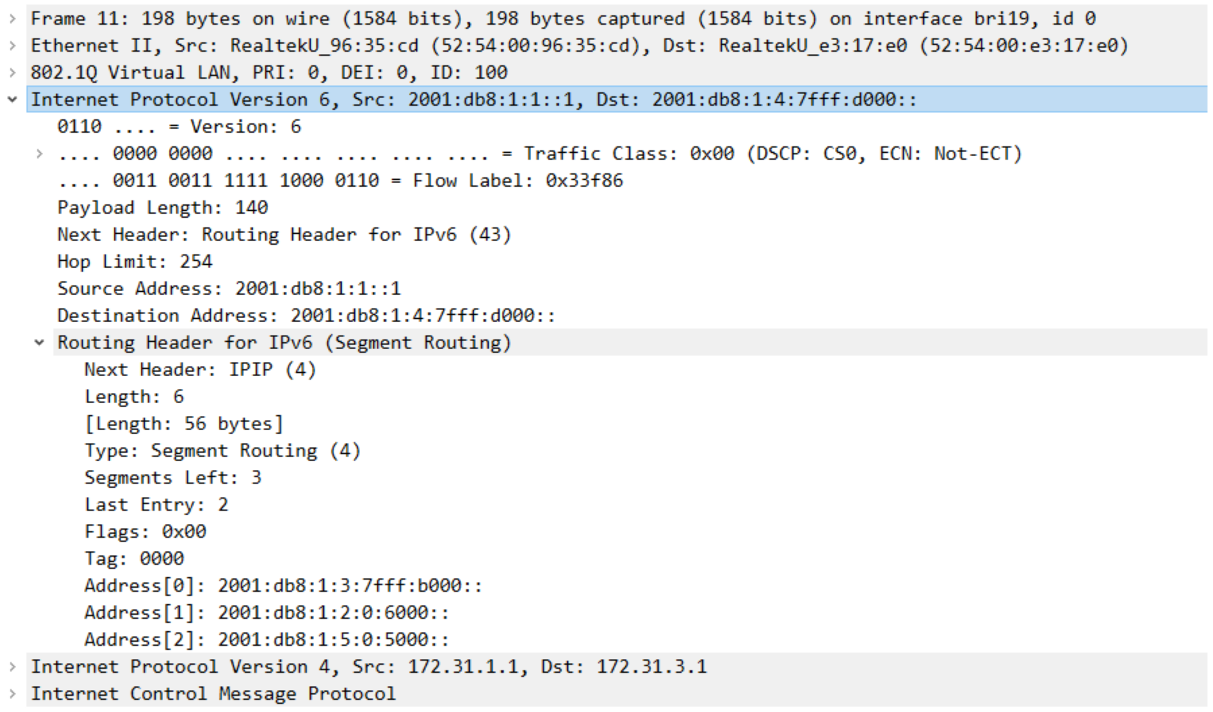
Output 12-36: Ping from CE1 to CE3 to Validate Data-Path Through the SRv6 Policy

```
A:ce1# ping router 200 172.31.3.1 source 172.31.1.1
PING 172.31.3.1 56 data bytes
64 bytes from 172.31.3.1: icmp_seq=1 ttl=62 time=8.81ms.
64 bytes from 172.31.3.1: icmp_seq=2 ttl=62 time=8.34ms.
64 bytes from 172.31.3.1: icmp_seq=3 ttl=62 time=8.77ms.
64 bytes from 172.31.3.1: icmp_seq=4 ttl=62 time=7.87ms.
64 bytes from 172.31.3.1: icmp_seq=5 ttl=62 time=7.93ms.

---- 172.31.3.1 PING Statistics ----
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min = 7.87ms, avg = 8.35ms, max = 8.81ms, stddev = 0.397ms
```

Whilst the ping is in progress, a packet capture is taken on the link between R1 and R4 and a data packet from this capture is shown in *Figure 12-13*. The packet is IPv6 with a source address of 2001:db8:1:1::1, which is the End SID for the algorithm 0 locator. The destination address is 2001:db8:1:4:7fff:d000:: which is R4's End.X SID for the link towards R5. This was segment 1 of the static SRv6 Policy, and as SR-OS implements the reduced SRH, it is placed in the destination address field of the SRv6 header and not into the SRH itself. The Next Header in the IPv6 header indicates a Routing Header for IPv6 follows, and the type of that Routing Header indicates that it is an SRH. There are three segments in the segment-list and recall that these are encoded starting with the last segment of the path, hence the first element of the segment-list (Address[0]) contains the last segment of the path. The next SID in the SRH that will be visited is Address[2], which is R5's End.X SID for the link towards R2 and was segment 2 of the static SRv6 Policy. Address[1] will follow that, which is R2's End.X SID for the link towards R3 and was segment 3 of the static SRv6 Policy. Finally, Address[0] 2001:db8:1:3:7fff:b000:: is the SRv6 service SID that represents an End.DT4 SID and was advertised by R3 for VPN-IPv4 prefix 172.31.3.0/24. The SRH indicates that the next header is IP in IP, and the IPv4 Echo Request is encapsulated within.

Figure 12-13: IPv4 Payload Encapsulated in SRv6 Policy


```
> Frame 11: 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_96:35:cd (52:54:00:96:35:cd), Dst: RealtekU_e3:17:e0 (52:54:00:e3:17:e0)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
v Internet Protocol Version 6, Src: 2001:db8:1:1::1, Dst: 2001:db8:1:4:7fff:d000::
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0011 0011 1111 1000 0110 = Flow Label: 0x33f86
    Payload Length: 140
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 254
    Source Address: 2001:db8:1:1::1
    Destination Address: 2001:db8:1:4:7fff:d000::
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPIP (4)
      Length: 6
      [Length: 56 bytes]
      Type: Segment Routing (4)
      Segments Left: 3
      Last Entry: 2
      Flags: 0x00
      Tag: 0000
      Address[0]: 2001:db8:1:3:7fff:b000::
      Address[1]: 2001:db8:1:2:0:6000::
      Address[2]: 2001:db8:1:5:0:5000::
> Internet Protocol Version 4, Src: 172.31.1.1, Dst: 172.31.3.1
> Internet Control Message Protocol
```

For completeness, validation is also made of the end-to-end IPv6 data-path through the SRv6 Policy by again initiating an IPv6 ping from CE1 (2001:db8:4001:100::1) to CE3 (2001:db8:4001:500::10).

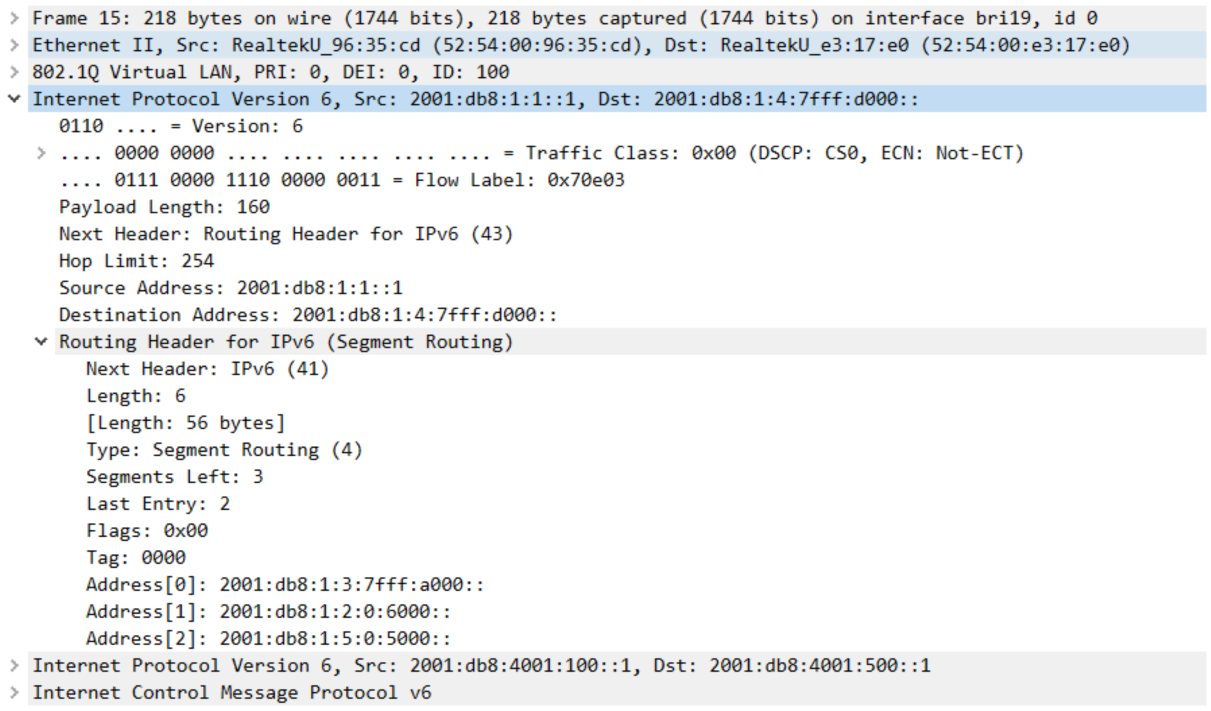
Output 12-37: ICMPv6 Ping from CE1 to CE3 to Validate SRv6 Policy

```
A:ce1# ping router 200 2001:db8:4001:500::1 source 2001:db8:4001:100::1
PING 2001:db8:4001:500::1 56 data bytes
64 bytes from 2001:db8:4001:500::1 icmp_seq=1 hlim=62 time=8.29ms.
64 bytes from 2001:db8:4001:500::1 icmp_seq=2 hlim=62 time=7.99ms.
64 bytes from 2001:db8:4001:500::1 icmp_seq=3 hlim=62 time=7.56ms.
64 bytes from 2001:db8:4001:500::1 icmp_seq=4 hlim=62 time=7.31ms.
64 bytes from 2001:db8:4001:500::1 icmp_seq=5 hlim=62 time=6.94ms.

---- 2001:db8:4001:500::1 PING Statistics ----
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min = 6.94ms, avg = 7.62ms, max = 8.29ms, stddev = 0.478ms
```

Once again, while the ping is in progress, a packet capture is taken on the link between R1 and R4 and this is shown in *Figure 12-14*. The contents of the encapsulating SRH header are largely the same as *Figure 12-13* and have been described above. It is worth noting however that the final segment (Address[0]) in the segment list is a different SRv6 SID to the one used for IPv4. This is because the SID 2001:db8:1:3:7fff:a000 shown here is an End.DT6 SID instantiated at R3 and advertised for the VPN-IPv6 prefix 2001:db8:4001:500::/56. The other notable difference of course is that the SRH indicates that the next header is IPv6, and the ICMPv6 packet is encapsulated within.

Figure 12-14: IPv6 Payload Encapsulated in SRv6 Policy



```
> Frame 15: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_96:35:cd (52:54:00:96:35:cd), Dst: RealtekU_e3:17:e0 (52:54:00:e3:17:e0)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
v Internet Protocol Version 6, Src: 2001:db8:1:1::1, Dst: 2001:db8:1:4:7fff:d000::
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0111 0000 1110 0000 0011 = Flow Label: 0x70e03
    Payload Length: 160
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 254
    Source Address: 2001:db8:1:1::1
    Destination Address: 2001:db8:1:4:7fff:d000::
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPv6 (41)
      Length: 6
      [Length: 56 bytes]
      Type: Segment Routing (4)
      Segments Left: 3
      Last Entry: 2
      Flags: 0x00
      Tag: 0000
      Address[0]: 2001:db8:1:3:7fff:a000::
      Address[1]: 2001:db8:1:2:0:6000::
      Address[2]: 2001:db8:1:5:0:5000::
> Internet Protocol Version 6, Src: 2001:db8:4001:100::1, Dst: 2001:db8:4001:500::1
> Internet Control Message Protocol v6
```

LFA Support for SRv6 Policies

LFA support is provided for SRv6 policies, but the manner in which an LFA repair tunnel is constructed differs slightly to that of SRv6 with shortest path tunnels previously covered in this chapter.

When forwarding shortest path SRv6 packets at an ingress router, the destination SRv6 service SID is copied directly into the IPv6 destination address and no SRH is present under normal forwarding conditions. When an LFA repair tunnel is imposed on the packet, either by the headend or a transit SRv6 router, an SRH is inserted immediately following the IPv6 header. The IPv6 destination address becomes the repair tunnel endpoint, whilst the SRH contains a single segment containing the original SRv6 service SID that was previously the destination address prior to the failure. When the packet reaches the tunnel endpoint, the SRv6 service SID is copied back from the SRH to the destination address and the SRH is removed. An example of this form of LFA backup is shown in *Figure 12-11* and described in surrounding dialogue.

When forwarding an SRv6 traffic engineered path at an ingress router in reduced SRH mode, the IPv6 destination address reflects the first segment of that traffic engineered path, while the remaining segments of the path are carried in the segment list of the SRH. When an LFA repair tunnel is imposed on the packet, either by the headend or a transit SRv6 router, a second SRH is inserted into the packet immediately following the IPv6 header and before the existing SRH. The IPv6 destination address of the packet becomes the repair tunnel endpoint, while the newly inserted SRH contains a single segment containing the IPv6 destination address that was previously the destination address prior to the failure (i.e. the next segment endpoint). When the packet reaches the repair tunnel endpoint, the original destination address is copied back from the LFA SRH to the destination address and the LFA SRH is removed.

To provide an example of an LFA repair tunnel the topology shown in *Figure 12-12* is used together with the SRv6 Policy from R1 to R3 through the path R1-R4-R5-R2-R3. At R1 the SRv6 Policy is learned through BGP and contains three segments shown in *Output 12-38*. The first SID is 2001:db8:1:4:7fff:d000:: which is R4’s End.X SID for the link towards R5. The second SID is 2001:db8:1:5:0:5000:: which is R5’s End.X SID for the link towards R2. The third and final SID is 2001:db8:1:2:0:3000:: which is R2’s End.X SID for the link towards R3.

Output 12-38: R1’s BGP-Learned SRv6 Policy to R3

```
A:admin@R1#   show   router   segment-routing   sr-policies   bgp   color   100   end-point
2001:db8:33ac:2200::3:3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Type            : srv6
Active          : Yes                    Owner           : bgp
Color           : 100
Head            : 0.0.0.0                Endpoint Addr   : 2001:db8:33ac:2200::*
RD              : 23456                  Preference      : 100
SRv6 BSID 1     : 2001:db8:1:1:0:a000::
```

```
TunnelId         : 917507                Age             : 115826
Origin ASN       : 64496                 Origin          : 2001:db8:33ac:2200::*
NumReEval        : 0                     ReEvalReason    : none
NumActPathChange: 0                      Last Change     : 09/20/2022 09:52:41
Maintenance Policy: N/A

Path Segment Lists:
Segment-List     : 1                     Weight          : 1
S-BFD State      : Down                  S-BFD Transitio*: 0
Num Segments     : 3                     Last Change     : 09/20/2022 09:41:26
  Seg 1 SID   : 2001:db8:1:4:7fff:d000::                State : resolved-up
  Seg 2 SID   : 2001:db8:1:5:0:5000::                   State : N/A
  Seg 3 SID   : 2001:db8:1:2:0:3000::                   State : N/A
```

The first SID of 2001:db8:1:4:7fff:d000:: is reachable using R4’s algorithm 0 locator prefix of 2001:db8:1:4::/64, with the shortest path to that prefix being via the R1-R4 link. As LFA is enabled R1 will attempt to compute an LFA backup to that prefix, and the programmed backup path can be seen in the FP tunnel-table shown in *Output 12-39* The primary path uses the link local address of R4 with an egress interface of 1/1/c2/1. It is not possible to compute a basic LFA backup to R4’s locator prefix 2001:db8:1:4::/64, therefore the LFA backup entry, denoted by the `(B)` flag, uses an RLFA repair tunnel to 2001:db8:1:5:0:1000:: which represents the End function of router R5. This backup entry uses a link local address of R2 with an egress interface of 1/1/c1/1.

Output 12-39: FP-Tunnel-Table Entry for R4's Locator Prefix

```
A:admin@R1# show router fp-tunnel-table 1 2001:db8:1:4::/64

===============================================================================
IPv6 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                              Protocol        Tunnel-ID
  Lbl/SID
    NextHop                                              Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
2001:db8:1:4::/64                        SRV6             524305
  -
    fe80::202:ffff:fe00:0-"link-to-R4"                    1/1/c2/1:100
  2001:db8:1:5:0:1000::
    fe80::265:ffff:fe00:0-"link-to-R2"(B)                 1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

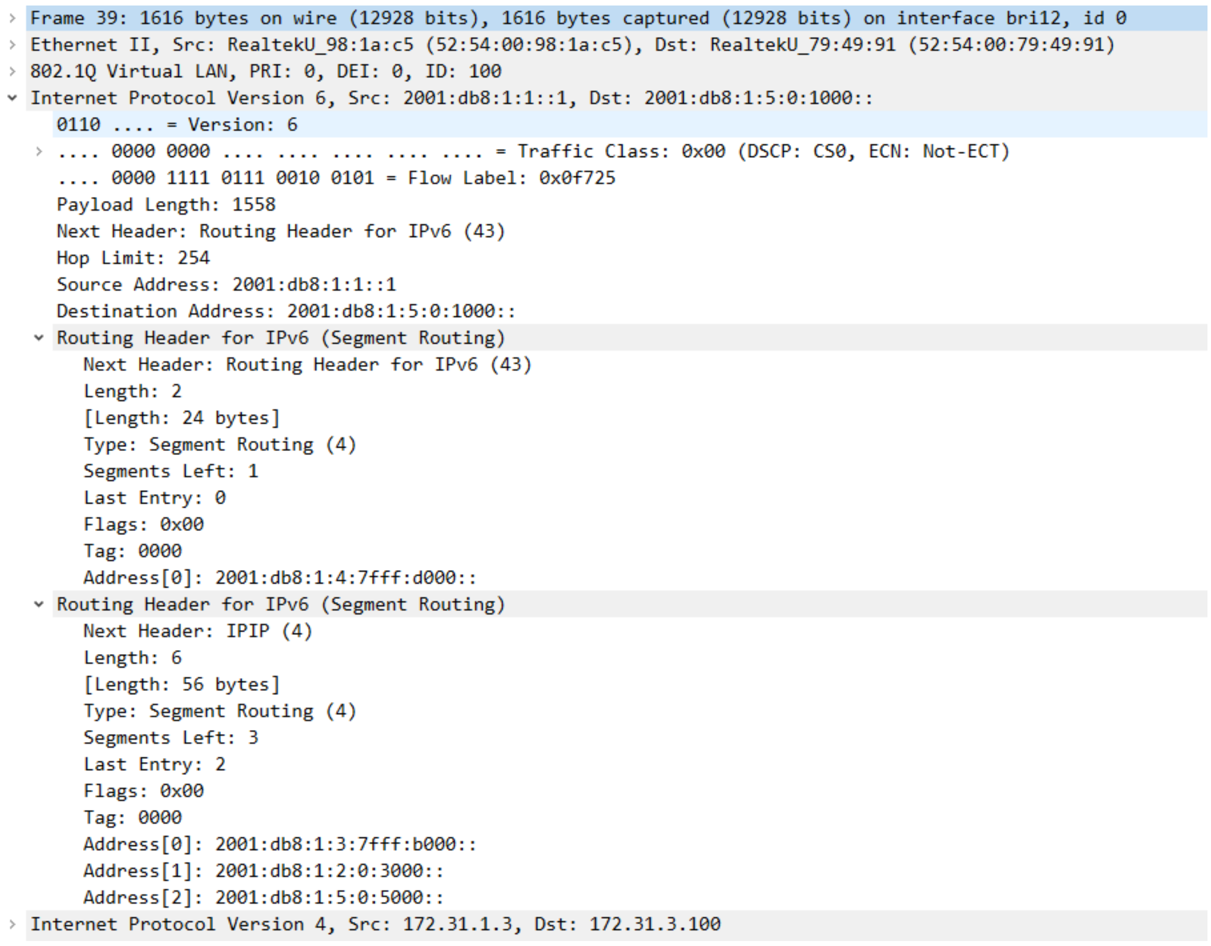
As a VPRN service has previously been extended between R1 and R3 and is configured to use SRv6 policies (*Output 12-34*). Within this VPRN, traffic is offered from CE1 (172.31.1.3) towards CE3 (172.31.3.100) that uses the SRv6 Policy path R1-R4-R5-R2-R3. With traffic being successfully forwarded between source and destination, the link R1-R4 is failed which triggers R1 to switch to the LFA backup. *Figure 12-15* shows the first packet of the traffic towards CE3 captured on the R1-R2 link with the LFA repair path active. The outer IPv6 header has a source address of R1 (2001:db8:1:1::1) and a destination address of R5’s End

Function 2001:db8:1:5:0:1000:: with a Next Header field indicating that an IPv6 Routing Header follows:

- The Routing Header itself is a Type 4 header (SRH) with the Segments Left field set to 1, and a single segment of R4's End.X SID (2001:db8:1:4:7fff:d000::). This SID was segment 1 of the BGP-learned SRv6 Policy shown in *Output 12-38* and would also have been the destination address of the IPv6 header if the packet had been forwarded along its primary path by R1. The SRH also has a Next Header field indicating that an IPv6 Routing Header follows.
- The second Routing Header is also a Type 4 header (SRH) with the Segments Left field set to 3. Segments 2 and 1 (shown as `Address[2]` and `Address[1]`) show segments 2 and 3 of the BGP-learned SRv6 Policy, recalling that the Segment List is encoded starting with the last segment of the path (i.e. the first element of the Segment List ([0]) contains the last segment of the path). Segment 2 represents R5's End.X SID for the link towards R2, while segment 1 represents R2's End.X SID for the link towards R3. The final segment, segment 0, contains the SRv6 service SID that R3 advertised for the VPN-IPv4 prefix 172.31.3.0/24. The Next Header field of this second SRH indicates that the following packet is IP-in-IP.

The encapsulated packet that follows is IPv4 with a source address of 172.31.1.3 and destination address of 172.31.3.100 representing the traffic from CE1 towards CE3.

Figure 12-15: LFA Repair Path with SRv6-TE



```
> Frame 39: 1616 bytes on wire (12928 bits), 1616 bytes captured (12928 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_98:1a:c5 (52:54:00:98:1a:c5), Dst: RealtekU_79:49:91 (52:54:00:79:49:91)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
v Internet Protocol Version 6, Src: 2001:db8:1:1::1, Dst: 2001:db8:1:5:0:1000::
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 1111 0111 0010 0101 = Flow Label: 0x0f725
    Payload Length: 1558
    Next Header: Routing Header for IPv6 (43)
    Hop Limit: 254
    Source Address: 2001:db8:1:1::1
    Destination Address: 2001:db8:1:5:0:1000::
  v Routing Header for IPv6 (Segment Routing)
      Next Header: Routing Header for IPv6 (43)
      Length: 2
      [Length: 24 bytes]
      Type: Segment Routing (4)
      Segments Left: 1
      Last Entry: 0
      Flags: 0x00
      Tag: 0000
      Address[0]: 2001:db8:1:4:7fff:d000::
  v Routing Header for IPv6 (Segment Routing)
      Next Header: IPIP (4)
      Length: 6
      [Length: 56 bytes]
      Type: Segment Routing (4)
      Segments Left: 3
      Last Entry: 2
      Flags: 0x00
      Tag: 0000
      Address[0]: 2001:db8:1:3:7fff:b000::
      Address[1]: 2001:db8:1:2:0:3000::
      Address[2]: 2001:db8:1:5:0:5000::
> Internet Protocol Version 4, Src: 172.31.1.3, Dst: 172.31.3.100
```

When this packet reaches the repair tunnel endpoint at R5, it will copy the remaining segment of the first SRH (2001:db8:1:4:7fff:d000) into the IPv6 header destination address, and after decrementing the Segments Left field to 0 completely remove this first SRH. R5 will then do a longest prefix lookup and forward the packet on the shortest path towards this destination.

To summarise the difference between LFA repair paths for shortest path SRv6 and SRv6-TE, with shortest path SRv6 the point of local repair (PLR) inserts the first SRH for the repair tunnel, whereas with SRv6-TE the PLR inserts a second SRH between the IPv6 header and the original SRH.

Compressed SRv6 SID (C-SID)

While SRv6 has been deployed in operator networks, there is an industry-wide acceptance that there are use-cases, such as strict path TE that require the imposition of long SRv6 SID lists in the SRH. The presence of multiple SIDs in the SRH is viewed as problematic for multiple reasons. The processing of long extension headers is computationally expensive, and as a result SRv6 is not ASIC friendly – a routing header with just three SIDs is 56 bytes long, and another 16 bytes per additional SID. SRv6 can also impose unreasonable bandwidth overhead. Short packets of ~500-bytes or less are common on the Internet, and this requires in excess of 10% additional routing header overhead.

As the overhead associated with native SRv6 became more widely accepted as an issue, several mechanisms were developed (and even deployed) to compress the SRv6 SID list, including the Compressed SID (C-SID), the Compact Routing Header (CRH), the Variable Length SID (VSID), and the Unified Identifier in SR (UIDSR). To unify vendors and operators and move forward with a single standardised mechanism, the SPRING working group commissioned a design team consisting of one person from all the major vendors and one or two operators. The design team produced two informational documents, a requirements document [59] and an analysis document [60]. The requirements document lists the criteria that the selected mechanism should resolve, such as forwarding efficiency, scale, SID summarisation, and address planning. The analysis document essentially analysed and compared each of the proposed SID compression mechanisms against the requirements but didn't explicitly specify the proposed way forward as this was not the intention of the document.

Both documents were submitted to the SPRING working group, and after a poll of the working group members the general consensus was to move ahead with C-SID [61], which is a short encoding of a SID in an SRv6 packet that does not include the SID Locator block. However, before the C-SID specification could be adopted as a working group document, its relationship to the IPv6 addressing architecture specified in [64] caused sufficient concern as to require clarification. To that end the 6MAN working group were tasked with writing a document to clarify and categorise that relationship and explain any deviations. That clarification was subsequently delivered in [71], clearing the way for the C-SID specification to be adopted as a working group document. However, [71] also pointed towards a few issues that the SPRING working group should address before any form of C-SID could move towards standards track. It also recommends the use of an IANA-assigned /16 address block to signal that the addresses within that space are not intended to comply with [64]. Since SRv6 is intended to operate within a contained SRv6 domain (or between collaborating domains), this dedicated /64 would simplify the identification and filtering of packets at the edge of those SRv6 domains.

While the general consensus was to move ahead with C-SID, it was clear that the working group members wished to standardise on one data plane solution only. C-SID unifies work from two previous individual contributions, the Micro-Segment or uSID [62], and the Generalised SID [63] and refers to them as different *flavours*. The uSID is referred to as the NEXT-CSID flavour, while the Generalised SID or G-SID is referred to as the REPLACE-CSID flavour. The basic premise of both flavours is the same however: when a sequence of consecutive SIDs in a segment list share a common Locator Block, the segment list can be compressed by avoiding the repetition of the Locator Block and trailing bits with each individual SID.

The C-SID specification was accepted as a working group document in February 2022. Working group adoption is only an acknowledgement that there is sufficient interest in the subject to warrant further work, and there remain number of issues that need to be resolved before a single data-plane solution can be agreed and standardised. Whilst this process continued, and still continues, a number of vendor implementations have emerged, including Nokia with an implementation of uSID. For completeness, a brief overview of the G-SID is given at the end of this section, but the dominant remaining part of this section provides an overview of the uSID and how to implement and configure it in SR-OS.

The uSID (NEXT-CSID Flavour)

The SRv6 '*micro segment*' (or uSID) instruction is an extension of the SRv6 network programming model in that both the SRv6 control plane and data plane can be leveraged without any change. In addition, it can be used in conjunction with generic SRv6 in that any SID in the SID list can carry micro segments.

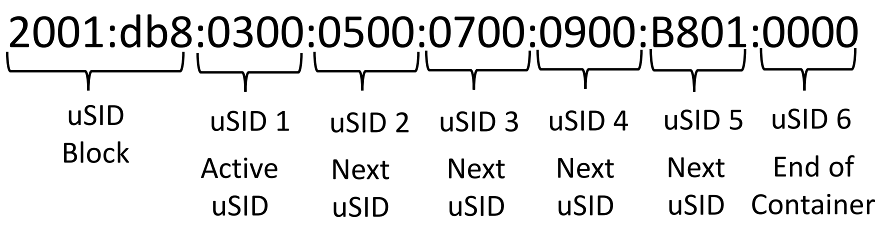
Introduction

SRv6 uSID extends the use of the SRv6 SRH to encode a series of uSID's. It is optimised for use with 16-bit uSIDs, but 32-bit uSIDs can also be used. A uSID container is a 128-bit SRv6 SID that can be encoded in the destination address of an IPv6 header or at any position in the segment list of an SRH. It takes the following format:

<uSID-Block><Active-uSID><Next-uSID>….<Last-uSID><End-of-Container>…<End-of-Container>

A uSID block (Locator Block) can be any IPv6 prefix allocated to the operator. A /32 uSID-Block can support encoding of up to six 16-bit uSIDs as shown in *Figure 12-16*. The <End-of-Container> is a reserved ID (0000) used to mark the end of a uSID container. All of the empty uSID carrier positions must be filled with the End-of-Container ID hence it can be present more than once in a uSID container.

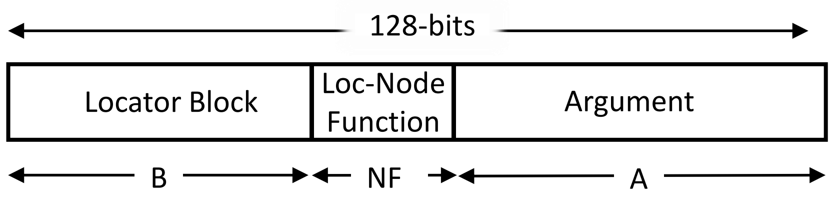
Figure 12-16: uSID Container



A given uSID block is sub-divided to allocate blocks to both global uSIDs and local uSIDs. The set of uSIDs available for global and local uSID allocation are referred to as the global information base (GIB) and local information base (LIB) respectively. A global uSID identifies the shortest path route to a node in the SR domain. An IP route (for example /64) is advertised by the parent node to each of its global uSIDs (analogous to an SRv6 locator), and the parent node executes a variant of the END behaviour. A node may have multiple global uSIDs allocated to it, for example to support Flex-Algorithm. Equally, multiple nodes may share the same global uSID to support anycast. A local uSID identifies an instruction that is specific to its parent, such as a cross-connect to a specific interface and is used to identify local instructions such as adjacency-SIDs and services. If R1 and R2 are different nodes of the uSID domain and L is a local uSID value, then R1 and R2 may bind different behaviours to L. No IP route is advertised by a parent node for its local uSID.

Native SRv6 SIDs consist of a Locator, a Function, and an Argument, that allows an SRv6 router to determine what behaviour, if any other than native IPv6 forwarding, it needs to execute for a given SID. That structure does not exist for uSIDs because the Locator and Function are combined in a single field as shown in *Figure 12-17*, while the argument is used to carry the remaining C-SIDs in the current C-SID container.

Figure 12-17: NEXT-C-SID Structure



As a result of this combined Locator and Function field, a given uSID needs to be identifiable as belonging to either the GIB or the LIB to allow a given node to execute the correct behaviour. This is achieved by splitting the available hex range of a uSID into non-overlapping GIB and LIB spaces such that GIB and LIB are identifiable as belonging to an allocated range. As an example, assume the following:

- 48-bit uSID block length of 2001:db8:0::/48.
- 16-bit uSID length.

- A uSID 2001:db8:0:*ABCD*::/64 is said to be allocated from its parent block 2001:db8:0::/48.The uSID *ABCD* is referred to as the Locator-Node.

The Locator-Node is a 16-bit uSID containing a hex range of 0x0001-FFFF, hence there are 65,534 possible non-zero decimal values. If hex range 0x0001-DFFF is allocated to GIB, and the remaining hex range E000-FFFF to LIB, then such a scheme the uSID block 2001:db8:0::/48 would support up to 57,343 global uSIDs (routers) and each router would support up to 8,191 local uSIDs (57,343 + 8,191 = 65,534). In this scheme, the uSID D123 belongs to GIB, whereas the uSID E123 belongs to LIB. The hex range can be split to suit operator requirements. It might be desirable to have a larger LIB than 8k, but it follows that less hex values are subsequently made available to GIB. For example, if hex range 0x0001-4FFF is allocated to GIB and the remaining hex range 5000-FFFF is allocated to LIB then this scheme would support up to 20,479 global uSIDs and each router would support up to 45,055 local uSIDs (20,479 + 45,055 = 65,534).

The SRv6 network programming model is extended to define a number of uSID-specific endpoint behaviours, which are described as follows assuming a 48-bit uSID block of 2001:db8:0::/48 and a 16-bit uSID length:

- uN is the End behaviour, with NEXT-CSID, PSP, and USD flavours. The prefix 2001:db8:0:0N00::/64 is bound to the End behaviour with NEXT-CSID and implements a shift-and-lookup (where 0N00 is a nonsensical example SID taken from the GIB). The prefix 2001:db8:0:0N00::/80 is bound to the End behaviour with PSP and USD flavours. The ::/80 can almost be considered a double-lookup of both the Locator Node uSID (0N00) and the presence of a '0000' End of Container uSID following it to verify that it should execute the End behaviour. Like a native SRv6 End SID, the uN uSID is advertised into the IGP in an SRv6 End SID Sub-TLV (*Figure 12-5*) which is carried as a Sub-TLV of the SRv6 Locator TLV. The SRv6 End SID Sub-TLV also carries an SRv6 SID Structure Sub-Sub-TLV to signal the structure of that uSID. In this example its structure is defined as {Locator Block Length (LBL) = 48, Locator Node Length (LNL) = 16, Function = 0, Argument Length (AL) = 64}.
- uA is the End.X behaviour with NEXT-CSID, PSP, and USD flavours. uA implements a shift-and-xconnect behaviour, and an instance of the uA SRv6 uSID behaviour is associated with a set of one or more layer 3 adjacencies. The prefix 2001:db8:0:FNAJ::/64 is bound to the shift-and-xconnect behaviour (where FNAJ is a nonsensical example SID taken from the LIB). The prefix 2001:db8:0:FNAJ::/80 is bound to the End.X behaviour with PSP and USD flavours and in order to provide the same routable semantics as End.X , the uA SID is advertised into the IGP as uN+uA where uN provides the route to the node (like the End behaviour) and uA provides the cross-connect function. The uA uSID is advertised into the IGP in an SRv6 End.X SID Sub-TLV (*Figure 12-6*), which is carried as a Sub-TLV of the Extended IS Reachability TLV. The SRv6 End.X SID Sub-TLV also carries an SRv6 SID Structure Sub-Sub-TLV which in this example is defined as {LBL = 48, LNL = 16, Function 16, AL = 48}.
- uDT is the uSID-specific behaviour for a table lookup in a VRF (which could be the global VRF). The prefix 2001:db8:0:0N00:FNVT::/80 implements the same behaviour as End.DT4 and End.DT6. It is signalled in BGP and uses the SRv6 Service TLV, SRv6

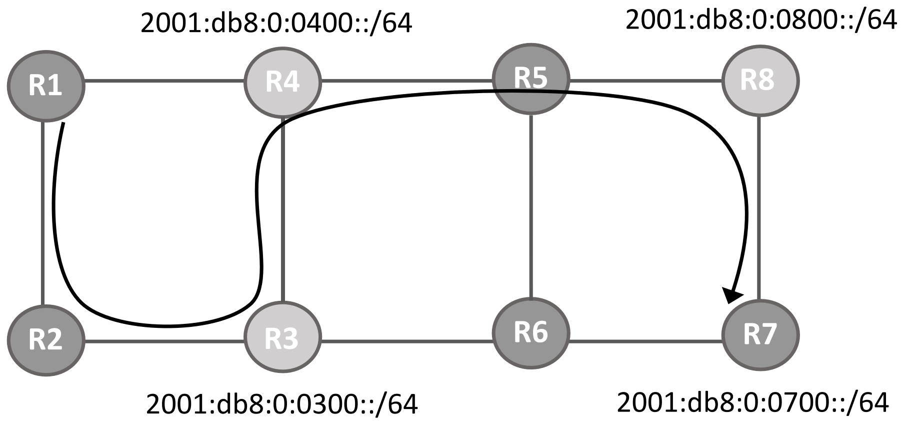
SID Information Sub-TLV, and SRv6 SID Structure Sub-Sub-TLV previously described in the 'Service Overlay' section of this chapter.

- uDX is the USID-specific behaviour for a decapsulation and cross-connect. The prefix 2001:db8:0:0N00:FNXJ::/80 is bound to the same behaviour as End.DX4 (IPv4), End.DX6 (IPv6), and End.DX2 (layer 2). As above, it is signalled in BGP and uses the SRv6 Service TLV, SRv6 SID Information Sub-TLV, and SRv6 SID Structure Sub-Sub-TLV previously described in the 'Service Overlay' section of this chapter.

To illustrate the packet forwarding concept with the NEXT-CSID flavour the schematic in *Figure 12-18* is used. Routers R1 to R8 belong to the same SRv6 domain and a path from R1 to R7 needs to travel through routers R3, R4, and R8. The uSID block is 2001:db8:0::/48, and global uSIDs representing node IDs are advertised as 2001:db8:0:0N00::/64, where 0N00 is 0300 for router 3, 0400 for router 4, 0700, for router 7, and 0800 for router 8. Each node advertises its uSID node ID into the IGP. R1 encapsulates an ingress packet in an outer IPv6 header with a destination address consisting of uSID container 2001:db8:0:0300:0400:0800:0700:8b7d, where:

- 2001:db8:0:0300:0400:0800:0700 encodes a source routed stateless path via router R3, R4, R8, and finally R7.
- 8b7d is uDT6 SID instantiated at router R7.

Figure 12-18: NEXT-CSID uSID Advertisement



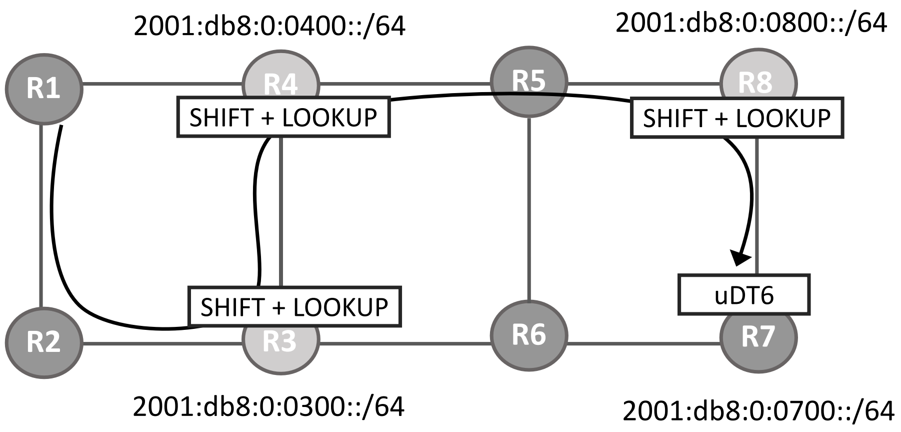
R1 forwards this packet to R2 as R2 is on the shortest path towards prefix 2001:db8:0:0300::/64 advertised by router R3. R2 is a transit router, and so performs a regular IPv6 longest prefix lookup and forwards the packet along the shortest path to R3.

- When R3 receives the packet, it has a match for 2001:db8:0:0300::/64 bound to the uN NEXT-CSID behaviour and therefore executes the shift-and-lookup function. R3 removes its own uSID from the destination address, shifts the remainder of the uSID container left, and inserts a '0000' marker at the tail of the container. The updated destination address becomes 2001:db8:0:0400:0800:0700:8b7d:0000 which R3 forwards using a longest prefix lookup towards R4.
- When R4 receives the packet, it has a match for 2001:db8:0:0400::/64 bound to the uN NEXT-CSID behaviour and therefore executes the shift-and-lookup function. R4

removes its own uSID from the destination address, shifts the remainder of the uSID container left, and inserts a '0000' marker at the tail of the container. The updated destination address becomes 2001:db8:0:0800:0700:8b7d:0000:0000 which R4 forwards on the shortest path towards R8 via R5.

- When R8 receives the packet, it has a match for 2001:db8:0:0800::/64 bound to the uN NEXT-CSID behaviour and again executes the shift-and-lookup function. R8 removes its own uSID from the destination address, shifts the remainder of the uSID container left, and inserts a '0000' marker at the tail of the container. The updated destination address then becomes 2001:db8:0:0700:8b7d:0000:0000:0000. R8 then forwards the packet on the shortest path towards router R7.
- When R7 receives the packet, it removes its global uSID from the destination address and recognises the following uSID as a local uSID bound to the uDT6 behaviour. It therefore performs a route-table lookup in a VRF and forwards the packet towards its destination.

Figure 12-19: NEXT-CSID Packet Forwarding Example



In the above example the next CSID is always extracted from the uSID container in the destination address, but it is equally possible that the next CSID is carried in an SRH. The encoding of uSIDs into the SRH means that traffic engineered paths are not limited to the number of uSIDs that can be carried in the single uSID container of the destination address. However, because the next CSID can be carried in either the uSID container or the SRH there needs to be a means for a uSID router to determine where the next CSID is. When using generic SRv6, the node ID and the End function are two distinct elements and the 128-bit SID is formally structured as {block, node, function}. With uSID that formal structure does not exist, and the node identifier also serves as the End function (the components are merged). As a result, the behaviour that a given uSID router needs to perform depends on the uSID following the active uSID in the container.

As an example, assume a uSID block of 2001:db8:0::/48 and a uSID length of 16-bits. A match on 2001:db8:0:0400::/64 is bound to the uN End behaviour with shift-and-lookup where the next SID is in the uSID container. This is the behaviour illustrated in the forwarding example shown in *Figure 12-19*, and in this case the packet transits through the uSID domain and the destination address is gradually emptied of all of its uSIDs until ultimately only a single uSID remains. However, it may be that the next uSID is not in the uSID container but rather contained is in the SRH, in which case the uSID router needs to detect. The ‘0000’ End of Container ID is used to provide that indication, and as such a match on 2001:db8:0:0400:0000/80 is bound to the uN End behaviour with the next SID contained in the SRH. In this case, the behaviour is exactly the same as SRv6; the SID in the SRH is extracted in its entirety and placed into the destination address of the IPv6 header before onward forwarding. The same behaviour applies to the uA function. A match on 2001:db8:0:0400:f202::/80 is bound to the uA End.X behaviour with shift-and-lookup where the next SID is in the uSID carrier, but a match on 2001:db8:0:0400:f202:0000::/96 is bound to the uA End.X behaviour where the next SID is in the SRH.

Configuration

At the time of writing SRv6 uSID is supported only on FP-based platforms with network interfaces on FP4 ports. It is not currently supported on 7250 IXR platforms of any generation. Additionally, SR-OS does not support SRv6 uSID for OSPFv3 with current support only for IS-IS in both Multi-Topology (MT) 0 (standard) and MT2 (IPv6).

Please check Release Notes for an updated list of 7250 IXR unsupported features.

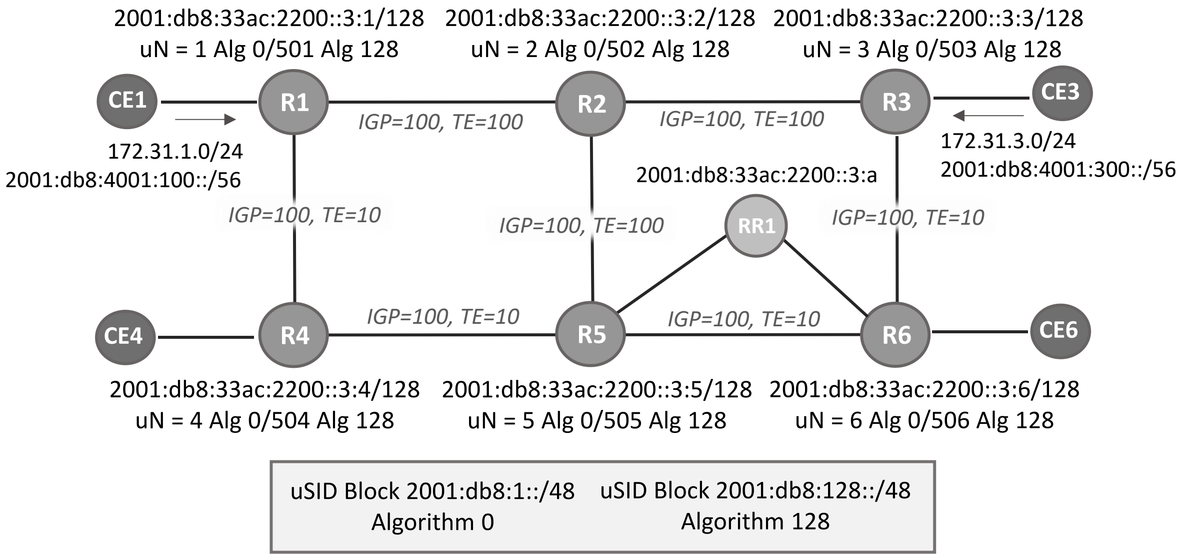
SR-OS currently provides support for shortest path SRv6 uSID only and offers the same capabilities such as service support and LFA as generic SRv6. To illustrate its use the network topology shown in *Figure 12-20* is used. Routers R1 to R6 and a Route-Reflector, RR1, are in a flat IS-IS Level 2 area. All routers belong to Autonomous System 64496 and Routers R1 to R6 peer in IBGP with RR1 as Route-Reflector clients for advertisement of service routes. The network is dual-stack IPv4/IPv6 and all BGP peering uses IPv6 addressing using system addresses shown. IPv6 BGP peering is used in this example simply because it’s considered a more likely scenario in an SRv6 network. It is not a pre-requisite for SRv6 or uSID because the BGP next-hop is not used to resolve BGP prefixes to SRv6 tunnels, but rather a longest-prefix lookup is made on the advertised service SID, which will resolve to an advertised uSID Locator. All IGP link metrics are 100. Unidirectional link delay is also configured, and all links have a delay of 10 milliseconds with the exception of links R1-R2, R2-R3, and R2-R5, which have a delay of 100 milliseconds. Flex-Algorithm is enabled and algorithm identifier 128 is configured to use latency as a metric. Two /48 uSID blocks are used. Prefix 2001:db8:1::/48 is assigned to the base algorithm (algorithm 0) while prefix 2001:db8:128::/48 is assigned to algorithm 128. Assignment of uN endpoints is as follows:

- Node numbers for routers R1-R6 are used for algorithm 0. For example, R1’s assigned uN for algorithm 0 is 1, R2’s assigned uN is 2, and R6’s assigned uN is 6. Hence, R1’s GIB entry is 2001:db8:1:1::/64, R2’s GIB entry is 2001:db8:1:2::/64, and R6’s GIB entry is 2001:db8:1:6::/64.
- Node numbers for routers R1-R6 are prefixed with the number 128 for algorithm 128. For example, R1’s assigned uN for algorithm 128 is 1281 (hex 501), R2’s assigned uN is 1282 (hex 502), and R6’s assigned uN is 1286 (hex 506). Hence, R1’s

GIB entry is 2001:db8:128:501::/64, R2's GIB entry is 2001:db8:128:502::/64, and R6's GIB entry is 2001:db8:128:506::/64.

A number of CE routers are attached to the topology that advertise IPv4 and IPv6 prefixes to their adjacent routers which will be used to verify that SRv6 is functioning as anticipated.

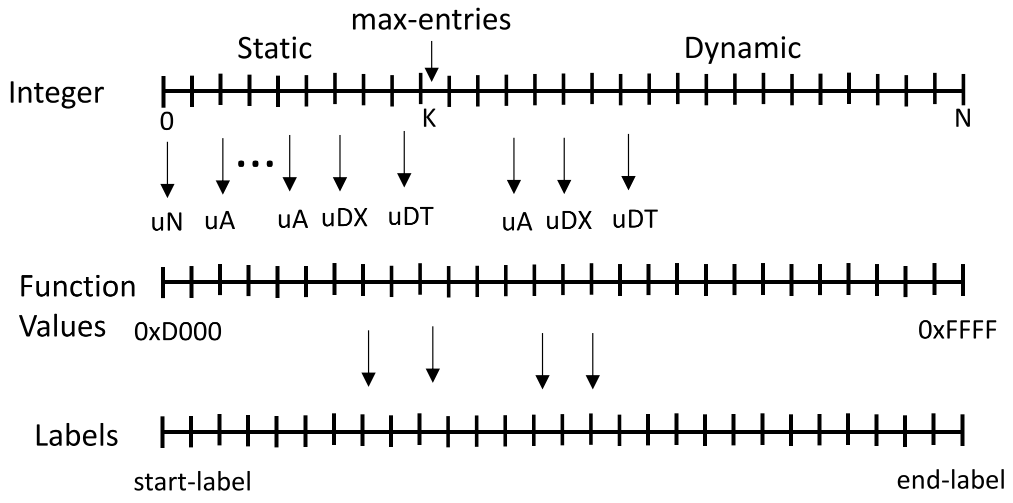
Figure 12-20: Test Topology for SRv6 uSID



As with generic SRv6, there is a requirement for a Forwarding Path Extension (FPE) for service origination and termination when using SRv6 uSID. The configuration for FPE have been previously described within this chapter and are therefore not repeated here.

When using SRv6 uSID with 16-bit uSID length (not 32-bit length), the labels of all static and dynamic service functions need to be drawn from a reserved label block. When configuring endpoint functions in the static range, an integer value is manually selected by the user which is bound by the max-entries value. Equally, in the dynamic range the system will automatically select an integer value above the max-entries value. The system will then derive a function value using a one-to-one mapping, which is encoded in the data-plane and also distributed in the control plane (IS-IS or BGP). Finally, the system will select a label from the reserved label block, once again in a one-to-one mapping. This does of course mean a small number of labels will be wasted because the system does not require a label for a uN endpoint behaviour, but this is a negligible waste, and the use of a reserved label-block cannot be avoided. The relationship between integer, function value, and labels is depicted in *Figure 12-21.*

Figure 12-21: Reserved Label Block and Label Allocation



Output 12-40 shows the reserved label blocks that will be used for SRv6 uSID algorithm 0 and algorithm 128, with both blocks consisting of 8,192 entries each. Recall that a uSID is identified as belonging to either the GIB or the LIB. As the LIB is used for support of local uSIDs and labels are allocated on a one-to-one basis with function values as described above, it follows that the number of labels allocated to the corresponding label-block should be the same as the size of the LIB. If the label-block is smaller than the configured LIB it will not be blocked by CLI, but it may prevent the system from allocating a label, which in turn will prevent the advertisement of prefixes associated with that uSID.

Output 12-40: Reserved Label Blocks for uSID Algorithm 0 and 128

```
    router "Base" {
        mpls-labels {
             reserved-label-block "SRv6-uSID-Alg0" {
                start-label 30000
                end-label 38191
            }
            reserved-label-block "SRv6-uSID-Alg128" {
                start-label 38192
                end-label 46383
            }
        }
    }
```

The process of enabling SRv6 uSID is similar in many ways to that of generic SRv6, however, there are some notable differences both in nomenclature and structure. The first step is to configure the global uSID settings and these are contained within the **segment-routing-v6 micro-segment** context as shown in *Output 12-41*. The **block-length** refers to the length of uSID block or blocks in use, which must all be of the same length network-wide. The range is 8-64 in steps of 8, with a default setting of 32. As the example network topology uses 2001:db8:1::/48 and 2001:db8:128::/48, the **block-length** is set to 48. The **sid-length** determines the size of the uSID and can currently only be set to a value of 16. The default value is therefore 16, but it is shown here for completeness (a uSID length of 32 will become available in a future release). The **global-sid-entries** command defines the size of the GIB, and as a consequence, the size of the LIB. The configurable range is 4-60 in steps of 4, and each value is a multiple of 1024. The possible values and resulting size of GIB and LIB are shown in *Table 12-3*. The number of **global-sid-entries** in this example is set to 56, hence this configuration will support up to 57,343 global uSIDs (routers) while each router will support up to 8,191 local uSIDs. Note that the 8,191 possible local uSIDs also reconciles with the size of the label blocks configured in *Output 12-40*.

Under the **segment-routing-v6** context, the **origination-fpe** and **source-address** are shown. Although their use has previously been described for generic SRv6, they are also necessary for SRv6 uSID operation and hence are shown again here. The **origination-fpe** command configures a single originating FPE for all SRv6-based services, and in this example FPE 1 is used. The **source-address** is used as the source address of data packets originated for SRv6-based services, and if necessary, this can be overridden at service level using the **segment-routing-v6 source-address** command.

Output 12-41: Micro-Segment Global Settings

```
        segment-routing {
            segment-routing-v6 {
                origination-fpe [1]
                source-address 2001:db8:1:1::1
                micro-segment {
                    block-length 48
                    global-sid-entries 56
                    sid-length 16
                }
            }
        }
```

Table 12-3: Possible GIB/LIB Sizes with 16-bit uSID

SID multiplier	Start of GIB (hex)	End of GIB (hex)	GIB Count	Start of LIB (hex)	End of LIB (hex)	LIB Count
16	0001	3FFF	16383	4000	FFFF	49151
20	0001	4FFF	20479	5000	FFFF	45055
24	0001	5FFF	24575	6000	FFFF	40959
28	0001	6FFF	28671	7000	FFFF	36863
32	0001	7FFF	32767	8000	FFFF	32767
36	0001	8FFF	36863	9000	FFFF	28671
40	0001	9FFF	40959	A000	FFFF	24575
44	0001	AFFF	45055	B000	FFFF	20479
48	0001	BFFF	49151	C000	FFFF	16383
52	0001	CFFF	53247	D000	FFFF	12287
56	0001	DFFF	57343	E000	FFFF	8191
60	0001	EFFF	61439	F000	FFFF	4095

The next step is to configure the uSID blocks. *Output 12-42* shows the configuration at R1, with two blocks created within the **segment-routing-v6 micro-segment** context. One **block** is intended for functions belonging to Flex-Algorithm 0, while the second **block** is intended for use with functions belonging to Flex-Algorithm 128. Within each block, the **termination-fpe** specifies the FPE instance that will be used for SRv6 uSID service termination. Unlike the **origination-fpe**, the **termination-fpe** is configurable on a per-block basis, and both blocks are configured to use FPE 2. The **label-block** refers to the previously configured **reserved-label-blocks**, and each block references its own label-block of 8,192 entries. The **prefix** sub-context allows for the configuration of the uSID block prefix using the **ip-prefix** command, which in this example is 2001:db8:1::/48 for algorithm 0 and 2001:db8:128::/48 for algorithm 128. The **static-function** sub-context contains a **max-entries** command that allows for the configuration of an upper limit on the maximum number function values that must be reserved for static uN, uA, and service SID function assignment. It has a default value of 1, and if more than one static uN, uA, or service SID is used, this value must be increased accordingly. Finally, both blocks are put into an **admin-state** of **enable**.

Output 12-42: uSID Block Configuration

```
        segment-routing {
            segment-routing-v6 {
                micro-segment {
                    block "SRv6-uSID-Alg0" {
                        admin-state enable
                        termination-fpe [2]
                        label-block "SRv6-uSID-Alg0"
                        prefix {
                            ip-prefix 2001:db8:1::/48
                        }
                        static-function {
                            max-entries 16
                        }
                    }
                    block "SRv6-uSID-Alg128" {
                        admin-state enable
                        termination-fpe [2]
                        label-block "SRv6-uSID-Alg128"
                        prefix {
                            ip-prefix 2001:db8:128::/48
                        }
                        static-function {
                            max-entries 16
                        }
                    }
                }
            }
        }
```

Once the uSID blocks have been defined, the uSID Locators can be created and base router endpoint behaviours can be defined. The uSID Locators are configured using the **micro-segment-locator** command under the **segment-routing-v6** context. Two uSID Locators are created. The first “uSID-Alg0-Locator” uses the **block** command to reference the previously configured uSID block for algorithm 0 and is assigned a static **un value** of 1. Similarly the second “uSID-Alg128-Locator” uses the **block** command to reference the previously configured uSID block for algorithm 128 and is assigned a static **un value** of 1281 (hex 501). The second uSID locator also uses the **algorithm** command to associate itself with algorithm

128 and this algorithm value will be advertised with the locator TLV. The default is the base algorithm 0, hence the algorithm command is not seen in the configuration of the first micro-locator. Both **micro-segment-locators** are then put into an **admin-state** of **enable**.

The **base-routing-instance** context provides the ability to configure the endpoint behaviour for uA and uDT4/uDT6 service SIDs for IPv4/IPv6 prefixes within the global routing table. In this example the **micro-segment-locator** command is used to reference the uSID locator configured to use algorithm 0, although it is possible to use any configured uSID locator. Within the following **function** context, the **ua-auto-allocate** command dynamically allocates a function value from the range [{max-entries+1}..{2^{20}-1}]. It is followed by the SRH mode which like generic SRv6 can be **usp** or **psp**, and a **protection** command that specifies whether or not the link is eligible for TI-LFA protection (which in turn determines if the B-flag set or unset in the advertised SID). Like generic SRv6, if a **protection** type of **protected** is selected, TI-LFA must be enabled in order to generate End.X/uA SIDs. Static uA functions are configured using the **ua** command followed by an integer value in the range 1-1048575. If multiple **ua** instances are required (which is likely to be the case if one is required for each adjacency), the **static-function max-entries** command within the relevant **micro-segment block** must be increased from the default value of 1. Each **ua** context provides for definition of the **srh-mode**, the **protection** mode, and an **interface-name** to which the function will be assigned. Statically configured uA functions are persistent in nature. Dynamically allocated entries are not persistent.

Output 12-43: Configuration of Micro-Segment-Locators

```
        segment-routing {
            segment-routing-v6 {
                micro-segment-locator "uSID-Alg0-Locator" {
                    admin-state enable
                    block "SRv6-uSID-Alg0"
                    un {
                        value 1
                    }
                }
                micro-segment-locator "uSID-Alg128-Locator" {
                    admin-state enable
                    algorithm 128
                    block "SRv6-uSID-Alg128"
                    un {
                        value 1281
                    }
                }
                base-routing-instance {
                    micro-segment-locator "uSID-Alg0-Locator" {
                        function {
                            ua-auto-allocate psp protection protected { }
                        }
                    }
                }
            }
        }
```

Output 12-44 shows the local uSIDs instantiated at router R1 as a result of applying the above configuration. There are two uSIDs with uN behaviour with function values of 1 and 1281 representing the uN endpoints for algorithm 0 and algorithm 128 respectively. These function values are taken from the GIB as they must be globally routable. With a 48-bit uSID length, the uSID locator prefix for algorithm 0 is shown as 2001:db8:1:1:: and the uSID locator prefix for algorithm 128 is shown as 2001:db8:128:501::. There are also two uA endpoints, one each to represent R1's network links to R2 and R4. Both of these entries are taken from R1's LIB and are allocated values 0xe012 and 0xe013 with function values 57362 and 57363 respectively.

Output 12-44: Router R1's Local-uSIDs

```
A:admin@R1# show router segment-routing-v6 micro-segment-local-sid

===============================================================================
Micro Segment Routing v6 Local SIDs
===============================================================================
SID                                                 Type           Function
  Micro Segment Locator
  Context
-------------------------------------------------------------------------------
2001:db8:1:1::                                      uN             1
  uSID-Alg0-Locator
    None
2001:db8:1:1:e012::                                 uA             57362
  uSID-Alg0-Locator
    None
2001:db8:1:1:e013::                                 uA             57363
  uSID-Alg0-Locator
    None
2001:db8:128:501::                                  uN             1281
  uSID-Alg128-Locator
    None
-------------------------------------------------------------------------------
SIDs : 4
-------------------------------------------------------------------------------
```

Once the locators and uSID functions are configured they need to be advertised into IS-IS for reachability. Like generic SRv6 Locators, micro-SID Locators can be enabled in single or multiple IS-IS instances, and each Locator can be enabled in either MT0 (standard topology) or MT2 (IPv6) of each of those instances. By default, Locators are added in the standard topology, MT0. Export policies can be used to redistribute remote locators between MT0 and MT2 topologies of different IS-IS instances.

Within the **segment-routing-v6** context of the relevant **isis** instance, each **micro-segment-locator** is configured and assigned a **level-capability** together with an optional **metric** that is advertised with the micro-locator.

Output 12-45: Advertising uSID Locators into IS-IS

```
    router "Base" {
        isis 0 {
            segment-routing-v6 {
                admin-state enable
                micro-segment-locator "uSID-Alg0-Locator" {
                    level-capability 2
                    level 2 {
                        metric 1
                    }
                }
                micro-segment-locator "uSID-Alg128-Locator" {
                    level-capability 2
                    level 2 {
                        metric 1
                    }
                }
            }
        }
    }
```

Output 12-46 shows router R1's advertised IS-IS L2 LSP with SRv6 uSID enabled, truncated to show only the parts relevant to SRv6 uSID. A key point is that SRv6 uSID makes no changes to the control plane and purely leverages the extensions already implemented for generic SRv6. The Router Capability TLV carries the SRv6 Capabilities sub-TLV and the Node MSD Advertisement sub-TLV contains the SRv6 MSDs previously described in *Table 12-2* for SRv6. The Extended Reachability TLV shows an End.X SID sub-TLV for the adjacency to R2 with its associated uSID value, Backup-flag (B) set, and algorithm set to 0. The endpoint behaviour differs from generic SRv6 and contains the uSID-specific endpoint End.X with NEXT-CSID and PSP as defined in [62]. The End.X SID sub-TLV also carries an SRv6 SID Structure sub-sub-TLV advertising the Locator block length, Locator node length, Function length, and Argument length. The Locator TLV contains the two uSID Locator prefixes of 2001:db8:1::1/64 and 2001:db8:128:501::/64 for algorithm 0 and 128 respectively. Both prefixes have an endpoint behaviour of End-NXT-CSID-PSP. Like the End.X SID sub-TLV, the Locator TLV contains an SRv6 SID Structure sub-TLV advertising the Locator block length, Locator node length, Function length, and Argument length. Not shown in the output, and like generic SRv6, the as prefix 2001:db8:1:1::/64 belongs to algorithm 0, it is also advertised in an IPv6 Prefix Reachability TLV so that SRv6-incapable routers can install a forwarding entry for algorithm 0 SRv6 uSID traffic.

Output 12-46: R1's IS-IS LSP with SRv6 uSID Enabled

```
A:admin@R1# show router isis database R1.00-00 level 2 detail

 --- [snip] ---
  Router Cap : 192.0.2.1, D:0, S:0
    Node MSD Cap: BMI : 12 ERLD : 15 SRH-MAX-SL : 10 SRH-MAX-END-POP : 9 SRH-MAX-H-
ENCAPS : 1 SRH-MAX-END-D : 9
    SRv6 Cap: 0x0000
 --- [snip] ---
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 193
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
 --- [snip] ---
    End.X-SID: 2001:db8:1:1:e012:: flags:B algo:0 weight:0 endpoint:End.X-NXT-CSID-PSP
lbLen:48 lnLen:16 funLen:16 argLen:48
 --- [snip] ---
  SRv6 Locator  :
    MT ID : 0
    Metric: ( ) 1 Algo:0
    Prefix   : 2001:db8:1:1::/64
    Sub TLV   :
      AttrFlags: N
      End-SID   : 2001:db8:1:1::, flags:0x0, endpoint:End-NXT-CSID-PSP, lbLen:48,
lnLen:16, funLen:0, argLen:64
    Metric: ( ) 1 Algo:128
    Prefix   : 2001:db8:128:501::/64
    Sub TLV   :
      AttrFlags: N
      End-SID   : 2001:db8:128:501::, flags:0x0, endpoint:End-NXT-CSID-PSP, lbLen:48,
lnLen:16, funLen:0, argLen:64
```

All routers in the test topology advertise a uSID Locator prefix and each router installs resolved remote uSID Locator prefixes in the route-table, FIB, and tunnel-table. *Output 12-47* shows R1's route-table entry for R3's algorithm 128 uSID Locator prefix 2001:db8:128:503::/64. It is programmed as a direct tunnel next-hop over SRv6 IS-IS and has a metric of 40001 representing the shortest path latency path via R1-R4-R5-R6-R3 with the addition of the metric 1 that was advertised with the Locator TLV.

Output 12-47: R1's Route-Table Entry for R3's Algorithm 128 uSID Locator Prefix

```
A:admin@R1# show router route-table ipv6 2001:db8:128:503::/64

===============================================================================
IPv6 Route Table (Router: Base)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                   Metric
-------------------------------------------------------------------------------
2001:db8:128:503::/64                         Remote  ISIS      01d19h26m  18
       2001:db8:128:503::/64 (tunneled:SRV6-ISIS)                  40001
```

An entry for the uSID Locator prefix is also added to the tunnel-table with a protocol owner of `srv6-isis`. The next-hop address is the IPv6 link local address of the neighbour R4, and the `[L]` flag indicates that an LFA backup is programmed. The preference for the entry is 0 simply because there is no requirement to have a comparative preference metric as it is the only tunnel-table protocol that can be used to forward SRv6 traffic. The tunnel ID shown in the output is 524324 and can be reconciled with the FIB output shown in *Output 12-49*.

Output 12-48: R1's Tunnel-Table Entry for R3's Algorithm 128 uSID Locator Prefix

```
A:admin@R1# show router tunnel-table ipv6 2001:db8:128:503::/64

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                     Owner     Encap TunnelId  Pref
Nexthop                                         Color           Metric
-------------------------------------------------------------------------------
2001:db8:128:503::/64 [L]                       srv6-isis SRV6  524324    0
  fe80::202:ffff:fe00:0-"link-to-R4"                              40001
```

Output 12-49: R1's FIB Entry for R3's Algorithm 128 uSID Locator

```
A:admin@R1# show router fib 1 ipv6 ip-prefix-prefix-length 2001:db8:128:503::/64

===============================================================================
FIB Display
===============================================================================
Prefix [Flags]                                              Protocol
  NextHop
-------------------------------------------------------------------------------
2001:db8:128:503::/64                                       ISIS
  2001:db8:128:503::/64 (Transport:SRV6-ISIS:524324)
-------------------------------------------------------------------------------
Total Entries : 1
```

As with generic SRv6, LFA support is provided for SRv6 uSID for the base routing instance and any Flex-Algorithm instances and is enabled for the former as described in chapter 9 and the latter as described in chapter 10.

Output 12-50 shows R1’s alternative route-table entry for R3’s algorithm 128 uSID Locator prefix 2001:db8:128:503::/64. The primary next-hop is the link local address of R4, while the backup next-hop is indicated with the `(LFA)` flag and is the link local address of R2.

Output 12-50: R1's Alternative Path to R3's Algorithm 128 uSID Locator Prefix

```
A:admin@R1# show router tunnel-table ipv6 2001:db8:128:503::/64 alternative

===============================================================================
IPv6 Tunnel Table (Router: Base)
===============================================================================
Destination                                  Owner     Encap TunnelId  Pref
Nexthop                                      Color           Metric
-------------------------------------------------------------------------------
2001:db8:128:503::/64                        srv6-isis SRV6  524324    0
  fe80::202:ffff:fe00:0-"link-to-R4"                           40001
2001:db8:128:503::/64 (LFA)                  srv6-isis SRV6  524324    0
  fe80::265:ffff:fe00:0-"link-to-R2"                            -
```

The programmed backup path can also be seen in the FP tunnel-table shown in *Output 12-51*. For uSID Locator Prefix 2001:db8:128:503::/64 which uses the algorithm 128 delay metric, R1 is able to compute a basic LFA. The primary path uses the link local address of R4 with an egress interface of 1/1/c1/1 while the backup path, denoted by the `(B)` flag uses the link local address of R2 with an egress interface of 1/1/c1/1.

Output 12-51: R1's FP-Tunnel-Table Entry for R3's Algorithm 128 uSID Locator Prefix

```
A:admin@R1# show router fp-tunnel-table 1 2001:db8:128:503::/64

===============================================================================
IPv6 Tunnel Table Display

Legend:
label stack is ordered from bottom-most to top-most
B - FRR Backup
===============================================================================
Destination                               Protocol         Tunnel-ID
  Lbl/SID
    NextHop                                                Intf/Tunnel
  Lbl/SID (backup)
    NextHop   (backup)
-------------------------------------------------------------------------------
2001:db8:128:503::/64                     SRV6              524324
  -
    fe80::202:ffff:fe00:0-"link-to-R4"                      1/1/c2/1:100
  -
    fe80::265:ffff:fe00:0-"link-to-R2"(B)                   1/1/c1/1:100
-------------------------------------------------------------------------------
Total Entries : 1
-------------------------------------------------------------------------------
```

Services

The services supported with SRv6 uSID are identical to the services supported with generic SRv6 and include support for VPRN (End.DT4, End.DT6), VPRN services with EVPN-IFL (again End.DT4, End.DT6), EVPN-VPLS (End.DT2U, End.DT2M), EVPN-VPWS (End.DX2), and IPv4/IPv6 BGP routes (End.DT4, End.DT6) in the base routing instance. However, the nomenclature used when applying SRv6 uSID within a service context differs slightly to that of generic SRv6. The intention of this sub-section therefore is not to describe the entire process of service SID advertisement and data-plane verification since it is the same as generic SRv6 previously described in this chapter. Rather, this section will simply provide two examples of SRv6 uSID configuration within a service context.

Output 12-17 and surrounding narrative provides an example of a VPRN service with generic SRv6. By contrast, *Output 12-52* provides an example of a VPRN service using SRv6 uSID and it can be seen that the structure is entirely the same as generic SRv6, with the notable differences being:

a) The service-level **segment-routing-v6** instance context refers to a **micro-segment-locator** (as opposed to a **locator**).
b) Within the **bgp-ipvpn** context the same **segment-routing-v6** instance uses the **srv6 default-locator** command to reference a uSID Locator.

As with generic SRv6, the use of this **default-locator** means that in the current SR-OS release the association of prefixes to a particular Flex-Algorithm identifier has per-VPRN granularity.

Output 12-52: VPRN Service with SRv6 uSID

```
    service {
        vprn "one" {
            segment-routing-v6 1 {
                micro-segment-locator "uSID-Alg128-Locator" {
                    function {
                        udt4 {
                        }
                        udt6 {
                        }
                    }
                }
            }
            bgp-ipvpn {
                segment-routing-v6 1 {
                    admin-state enable
                    route-distinguisher "64496:1"
                    vrf-import {
                        policy ["vrf-one-import"]
                    }
                    vrf-export {
                        policy ["vrf-one-export"]
                    }
                    srv6 {
                        instance 1
                        default-locator "uSID-Alg128-Locator"
                    }
                }
            }
        }
    }
```

Output 12-53 provides a second example of SRv6 uSID application within a service context, this time using an EVPN-VPWS service. Like the VPRN service the service-level **segment-routing-v6** context references the associated **micro-segment-locator**, which in this example is the algorithm 0 uSID Locator. The desired endpoint behaviour is defined within the **function** context, which in this case is uDX2 for decapsulation and L2 cross-connect with NEXT-CSID. Within the **bgp-evpn** context the **segment-routing-v6** context has exactly the same structure as a VPRN (or **bgp-ipvpn**), and in this example the **default-locator** references the algorithm 0 uSID Locator.

Output 12-53: EVPN-VPWS Service with SRv6 uSID

```
    service {
        epipe "four" {
            segment-routing-v6 1 {
                micro-segment-locator "uSID-Alg0-Locator" {
                    function {
                        udx2 {
                        }
                    }
                }
            }
            bgp-evpn {
                evi 4
                local-attachment-circuit "local-ac" {
                    eth-tag 4
                }
                remote-attachment-circuit "remote-ac" {
                    eth-tag 4
                }
                segment-routing-v6 1 {
                    admin-state enable
                    srv6 {
                        instance 1
                        default-locator "uSID-Alg0-Locator"
                    }
                    route-next-hop {
                        system-ipv6
                    }
                }
            }
        }
    }
```

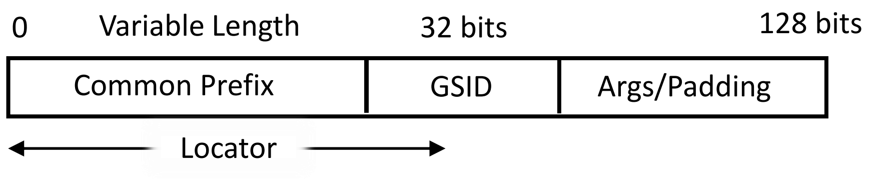
This section has detailed the use of shortest-path SRv6 uSID, and it should be reasonably clear that the use and application of SRv6 uSID has a configuration structure that is almost identical to that of generic SRv6. Although the current release of SR-OS only provides support for shortest-path SRv6 uSID, it will show its capability over that of generic SRv6 when it is used in an SRv6 traffic engineering environment. That capability will be supported in a future SR-OS release.

The G-SID (REPLACE-CSID Flavour)

Like the uSID proposal, G-SID attempts to avoid repeating the Locator Block in the SIDs contained in the segment list. A G-SID can be a 32-bit value or a 16-bit value of the original

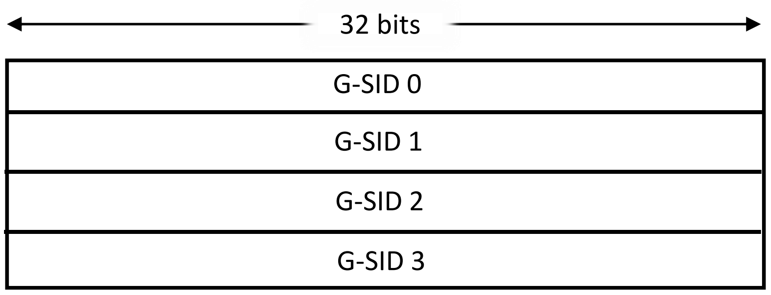
SRv6 SID with the common prefix removed, and contains the Node ID and Function ID. The intent is that by carrying G-SIDs instead of 128-bit SRv6 SIDs the size of the SRH can be significantly reduced. The G-SID follows the common prefix in the destination address. The common prefix is the prefix shared by the compressible SRv6 SIDs and is usually the Locator Block.

Figure 12-22: G-SID Format



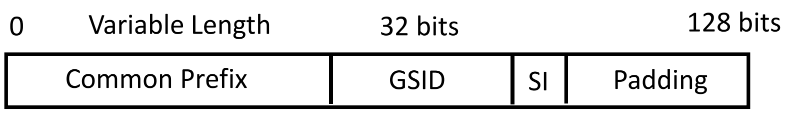
G-SRv6 defines 128-bit G-SID Containers within the SRH that may contain a conventional SRv6 SID, a uSID carrier, or multiple G-SIDs. Assuming a G-SID of 32-bits, then up to four G-SIDs can be carried in a single 128-bit G-SID Container. If the length of the G-SIDs in a G-SID Container is less than 128-bits, padding is required.

Figure 12-23: G-SID Container with Four G-SIDs



To locate the G-SID within the G-SID Container a Generalised SID Index (SI) is used in the IPv6 destination address. The SI is a location argument of the G-SID, and when the G-SID is 32-bits, the SI is the least significant two bits of the argument. The SI is used in conjunction with the SRH Segments Left (SL) field. The SL indicates the whereabouts of the G-SID Container, and therefore represents no change in behaviour from native SRv6. The SI indicates the whereabouts of the G-SID within that G-SID Container.

Figure 12-24: IPv6 Destination Address with G-SID and SI

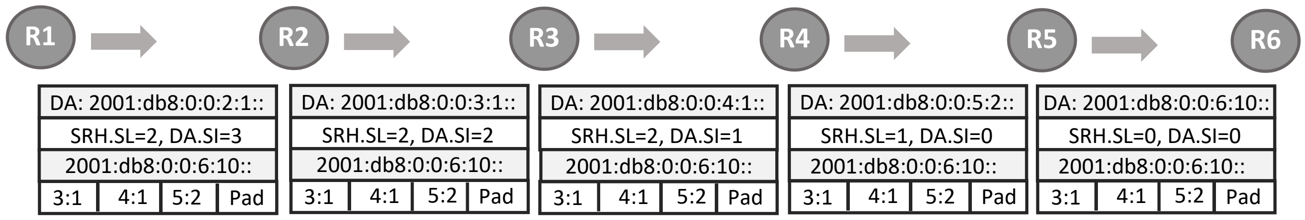


During G-SRv6 compression processing, each segment endpoint must update the G-SID part of the IPv6 destination address with the next 32-bit G-SID in the SRH. In the original G-SID specification this was referred to as Continue of Compression (COC). In the C-SID specification it is referred to as the REPLACE-C-SID flavour. When a node receives a SID without the COC flavour the node processes the packet as a normal SRv6 packet, updating the IPv6 destination address with the next 128-bit SID if SL is greater than 0. When a node receives a SID with COC flavour, the node updates the G-SID part of the IPv6 destination address with the next 32-bit G-SID. The last G-SID in the G-SID list containing the service SID has no flavour, therefore the next 128-bit SID will be copied to the IPv6 destination address (reflecting the end of the compression sub-path).

Consider the example of packet forwarding with G-SID in *Figure 12-25*. Routers R1 through R6 are G-SRv6 enabled nodes in the SRv6 domain and use 2001:db8::/64 as the common prefix (Locator Block). The G-SID follows the common prefix and is 32-bits in this illustration. The first 16-bits of the G-SID represent the Node ID using a simple representation of the router number, while the second 16-bits represent the Function. Function 1 represents an End.X SID with COC. Function 2 represents an End.X SID without COC. For example, the SID 2001:db8:0:0:2:1:: represents an End.X SID with COC at node 2, and the SID 2001:db8:0:0:4:2:: represents an End.X SID without COC at node 4. The SID 2001:db8:0:0:6:10:: is an End.DT4 SID initiated by node 6 which is associated with VRF 10. With G-SID the last segment in the SID list is always uncompressed in this manner.

A G-SID path from R1 to R6 containing strict hops is shown in the figure. The packet contains two G-SID Containers. Container 0 contains the uncompressed service SID for VRF 10 at R6, while container 1 contains three compressed G-SIDs representing the End.X SIDs with COC at R3 and R4, followed by an End.X SID without COC at R5. When R1 forwards the packet the destination address contains the common prefix and an End.X SID at R2, the SL is 2, and the SI is 3. When the packet reaches R2 it copies the G-SID indicated by the SL and SI fields into the destination address, decrements the SI field, and forwards the packet towards R3. Routers R3 and R4 do likewise, with R4 also decrementing the SL field in addition to the SI field (G-SID position 0 is ignored as it is padding) before forwarding to R5. When router R5 receives the packet the SID 2001:db8:0:0:5:2 indicates an End.X SID without COC, so it is the end of the compression chain. The SL is also 1, so R5 copies the uncompressed SID 2001:db8:0:0:6:10 into the destination address and forwards to R6 where it is processed as an End.DT4 SID.

Figure 12-25: G-SID Packet Forwarding Example



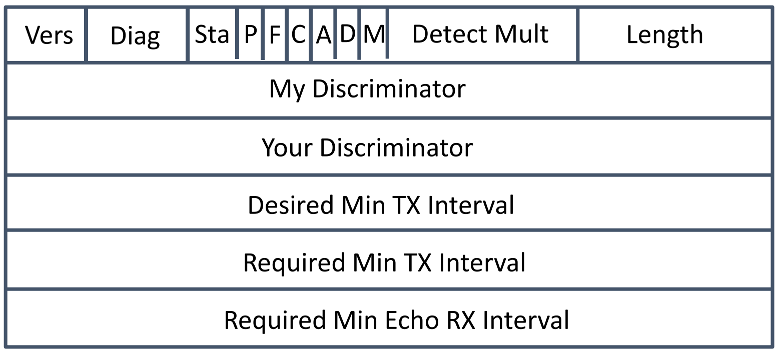
13 Seamless-BFD and End-to-End Protection

This chapter discusses how Seamless-BFD (S-BFD) may be used to provide a continuity check for SR-TE LSPs and SR Policies, and how it can be used to provide end-to-end protection by triggering a switchover to a protect path at the LSP headend. The chapter begins with a short overview of BFD and how it is modified for S-BFD, and then illustrates how it can be used for SR-TE LSPs and SR Policies.

BFD Overview

BFD is a lightweight 'hello' protocol used to detect forwarding failures. It is widely used in operator networks and can be bootstrapped to various protocols such as IS-IS, OSPF, and BGP. Even though the protocol is lightweight and simple, there are certain scenarios where faster setup of sessions and faster continuity checks of the forwarding plane are necessary. BFD requires two nodes to exchange locally allocated discriminators. These discriminators enable the identification of the sender and the receiver of BFD packets over the particular session and act as a demultiplexer in the event of multiple BFD sessions running between two systems. A BFD session begins with the periodic, slow transmission of BFD control packets during which the local and remote discriminators are exchanged. The local discriminator is sent in the My Discriminator field in the BFD control packet and is echoed back in the Your Discriminator field of packets sent from the remote end. When bidirectional communication is achieved, the BFD session is declared up. BFD packets are then sent using the appropriate encapsulation and perform proactive continuity monitoring of the forwarding path between the two nodes. For reference the mandatory section of a BFD Control packet format is shown in *Figure 13-1*. An optional Authentication section exists but is not shown as it is not supported in SR-OS.

Figure 13-1: BFD Control Packet



Diagnostics Field
0 – No Diagnostic
1 – Control Detection Time Expired
2 – Echo Function Failed
3 – Neighbour Signalled Session Down
4 – Forwarding Plane Reset
5 – Path Down
6 – Concatenated Path Down
7 – Administratively Down
8 – Reverse Concatenated Path Down
9-31 Reserved

State Field
0 – AdminDown
3 -- Up

The diagnostics (Diag) field is used to specify the local system's reason for the last change in session state. The state (Sta) field indicates the current BFD session state as seen by the transmitting system. Possible settings for the diagnostics and state fields are contained within the figure. The flags field contains six flags that are defined as follows:

Poll (P)	If set, the transmitting system is requesting verification of connectivity and is expecting a packet with the Final (F) bit in reply.
Final (F)	If set, the transmitting system is responding to a received S-BFD Control packet that had the Poll (P) bit set.
Control Plane Independent (C)	If set, the transmitting system's BFD implementation does not share fate with its control plane. In other words, BFD can survive a control plane restart. SR-OS sets this bit to 1 on FP-based platforms for BFD sessions that run on the CPM P-chip or the line-card, and to 0 for BFD sessions that run on the CPU of the CPM. On the 7250 IXR all session types are central except those which are of type FP.
Authentication Present (A)	If set the Authentication Section is present.
Demand (D)	If set, Demand mode is active in the transmitting system.
Multipoint (M)	Reserved for future point-to-multipoint extensions to BFD.

If the discriminator information exchanged in control packets is already known to the endpoints of a potential BFD session, the initial handshake including an exchange of discriminators is unnecessary, and it is possible for the endpoints to begin BFD messaging seamlessly. This is the premise of Seamless-BFD.

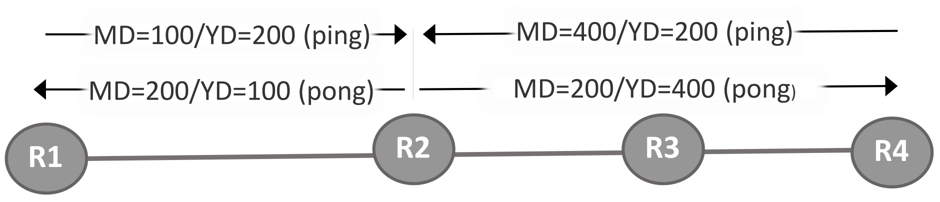
A node that runs Seamless BFD (S-BFD) allocates one or more local S-BFD Discriminators and creates a Reflector BFD session. Mappings between allocated S-BFD Discriminators and network nodes may be statically configured network-wide, or each node may advertise its allocated S-BFD Discriminators into OSPF or IS-IS. The net result is that other network nodes learn about the S-BFD Discriminators allocated by a remote node to a remote entity. Once this information is known, any node can quickly perform a continuity test to the remote entity simply by sending S-BFD Control packets with a corresponding S-BFD Discriminator value in the Your Discriminator field. The Reflector BFD session, upon receiving an S-BFD Control packet targeted to one of the local S-BFD Discriminator values transmits a response S-BFD Control packet back to the initiator.

The Conventional BFD state machine has four potential states, AdminDown, Down, Init, and Up. There are three states through which a session normally proceeds, two for establishing a session (Init and Up) and one for tearing down a session (Down). During setup these states are exchanged in a three-way handshake, starting with a Down state, then moving to the Init state once the local and remote systems start to communicate, then advancing to the Up state when a BFD Control packet is received that is signalling Init or Up. With S-BFD, there is no three-way exchange to bring the session up, hence only two states exist, Up and AdminDown. An S-BFD Reflector can only ever return one of these two states.

Consider the example in *Figure 13-2*. Routers R1, R2, and R4 are allocated S-BFD Discriminator values 100, 200, and 400 respectively, which are advertised into the IGP. Upon receiving the advertisement, a node can associate an S-BFD Discriminator value with a

Router-ID or System-ID for a given remote node. Hence, all nodes know Discriminator values for remote nodes. Routers R1, R2, and R4 subsequently create Reflector sessions and listen on the well-known port for S-BFD, 7784. When R1 wants to send a continuity check to R2 it sends an S-BFD Control packet with the Your Discriminator field set to 200. When the Reflector BFD session on R2 receives this packet, a response S-BFD Control packet is sent back to R1, inverting the My Discriminator and Your Discriminator values, and indicating a state of Up or AdminDown. Similarly, when R4 wants to send a continuity check to R2 it sends an S-BFD Control packet with the Your Discriminator field set to 200. When the Reflector BFD session on R2 receives this packet, a response S-BFD Control packet is sent back to R4, again inverting the My Discriminator and your Discriminator values, and indicating a state of Up or AdminDown. The reception of a BFD response Control packet at the Initiator with state of AdminDown does not mean that reachability of the corresponding remote entity is lost, but rather that the packet transmission interval should be backed off. Loss of continuity to the remote entity can only be concluded when a sufficient number of S-BFD packets have not arrived as they should.

Figure 13-2: S-BFD Example



Router	Discriminator Value
R1	100
R2	200
R4	400

To allow dissemination of S-BFD Discriminator values and therefore permit a dynamic mapping to S-BFD Discriminator allocation to remote System ID, both IS-IS and OSPF have S-BFD extensions [47] [48]. In IS-IS an S-BFD Discriminators sub-TLV is defined that is carried as a sub-TLV of the Router Capability TLV. In OSPF an S-BFD Discriminator TLV is defined that is carried as a TLV of the Router Information Opaque LSA. It's worth noting that by default the Router Capability TLV and the RI Opaque LSA both have area flooding scope. That is to say that a dynamic mapping of system ID to allocated S-BFD Discriminator can only be made within the flooding scope of those TLVs/LSAs.

BFD has two operating modes that may be used, as well as an additional function that can be used in either mode:

- The primary mode is known as asynchronous mode. In this mode the systems periodically send BFD control packets to one another, and if a number of consecutive packets are not received by the peer the session is declared down. This is the default mode used in SR-OS.

– A second mode is known as demand mode. In this mode, it is assumed that a system has another independent way of verifying connectivity to the remote system. As a result, it may ask the other system to stop sending BFD control packets unless there is an explicit need to verify connectivity, in which case a short sequence of BFD control packets is exchanged. Demand mode is not currently supported in SR-OS.

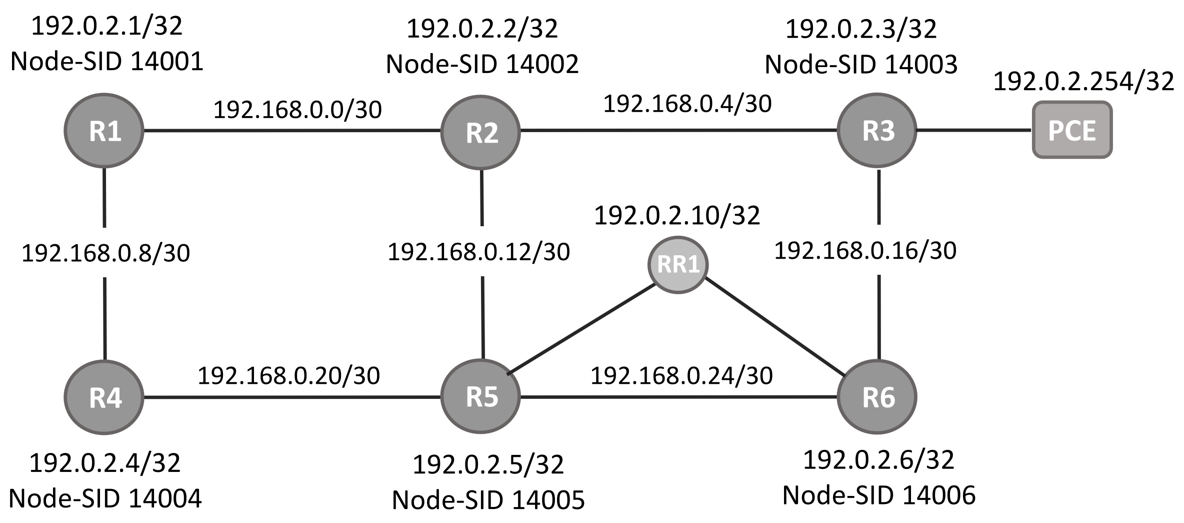
An additional function that may be used with both modes is known as the echo function. When the echo function is used, a stream of BFD echo packets is transmitted to the remote system in such a way as to have the other system loop them back through its forwarding path. If a number of packets of the echoed data stream are not received, the session is declared down. The echo function has the advantage of truly testing the forwarding path on the remote system, but since the system initiating BFD echo packets also has the task of fault detection it has the disadvantage that it needs twice as many packets as asynchronous mode to detect a failure. The echo function can be enabled for S-BFD operating on SR-TE LSPs, and this is described later in this section.

S-BFD Control packets are transmitted with an IP header, UDP header, and BFD Control packet. The source IP address is be set to a local IP address that is reachable from the destination. For IP-routed S-BFD packets the destination address is set to the IP address of the target. For label-switched S-BFD packets, a label stack is prepended, and the destination address IP address is chosen from the 127/8 range for IPv4 and from the 0:0:0:0:0:ffff:7f00:0/104 range for IPv6. For S-BFD the UDP port is set to the well-known port of 7784.

S-BFD Configuration

Before describing how S-BFD is employed by SR-TE LSPs and SR Policies some basic configuration requirements for S-BFD that are applicable to both are described. The network topology in *Figure 13-3* is used for the illustration. Routers R1 to R6 are in a flat IS-IS Level 2 domain and SR is enabled with an SRGB of 12000-19999. The relevant IP addresses and Node-SIDs are shown in the figure. A Route-Reflector is used to simplify propagation of SR Policies, and to that extent Routers R1 to R6 are configured as clients of the Route-Reflector for the IPv4 SR Policy Address Family.

Figure 13-3: Test Topology for S-BFD



Initially the S-BFD Reflector is enabled, and this is any node that needs to reflect S-BFD packets. Within the **seamless-bfd** context a reflector is created and is allocated a **discriminator** value in the range 524288-526335. SR-OS currently supports a single Discriminator per system and the same discriminator can be used at every node as the initiator local discriminator value is randomly generated thereby creating uniqueness for the purpose of demultiplexing. Lastly the reflector is put into an **admin-state** of **enable**.

Output 13-1: R3 S-BFD Reflector Configuration

```
    bfd {
        seamless-bfd {
            reflector "sbfd-reflector" {
                admin-state enable
                discriminator 524288
            }
        }
    }
```

Note: If the Reflector is an FP-based platform it should use FP3 or later on all linecards used for network ports as received packets are turned around on the linecard. By default, SR-OS returns all S-BFD packets as native IP unless BFD Echo mode is used, and a **return-path-label** is specified (described later in this chapter).

The propagation of S-BFD Discriminator values into IS-IS or OSPF requires no explicit configuration as long as the **advertise-router-capability** command is set to **true** within the IS-IS or OSPF context. In the test topology where IS-IS is used, the S-BFD Discriminator value is carried in the Router Capability TLV and is flooded through the entire topology as it is a flat Level-2 area. *Output 13-2* shows the Level-2 LSP advertised by R3 as received at R1, truncated to show only the Router ID and Router Capability TLV (the final line of the output shows the S-BFD Discriminator). Together these two TLVs provide sufficient information to allow a remote node to create a dynamic peer-to-discriminator mapping table.

Output 13-2: Propagation of S-BFD Discriminator into IS-IS

```
A:admin@R1# show router isis database R3.00-00 detail level 2

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================
...[snip]
TLVs :
  IS-Hostname   : R3
  Router ID   :
    Router ID   : 192.0.2.3
  Router Cap : 192.0.2.3, D:0, S:0
    TE Node Cap : B E M  P
    SR Cap: IPv4 MPLS-IPv6
       SRGB Base:12000, Range:8000
    SR Alg: metric based SPF
    Node MSD Cap: BMI : 12 ERLD : 15
    S-BFD discr : 524288
... [snip]
```

At the initiator, a **bfd-template** must be created within the **bfd** context. The template allows for configuration of the **receive-interval** and **transmit-interval** of BFD packets, specified in milliseconds. The minimum **transmit-interval** is 10 milliseconds, and an SR-OS Reflector will respond with a Required Minimum RX Interval of 3 milliseconds to indicate that this is the minimum interval of BFD Control packets that it supports. The **type** command specifies the whereabouts in the system that the BFD packets will be processed. For maximum scale it is recommended that BFD sessions are terminated on the P-chip of the CPM, and this is selected using the **cpm-np** argument. The **bfd-template** has no default setting for the **type** command, but the system will automatically associate multi-hop BFD sessions with the CPM P-chip (CPM-NP). The configuration of **type cpm-np** shown in the output is for clarity about where the BFD packets are processed.

Output 13-3: Initiator S-BFD Template

```
    bfd {
        bfd-template "sbfd-template" {
            receive-interval 5000
            transmit-interval 5000
            type cpm-np
        }
    }
```

Note: SR-OS automatically associates a BFD session with the CPM P-chip under the following circumstances:

- Any session with a timer interval less than 100 milliseconds.
- Any BFD session associated with a LAG interface.
- BFD sessions associated with spoke-termination interfaces.
- All multi-hop BFD sessions.
- If the number of sessions configured on a line-card exceeds the line-card’s limits.

In the test topology the Router Capability TLV in IS-IS is propagated throughout the entire domain, therefore it is possible to create the dynamic peer-to-discriminator mapping table.

In the event that the network is multi-area or multi-domain this is not possible, and a static peer-to-discriminator mapping table must be configured. Although this is not required in this test topology, for reference *Output 13-4* shows the configuration of a static mapping table at R1 for peer R3 (192.0.2.3). Within the **seamless-bfd** context, this simply requires the definition of a **peer**, and the relevant **discriminator** value configured within that **peer** context. This table is used to determine the relevant discriminator value as follows:

- PCC-initiated SR-TE LSPs use the '**to**' value in the LSP configuration to reference this table.
- PCE-initiated SR-TE LSPs use the Tunnel Endpoint Address of the PCInit message to reference this table.
- SR Policies use the Endpoint address to reference this table.

Output 13-4: Initiator Peer-to-Discriminator Mapping Table

```
    router "Base" {
        bfd {
            seamless-bfd {
                peer 192.0.2.3 {
                    discriminator 524288
                }
            }
        }
    }
```

Note: If the Initiator is an FP-based platform it requires FP2 hardware or higher and CPM5 or higher.

SR-TE LSPs with S-BFD

This section covers the configuration of S-BFD on both PCC-initiated SR-TE LSPs and PCE-initiated SR-TE LSPs. The configuration requirements differ a little given the nature of how they are instantiated. This is followed by a look at the use of the BFD echo function.

PCC-Initiated SR-TE LSPs with S-BFD

To provide an example of the use of PCC-Initiated SR-TE LSPs with S-BFD, an SR-TE LSP is created from R1 to R3 consisting of primary and secondary paths. To enable diversity between the primary path and the secondary path of the R1-to-R3 LSP their paths are computed by a PCE, and a Path-Profile is used by the PCC to indicate the requirement for diversity. This does not imply that using a PCE is the only way to support Primary/Secondary SR-TE LSPs; it is equally applicable to any form of SR-TE LSP.

SR-OS allows for S-BFD to be applied to both PCC-Initiated and PCE-Initiated SR-TE LSPs at LSP level, and at primary/secondary path level for PCC-Initiated SR-TE LSPs. When applied at LSP level asynchronous mode S-BFD sessions are created in each path of the LSP. In the absence of any control plane for path setup and maintenance, S-BFD provides a continuity check and health-check mechanism that can trigger restorative action in the event of failure.

Output 13-5 shows the configuration of the SR-TE LSP with primary and secondary paths. Most of the SR-TE LSP parameters have already been described in chapter 6, so only the additional configuration is described here. The **path-profile**, which as discussed in chapter 6 is used to associate the LSP with a centralised policy, has an additional **path-group** command configured. Recall that the purpose of path-profiles or association groups is to allow policy to be applied to a set of LSPs. In this example the primary path and secondary paths are separate path computation requests that will be subject to separate path computations by the PCE. These two paths are therefore grouped together by virtue of the fact that they belong to the same **path-group**. In other words, the **path-profile** of 30 references a policy that specifies the constraints (link diversity), and the **path-group** of 1 specifies that both paths have the same group membership and therefore should be diverse from each other.

The **bfd** context uses the **bfd-template** to reference a template defining the session parameters, and in this case calls the template configured in *Output 13-3*. The **failure-action** command specifies the action to be taken when an S-BFD session fails. Although options exist for actions of **down**, **failover**, and **failover-or-down**, only the latter can be used with SR-TE LSPs. A path will be considered operationally down when its associated S-BFD session is down. When the **failure-action** is **failover-or-down** and S-BFD is enabled at LSP level, the failure of an S-BFD session will trigger the switchover of the active path to another available path (primary or secondary). Using a **failure-action** of **failover-or-down** is a valid configuration for SR-TE LSPs that consist of only a primary path if fallback to another tunnel technology of higher numerical preference is a desired action. For example, assume a service that uses **auto-bind-tunnel** with a **resolution-filter** of **sr-te** and **sr-isis**. The SR-TE LSP will be preferred for next-hop resolution all the time it is operationally up as it has a lower tunnel-table preference than shortest path SR. If the SR-TE LSP fails, it will be detected by S-BFD and declared down, after which a shortest path SR tunnel will be used until such time that the S-BFD is restored, and the SR-TE LSP declared operationally up.

Last but not least, the **bfd-liveliness** command is set to **true** to enable the use of S-BFD.

Output 13-5: R1-to-R3 Primary/Secondary LSP with S-BFD

```
        mpls {
            path "empty" {
                admin-state enable
            }
            path "sec-empty" {
                admin-state enable
            }
            lsp "R1-to-R3" {
                admin-state enable
                type p2p-sr-te
                to 192.0.2.3
                pce-control true
                pce-report true
                path-computation-method pce
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                bfd {
                    bfd-liveness true
                    bfd-template "sbfd-template"
                    failure-action failover-or-down
                }
                path-profile 30 {
                    path-group 1
                }
                primary "empty" {
                    admin-state enable
                }
                secondary "sec-empty" {
                    admin-state enable
                    standby true
                }
            }
        }
```

Output 13-6 shows the operational state of the S-BFD sessions for the LSP R1-to-R3 and as can be seen two S-BFD sessions are present, one for each path of the LSP. The additional 'detail' argument of this command gives further information on session state, such as up time, number of transitions, as well as local and remote discriminator values.

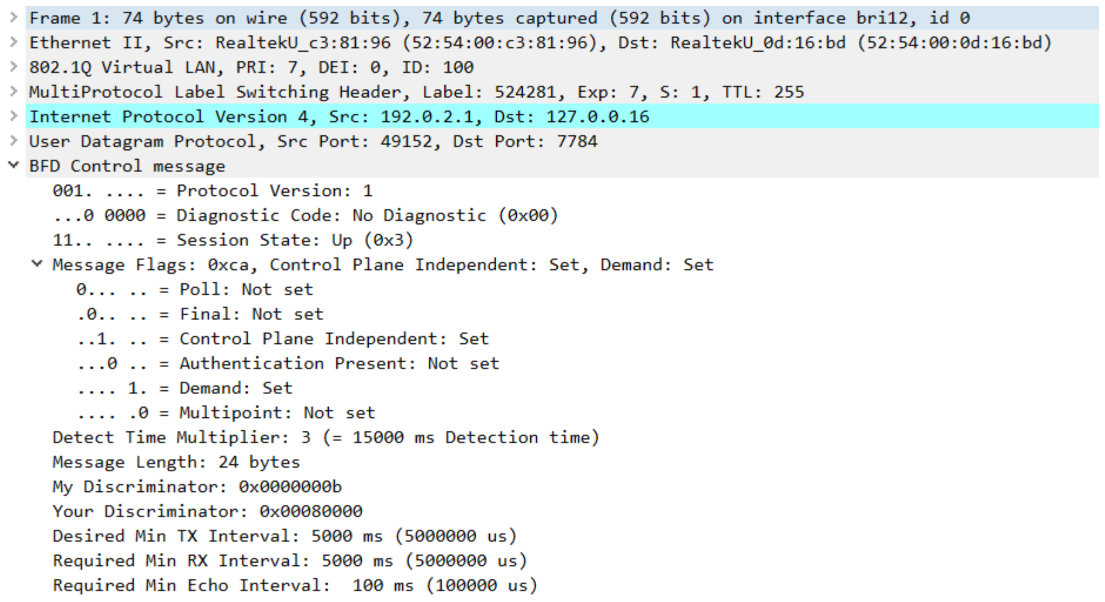
Output 13-6: Verification of S-BFD Session State

```
A:admin@R1# show router bfd seamless-bfd session lsp-name "R1-to-R3"

===============================================================================
Legend:
  Session Id = Interface Name | LSP Name | Prefix | RSVP Sess Name | Service Id
  wp = Working path   pp = Protecting path
===============================================================================
BFD Session
===============================================================================
Session Id                                        State      Tx Pkts    Rx Pkts
  Rem Addr/Info/SdpId:VcId                      Multipl     Tx Intvl   Rx Intvl
  Protocols                                        Type     LAG Port     LAG ID
  Loc Addr                                                             LAG name
-------------------------------------------------------------------------------
192.0.2.3/32                                         Up          N/A        N/A
  192.0.2.3                                           3         5000       5000
  mplsLsp                                        cpm-np          N/A        N/A
  192.0.2.1
192.0.2.3/32                                         Up          N/A        N/A
  192.0.2.3                                           3         5000       5000
  mplsLsp                                        cpm-np          N/A        N/A
  192.0.2.1
-------------------------------------------------------------------------------
No. of BFD sessions: 2
```

Figure 13-4 is a packet capture of an S-BFD packet forwarded on the R1-to-R3 SR-TE LSP. The source IP address is R1's system address, while the destination address is 127.0.0.16. The destination port is the well-known port for S-BFD. The session state is shown as Up.

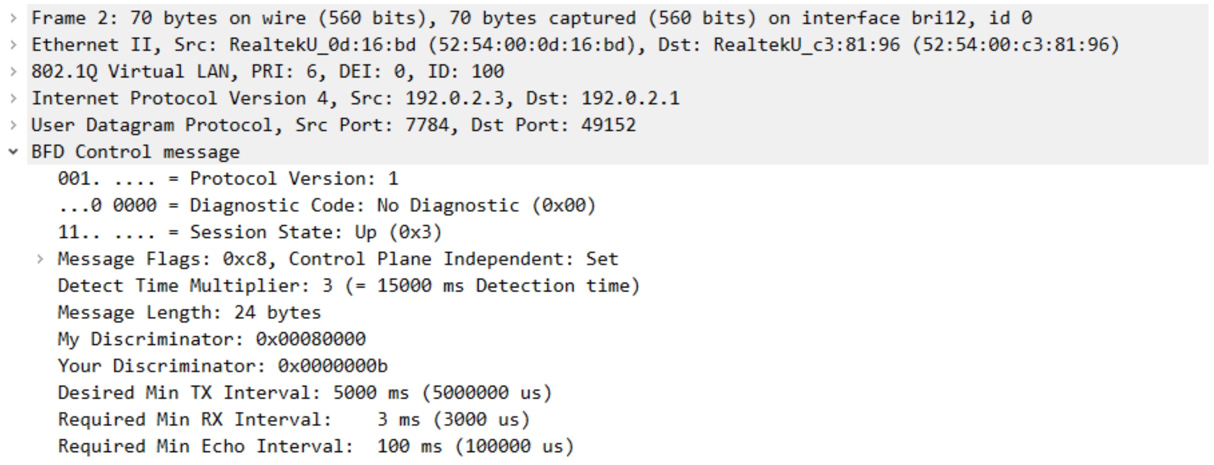
Figure 13-4: S-BFD Control Packet R1 to R3



```
> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_c3:81:96 (52:54:00:c3:81:96), Dst: RealtekU_0d:16:bd (52:54:00:0d:16:bd)
> 802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 524281, Exp: 7, S: 1, TTL: 255
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.16
> User Datagram Protocol, Src Port: 49152, Dst Port: 7784
v BFD Control message
    001. .... = Protocol Version: 1
    ...0 0000 = Diagnostic Code: No Diagnostic (0x00)
    11.. .... = Session State: Up (0x3)
  v Message Flags: 0xca, Control Plane Independent: Set, Demand: Set
      0... .. = Poll: Not set
      .0.. .. = Final: Not set
      ..1. .. = Control Plane Independent: Set
      ...0 .. = Authentication Present: Not set
      .... 1. = Demand: Set
      .... .0 = Multipoint: Not set
    Detect Time Multiplier: 3 (= 15000 ms Detection time)
    Message Length: 24 bytes
    My Discriminator: 0x0000000b
    Your Discriminator: 0x00080000
    Desired Min TX Interval: 5000 ms (5000000 us)
    Required Min RX Interval: 5000 ms (5000000 us)
    Required Min Echo Interval:  100 ms (100000 us)
```

A corresponding S-BFD control packet from R3 to R1 is shown in *Figure 13-5*. Unlike the packet from R1 to R3 that is encapsulated in an MPLS label stack, the return control packet is native IP/UDP. Note also that while the source/destination ports are inverted, the source/destination IP addresses are not. The destination address of 127.0.0.16 is removed and replaced with R3's system address (192.0.2.3) as a source IP address. This is perfectly acceptable because R1 uses the Your Discriminator field of the returned BFD control packets to identify the session. Once the reflector echoes back the local discriminator, all further received packets are demultiplexed based on the Your Discriminator field only.

Figure 13-5: S-BFD Control Packet R3 to R1



```
> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_0d:16:bd (52:54:00:0d:16:bd), Dst: RealtekU_c3:81:96 (52:54:00:c3:81:96)
> 802.1Q Virtual LAN, PRI: 6, DEI: 0, ID: 100
> Internet Protocol Version 4, Src: 192.0.2.3, Dst: 192.0.2.1
> User Datagram Protocol, Src Port: 7784, Dst Port: 49152
v BFD Control message
    001. .... = Protocol Version: 1
    ...0 0000 = Diagnostic Code: No Diagnostic (0x00)
    11.. .... = Session State: Up (0x3)
  > Message Flags: 0xc8, Control Plane Independent: Set
    Detect Time Multiplier: 3 (= 15000 ms Detection time)
    Message Length: 24 bytes
    My Discriminator: 0x00080000
    Your Discriminator: 0x0000000b
    Desired Min TX Interval: 5000 ms (5000000 us)
    Required Min RX Interval:    3 ms (3000 us)
    Required Min Echo Interval:  100 ms (100000 us)
```

The primary path of the LSP routes via R1-R2-R3, while the secondary path routes via R1-R4-R5-R6-R3 due to the application of the diversity policy during PCE path computation. In steady state both paths are operationally up.

Output 13-7: Primary/Secondary LSP Steady-State

```
A:admin@R1#  show router mpls sr-te-lsp "R1-to-R3" path detail | match "Path Type" post-
lines 1
```

```
Path Type          : Primary
Path Admin         : Up                         Path Oper          : Up
Path Type          : Standby
Path Admin         : Up                         Path Oper          : Up
```

The link R2-R3 is disabled causing the S-BFD session on the primary path to fail. The operational state of the path is immediately updated to reflect the failure.

Output 13-8: Primary/Secondary LSP Failure Condition

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3" path detail | match "Path Type" post-
lines 1
Path Type          : Primary
Path Admin         : Up                         Path Oper          : Down
Path Type          : Standby
Path Admin         : Up                         Path Oper          : Up
```

When the link R2-R3 is restored, the S-BFD session is re-established, and the primary path is once again declared operationally up.

PCE-Initiated SR-TE LSPs with S-BFD

To enable support of S-BFD for PCE-Initiated SR-TE LSPs, the **lsp-template** used to describe the characteristics of PCE-Initiated LSPs as described in chapter 6 has additional configuration added to include a **bfd-template** reference, a **failure-action** of **failover-or-down** and has **bfd-liveness** set to **true**.

Output 13-9: S-BFD Configuration for PCE-Initiated SR-TE LSPs

```
        mpls {
            lsp-template "pce-init-template" {
                bfd {
                    bfd-liveness true
                    bfd-template "sbfd-template"
                    failure-action failover-or-down
                }
            }
        }
```

A PCE-Initiated SR-TE LSP is sent from the PCE to R1 to establish an SR-TE LSP to R6. As shown the PCInit message successfully creates the SR-TE LSP.

Output 13-10: PCE-Initiated SR-TE LSP R1-to-R6

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R6"

===============================================================================
MPLS SR-TE LSPs (Originating)
===============================================================================
LSP Name                                           Tun     Protect   Adm  Opr
  To                                               Id      Path
-------------------------------------------------------------------------------
R1-to-R6                                           16388   N/A       Up   Up
  192.0.2.6
-------------------------------------------------------------------------------
LSPs : 1
```

And once again the S-BFD session is created to monitor the liveness of the R1-to-R6 LSP. In this example only a primary path was included in the PCInit message, hence only a single S-BFD session is spawned.

Output 13-11: S-BFD Session R1 to R6

```
A:admin@R1# show router bfd seamless-bfd session lsp-name R1-to-R6

===============================================================================
Legend:
  Session Id = Interface Name | LSP Name | Prefix | RSVP Sess Name | Service Id
  wp = Working path    pp = Protecting path
===============================================================================
BFD Session
===============================================================================
Session Id                                        State      Tx Pkts    Rx Pkts
  Rem Addr/Info/SdpId:VcId                      Multipl     Tx Intvl   Rx Intvl
  Protocols                                        Type     LAG Port     LAG ID
  Loc Addr                                                             LAG name
-------------------------------------------------------------------------------
192.0.2.6/32                                         Up          N/A        N/A
  192.0.2.6                                           3         5000       5000
  mplsLsp                                        cpm-np          N/A        N/A
  192.0.2.1
-------------------------------------------------------------------------------
No. of BFD sessions: 1
```

When using S-BFD to protect SR-TE LSPs, some consideration should be given to the transmit intervals used. Although a transmit interval of 10 milliseconds can be configured, it is not suggested that it should be used, particularly if LFA is enabled as this may well create a race condition in the event of failure. A reasonable suggestion would be to use timers of 100 milliseconds or greater.

SR-Policy with S-BFD

At the time of writing support of multiple Segment Lists within an SR Policy is not currently supported on 7250 IXR generation one or two platforms. Please check Release Notes for an updated list of 7250 IXR unsupported features.

S-BFD can provide a continuity check for the integrity of one or more segment lists of an MPLS or SRv6 SR Policy, and like SR-TE LSPs it can provide a trigger for taking restorative action when a session fails. When using S-BFD with SR Policies there are two potential modes of operation, ECMP protected mode and linear mode. In ECMP protected mode all segment lists of the top two candidate paths of an SR Policy are programmed into the datapath. Up to 32 segment lists can be supported per candidate path, and traffic is load-balanced across all of the active segment lists. A segment list is only included if its S-BFD session is up. ECMP protected mode is best suited to networks where there is a high level of ECMP coverage. Conversely, linear mode functions as active/standby with multiple backup paths to protect a primary path. It programs one segment list from each of the top three candidate paths into the datapath, with only one as active.

To illustrate the use of S-BFD with SR Policy the test topology from *Figure 13-3* is again used and an SR Policy is created from R1 to R3 with two candidate paths, each consisting of a single segment-list. The example will use S-BFD protection in linear mode. Routers R2 and R5 each advertise an IPv4 SR Policy with the tuple {headend=R1, color=100, endpoint=R3}. The candidate path advertised by R2 contains a single segment-list that routes R1-R4-R5-R6-R3. The candidate path advertised by R5 contains a single segment-list that routes R1-R2-R3. Different distinguisher values are used by each advertising router to ensure that both paths are propagated through the Route-Reflector towards R1. Note that I use SR-OS routers to source the BGP SR Policy paths simply because my PCE does not currently support the capability to advertise multiple candidate paths for the same SR Policy.

S-BFD is applied to an SR Policy using a **maintenance-policy** configured within the **segment-routing** context, and *Output 13-12* shows the **maintenance-policy** configured at the headend R1. The **maintenance-policy** defines the characteristics of the S-BFD protection applied to a given SR policy. Within that policy the **mode** command allows the user to select the desired protection mode and can be either **ecmp-protected** or **linear** as previously described. The example here uses **linear** mode. The **bfd-template** command references an active/configured **bfd-template**, and in this case the template previously configured in *Output 13-3* is used. The **bfd-liveness** command when set to **true** enables BFD on all of the segment lists of the relevant candidate path(s), and when set to **false** disables BFD. The **revert-timer** is used when the primary path recovers after a failure and specifies the time period in minutes that the system waits before reverting back to that path. If no **revert-timer** is configured the system will immediately revert after primary path recovery.

Output 13-12: SR Maintenance Policy

```
    router "Base" {
        segment-routing {
            maintenance-policy "sr-maint-policy" {
                admin-state enable
                bfd-liveness true
                bfd-template "sbfd-template"
                mode linear
                revert-timer 1
            }
        }
    }
```

Although not shown in *Output 13-12* the **threshold** command is worth discussing. When using ECMP protected mode the **threshold** command allows the user to specify how many S-BFD sessions need to be up before the candidate path itself is considered to be up. The command is not valid in linear mode since only one segment list and one S-BFD session is active for each candidate path. When the **threshold** command is used, an additional **hold-down-timer** can also be used to avoid situations where flapping BFD sessions cause the threshold to be repeatedly crossed. Once the number of S-BFD sessions that have failed drop below the configured **threshold**, the **hold-down-timer** specifies the amount of time that must elapse before the path will be considered up again providing the number of active S-BFD sessions is equal to or greater than the threshold.

The method for associating the **maintenance-policy** with an SR Policy depends on whether the SR Policy is statically configured or learned through BGP. For a static SR policy, the **maintenance-policy** can simply be configured within the **static-policy** context. For an SR policy learned through BGP the route-policy framework is used. The match or **from** criteria within a **policy-statement** entry allows the user to select any or all of family, color, distinguisher, and endpoint to identify a given SR policy, and in *Output 13-13* only the **family** and **color** are used. The action accepts the route(s) and associates the previously configured **maintenance-policy**.

Output 13-13: Associating the SR Maintenance Policy with BGP SR Policy

```
    policy-options {
        policy-statement "sr-policy-sbfd" {
            entry 10 {
                from {
                    family [sr-policy-ipv4]
                    color 100
                }
                action {
                    action-type accept
                    sr-maintenance-policy "sr-maint-policy"
                }
            }
        }
    }
```

It is then necessary to apply the policy as appropriate. If SR Policies are being used for BGP-based VPN services, my preference would be to apply the policy at VRF-import level. In this example however, I am not running any BGP-based VPN services, so the policy is applied at global BGP level.

Output 13-14: Applying Import Route-Policy

```
    router "Base" {
        bgp {
            neighbor "192.0.2.10" {
                import {
                    policy ["sr-policy-sbfd"]
                }
            }
        }
    }
```

R1 receives the candidate paths for the SR Policy from R2 and R5 and these are shown in *Output 13-15*. Note that these two paths are viewed as different NLRI because although they use the same color and endpoint, they were advertised with different distinguisher values to avoid the Route-Reflector running the best-path selection algorithm. Because the NLRI is formed of the tuple {color, distinguisher, endpoint} each path represents distinctly different NLRI. As a result, both are considered best and used by BGP.

Output 13-15: R1's RIB-In Showing SR Policy with Multiple Candidate Paths

```
A:admin@R1# show router bgp routes sr-policy-ipv4 color 100
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP SR-POLICY-v4 Routes
===============================================================================
Flag  RD/Color/End Point                                 LocalPref   MED
      Nexthop (Router)                                   Path-Id     IGP Cost
      As-Path                                                        Label
-------------------------------------------------------------------------------
u*>i  2001003/100/192.0.2.3                              100         None
      192.0.2.2                                          None        N/A
      No As-Path
u*>i  5001003/100/192.0.2.3                              100         None
      192.0.2.5                                          None        N/A
      No As-Path
-------------------------------------------------------------------------------
Routes : 2
```

Output 13-16 shows R1's BGP SR Policy candidate paths for color 100 and endpoint 192.0.2.3 (R3) and there are two paths present. The first path is the one advertised by R2 with route R1-R4-R5-R6-R3, and this path is currently not active. The second path is the one advertised by R5 with route R1-R2-R3, and this is active. SR Policy has a preference attribute that is used to select the best candidate path, and the higher preference wins. The route from R2 has a preference of 200, while the route from R5 has a preference of 300 therefore R5's candidate path is selected as active. It can also be seen from *Output 13-16* that the S-BFD state for both paths is up. One final point worth noting is that the BSID of the candidate paths should be the same in order for them to be considered part of the same SR Policy.

Output 13-16: BGP SR Policy with Linear Protection

```
A:admin@R1# show router segment-routing sr-policies bgp color 100 endpoint 192.0.2.3

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active          : No                    Owner           : bgp
Color           : 100
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
RD              : 2001003               Preference      : 200
BSID            : 20001
TunnelId        : 917511                Age             : 62693
Origin ASN      : 64496                 Origin          : 192.0.2.2
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 11/15/2021 17:16:19
Maintenance Policy: sr-maint-policy

Path Segment Lists:
Segment-List    : 1                     Weight          : 1
S-BFD State     : Up                    S-BFD Transitio*: 0
Num Segments    : 4                     Last Change     : 11/06/2021 11:37:33
  Seg 1 Label   : 524284                State           : resolved-up
  Seg 2 Label   : 524283                State           : N/A
  Seg 3 Label   : 524279                State           : N/A
  Seg 4 Label   : 14003                 State           : N/A

-------------------------------------------------------------------------------
Active          : Yes                   Owner           : bgp
Color           : 100
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
```

```
RD              : 5001003               Preference      : 300
BSID            : 20001
TunnelId        : 917511                Age             : 62693
Origin ASN      : 64496                 Origin          : 192.0.2.5
NumReEval       : 0                     ReEvalReason    : none
NumActPathChange: 0                     Last Change     : 11/15/2021 17:16:19
Maintenance Policy: sr-maint-policy

Path Segment Lists:
Segment-List    : 1                     Weight          : 1
S-BFD State     : Up                    S-BFD Transitio*: 0
Num Segments    : 2                     Last Change     : 11/06/2021 11:37:33
  Seg 1 Label   : 524286                State           : resolved-up
  Seg 2 Label   : 524281                State           : N/A

===============================================================================
* indicates that the corresponding row element may have been truncated.
```

Output 13-17 shows the operational state of the S-BFD sessions for the SR Policy from R1 to R3 and as can be seen two S-BFD sessions are present, one for each path of the policy. The additional 'detail' argument of this command gives further information on session state, such as up time, number of transitions, as well as local and remote discriminator values.

Output 13-17: S-BFD Sessions Spawned from the SR Policy

```
A:admin@R1# show router bfd seamless-bfd session sr-policy

===============================================================================
Legend:
  Session Id = Interface Name | LSP Name | Prefix | RSVP Sess Name | Service Id
  wp = Working path   pp = Protecting path
===============================================================================
BFD Session
===============================================================================
Session Id                                        State      Tx Pkts    Rx Pkts
  Rem Addr/Info/SdpId:VcId                      Multipl     Tx Intvl   Rx Intvl
  Protocols                                        Type     LAG Port     LAG ID
  Loc Addr                                                             LAG name
-------------------------------------------------------------------------------
192.0.2.3/32                                         Up          N/A        N/A
  192.0.2.3                                           3         5000       5000
  srPolicy                                       cpm-np          N/A        N/A
  192.0.2.1
192.0.2.3/32                                         Up          N/A        N/A
  192.0.2.3                                           3         5000       5000
  srPolicy                                       cpm-np          N/A        N/A
  192.0.2.1
-------------------------------------------------------------------------------
No. of BFD sessions: 2
```

The active candidate path of the SR Policy routes via R1-R2-R3, while the inactive candidate path routes via R1-R4-R5-R6-R3. In steady state the path from R2 with discriminator (shown as RD) 2001003 and preference 200 is inactive and the path from R5 with RD 5001003 and preference 300 is active. S-BFD is up on both paths.

Output 13-18: SR Policy in Non-Failure Mode

```
A:R1# show router segment-routing sr-policies bgp color 100 end-point 192.0.2.3 | match
expression "Active|RD|S-BFD State"
Active          : No                    Owner           : bgp
RD              : 2001003               Preference      : 200
S-BFD State     : Up                    S-BFD Transitio*: 0
```

```
Active              : Yes                     Owner           : bgp
RD                  : 5001003                 Preference      : 300
S-BFD State         : Up                      S-BFD Transitio*: 0
```

The link R2-R3 is disabled causing the S-BFD session on the active candidate path to fail. The operational state of the SR Policy is immediately updated to reflect the failure. The previously active candidate path from R5 is now inactive and its S-BFD state is shown as down. Conversely, the previously inactive candidate path from R2 is now active.

Output 13-19: SR Policy in Failure Condition

```
A:R1# show router segment-routing sr-policies bgp color 100 end-point 192.0.2.3 | match
expression "Active|RD|S-BFD State"
Active              : Yes                     Owner           : bgp
RD                  : 2001003                 Preference      : 200
S-BFD State         : Up                      S-BFD Transitio*: 0
Active              : No                      Owner           : bgp
RD                  : 5001003                 Preference      : 300
S-BFD State         : Down                    S-BFD Transitio*: 1
```

When the link R2-R3 is restored, the S-BFD session is re-established, and the active candidate path is reverted to the one advertised by R5 with RD 5001003 and preference 300. Both S-BFD sessions are now re-established.

Output 13-20: SR Policy after Link Failure Restoral

```
A:R1#  show router segment-routing sr-policies bgp color 100 end-point 192.0.2.3 | match
expression "Active|RD|S-BFD State"
Active              : No                      Owner           : bgp
RD                  : 2001003                 Preference      : 200
S-BFD State         : Up                      S-BFD Transitio*: 0
Active              : Yes                     Owner           : bgp
RD                  : 5001003                 Preference      : 300
S-BFD State         : Up                      S-BFD Transitio*: 1
```

One final point of note with regard to the use of S-BFD with SR Policies. If the maximum supported number of paths has already been programmed (two for ECMP protected mode and three for linear mode) and a new path is received with a higher (better) preference value than the existing active path, this new path is not immediately installed. It is only considered for programming if/when one of the currently programmed paths is removed or withdrawn.

Echo Function (Controlled Return Path)

Echo mode S-BFD is supported for both PCC-initiated and PCE-initiated SR-TE LSPs and MPLS-based SR Policies. Enabling echo mode requires the addition of a **return-path-label** to the BFD configuration, followed by a numeric value [32...1048512] representing the MPLS label that the reflector should use for the return path. The **return-path-label** command is entered as follows:

- For PCC-initiated SR-TE LSPs the **return-path-label** command is specified under the **bfd** context of the SR-TE LSP.

- For PCE-initiated SR-TE LSPs the **return-path-label** command is specified under the **bfd** context of the **lsp-template** (*Output 13-9*).
- For MPLS-based SR Policies, the return-path-label command is specified within the maintenance-policy (*Output 13-12*).

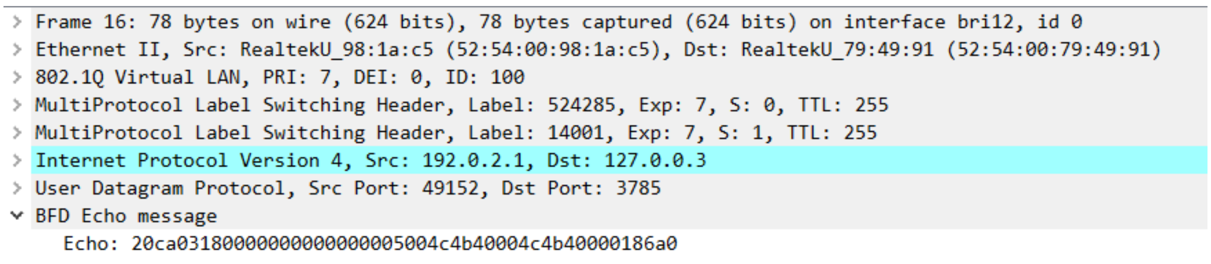
An example of controlled return path using Echo mode s-BFD for a PCC-initiated SR-TE LSP is shown in *Output 13-21* for an LSP from R1 to R3. The **return-path-label** specified at R1 is 14001, which is its own Node-SID. This instructs the reflector R3 to use this label when returning BFD echo packets.

Output 13-21: SR-TE LSP with Echo Mode

```
        mpls {
            lsp "R1-to-R3" {
                bfd {
                    bfd-liveness true
                    bfd-template "sbfd-template"
                    failure-action failover-or-down
                    return-path-label 14001
                }
            }
        }
```

Figure 13-6 shows a packet capture of an S-BFD echo packet on an SR-TE LSP from R1 towards R3 taken on the link R1-R2. The label stack on the packet is {524285, 14001} from top to bottom, where 524284 is the SR-TE LSP label stack representing R2's Adj-SID towards R3, and label 14001 is bottom of stack which will be the active label at R3. It represents the label R3 should use to return the packet towards R1. The source IP address is R1's system address, while the destination address is 127.0.0.3. The destination port is the well-known port for BFD echo, port 3785. Unlike the contents of the S-BFD control packets used in asynchronous mode *(Figure 13-4*) which are well-structured and standard, the contents of the S-BFD echo packet contain originator specific content. This is simply because the reflector does not need to process and parse the packet – it simply label-switches the packet back towards the initiator.

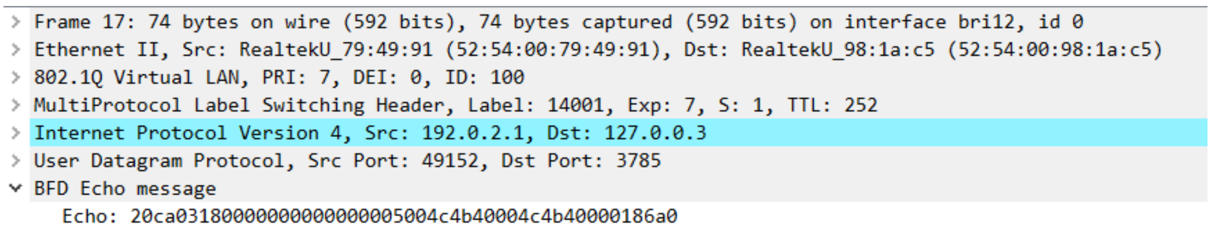
Figure 13-6: S-BFD in Echo Mode from R1 to R3



```
> Frame 16: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_98:1a:c5 (52:54:00:98:1a:c5), Dst: RealtekU_79:49:91 (52:54:00:79:49:91)
> 802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 524285, Exp: 7, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 14001, Exp: 7, S: 1, TTL: 255
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.3
> User Datagram Protocol, Src Port: 49152, Dst Port: 3785
v BFD Echo message
    Echo: 20ca03180000000000000005004c4b40004c4b40000186a0
```

Figure 13-7 shows the return echo packet from R3 to R1. The packet is encapsulated in MPLS with a single label of 14001 representing R1's Node-SID (and the value specified in the **return-path-label** at R1 and encapsulated in the label stack of the echo packet from R1 to R3). Because the echo packet is simply label-switched by R3 without control plane processing, the source/destination IP addresses and ports are not inverted or changed but remain exactly the same as that originated by R1. At first glance it looks strange to see a packet with a source IP address of R1's system address (192.0.2.1) being forwarded to R1, but this the nature of echo mode.

Figure 13-7: S-BFD in Echo Mode from R3 to R1



```
> Frame 17: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_79:49:91 (52:54:00:79:49:91), Dst: RealtekU_98:1a:c5 (52:54:00:98:1a:c5)
> 802.1Q Virtual LAN, PRI: 7, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 14001, Exp: 7, S: 1, TTL: 252
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.3
> User Datagram Protocol, Src Port: 49152, Dst Port: 3785
v BFD Echo message
    Echo: 20ca031800000000000000005004c4b40004c4b40000186a0
```

14 General Topics

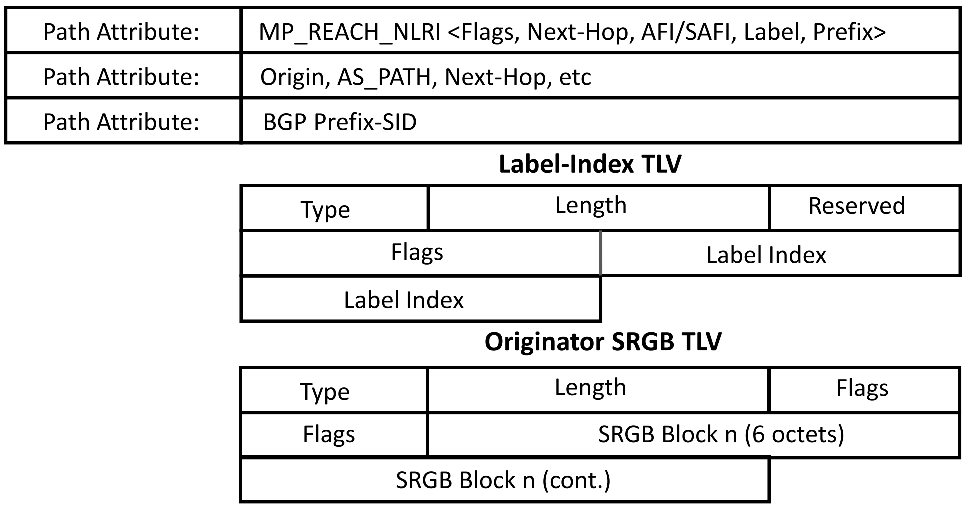
So far throughout this document the use-cases for applicability of SR have been divided into chapter-length chunks. However, there are a number of applications for SR that need to be covered but do not warrant a dedicated chapter to describe their purpose and implementation. They are therefore covered within this general topics chapter.

BGP Prefix-SID

A BGP Prefix Segment is a BGP prefix with a Prefix-SID attached. It is a globally unique SID within the SR domain and identifies an instruction to forward SR-encapsulated packets over the ECMP-aware best path computed by BGP to the prefix. BGP Prefix-SID allows Prefix-SIDs to be advertised across domains, or in environments where there is no IGP present, which is becoming more common in large data centres.

To carry the Prefix-SID in BGP, a Prefix-SID attribute is defined [58] that can be attached to prefixes in (Multiprotocol BGP) IPv4 and IPv6 Labeled Unicast NLRI [67]. In SR-OS, the Prefix-SID attribute is currently supported for the Labeled IPv4 Unicast address family. The BGP Prefix-SID attribute is an optional, transitive path attribute that currently defines two TLVs, the Label-Index TLV and the Originator SRGB TLV.

Figure 14-1: BGP Prefix-SID Attribute Label-Index TLV

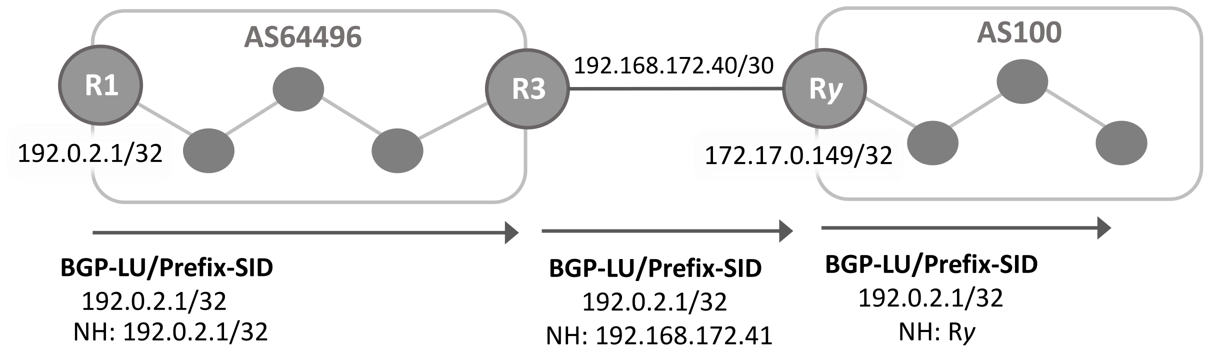


The Label-Index TLV encodes a 32-bit label from the SRGB and must be present in the BGP Prefix-SID attribute attached for IPv4/IPv6 Labeled Unicast prefixes. The Label Index field defines the assigned label as an offset from the SRGB start label, and there are currently no flags defined. A BGP Prefix-SID attribute received without the Label-Index TLV is considered invalid. The Originator SRGB TLV is an optional TLV and contains the SRGB of the router originating the prefix to which the BGP Prefix-SID is attached. The SRGB field is 6 octets and contains 3 octets specifying the first label in the range followed by 3 octets specifying the number of labels in the range. The SRGB field may appear multiple times in an Originator

SRGB TLV if the SRGB consists of multiple ranges that are concatenated. Like the Label-Index TLV there are currently no flags defined.

To demonstrate the use of BGP Prefix-SID the simple Inter-Domain topology illustrated in *Figure 14-2* is used. The intent is to exchange BGP Prefix-SIDs between AS64496 and AS100 and will focus on the advertisement of R1's system address 192.0.2.1/32 from AS64496 to AS100, and Ry's system address 172.17.0.149/32 from AS100 to AS64496 using labeled unicast BGP with associated BGP Prefix-SIDs.

Figure 14-2: Inter-Domain SR-MPLS with BGP Prefix-SID



To enable processing of BGP Prefix-SIDs **segment-routing** first needs to be enabled in the base BGP context, and some or all of the SRGB allocated to BGP. The **prefix-sid-range** context allows for allocation of all of the SRGB to BGP Prefix-SIDs using the **global** parameter, or it can use the **start-label** and **max-index** parameters to constrain BGP Prefix-SIDs to a subset of the SRGB. The range defined by the **start-label** and **max-index** parameters must reside within the global SRGB for the command to be successful. They also need to be unique to avoid conflict or overlap with other SID allocations such as IGP-based Node-SIDs. In this example the SRGB within the test setup is 12000-19999, of which the range 12000-19499 is allocated to IS-IS, hence the remaining subset 19500-19999 of the SRGB is allocated to BGP. The **prefix-sid-range** command also defines the label block that is advertised in the Originator SRGB TLV of the BGP Prefix-SID attribute. Once the label block is assigned, **segment-routing** can be put into an **admin-state** of **enable**.

Output 14-1: Segment Routing for BGP

```
    router "Base" {
        bgp {
            segment-routing {
                admin-state enable
                prefix-sid-range {
                    start-label 19500
                    max-index 499
                }
            }
        }
    }
```

Labeled unicast BGP routes are retained in a separate RIB from unlabeled BGP routes. Non-labeled unicast BGP routes are imported into the labeled unicast BGP RIB using the route-policy framework and the BGP **rib-management** context, and a separate RIB exists for Label-

IPv4 and Label-IPv6. *Output 14-2* shows the configuration at router R1 to import its system IP address 192.0.2.1/32 to the Label-IPv4 RIB with a Prefix-SID attached. The **policy-statement** is fairly generic but notably an **sr-label-index** is used as an **action** output in the policy and can have the value 1-524287. The value defined by the **sr-label-index** is advertised in the Label-Index TLV of the BGP Prefix-SID attribute. The **sr-label-index** command has an optional **prefer-igp** argument. If this argument is selected and the BGP **segment-routing prefix-sid-range** is **global**, the system will initially try to use the IGP SR label index for the IGP route matched by the policy entry. If the IGP route does not have a SID index, or **prefer-igp** argument is not selected, or the **prefix-sid-range** is not **global**, the value specified by the **sr-label-index** is used.

Output 14-2: Assigning a Prefix-SID to Labeled Unicast BGP Routes

```
    policy-options {
        prefix-list "system-IP" {
            prefix 192.0.2.1/32 type exact {
            }
        }
        policy-statement "Label-Unicast-Import" {
            entry 10 {
                from {
                    prefix-list ["system-IP"]
                }
                to {
                    protocol {
                        name [bgp-label]
                    }
                }
                action {
                    action-type accept
                    sr-label-index {
                        value 1
                    }
                }
            }
        }
    }
```

Note: By default, SR-OS enforces the uniqueness of a Node-SID between BGP and any existing IGP instance. That is, if a SID is assigned to an interface, that interface and SID cannot be advertised concurrently in an IGP and BGP, as BGP will consider this a conflict and will not advertise the SID. If there is a requirement to use the same Node-SID in an IGP and BGP, a shared SID must be used which is described later in this chapter. When a shared SID is used, BGP will automatically add the Prefix-SID attribute whenever a local interface with Prefix-SID information is redistributed as a labeled unicast BGP route and BGP **segment-routing** is enabled.

The configured **policy-statement** is then applied as a **route-table-import** within the **rib-management** context of BGP and the Prefix-SID attribute is attached to the imported routes as a result of the **sr-label-index** action output within that policy.

Output 14-3: Importing Routes into the Label-IPv4 RIB

```
    router "Base" {
        bgp {
            rib-management {
                label-ipv4 {
                    route-table-import {
                        policy-name "Label-Unicast-Import"
                    }
                }
            }
        }
    }
```

An export policy is then applied at **group** or **neighbor** level to advertise the labeled unicast BGP route. The **export** policy applied at **group** or **neighbor** level can be the same policy as the one used in the **rib-management** context, or it can be a different policy since the Prefix-SID attribute has already been attached to the prefix during the process of import to the Label-IPv4 RIB. The example below uses the same policy for simplicity and applies it to a Route-Reflector peer within AS64496. The peer has the **label-ipv4** address-family enabled to allow the prefix to be advertised.

Although not shown in the output, BGP requires the **advertise-inactive** command to be set to **true** to allow BGP speakers to advertise system IP addresses, or indeed any other addresses known through the IGP. Because of default route preferences, any local or IGP-learnt prefix is installed in the route-table and as the BGP prefix is not preferred it is not advertised to other peers. When set to **true**, the **advertise-inactive** command causes the best BGP route (and only the best route) to be advertised even if it is not the most preferred route within the system for a given destination (an IGP route also exists).

Output 14-4: Exporting Labeled Unicast IPv4 Routes

```
    router "Base" {
        bgp {
            neighbor "192.0.2.10" {
                group "IBGP-Core"
                export {
                    policy ["Label-Unicast-Import"]
                }
            }
            group "IBGP-Core" {
                family {
                    label-ipv4 true
                }
            }
        }
    }
```

The prefix is subsequently advertised by R1. The Prefix-SID attribute is shown on the last line of the output with the Label-Index TLV having a value of 1, and the Originator SRGB TLV showing the first label in the range as 19500 and the number of labels as 499 as configured. The resultant (start_label + index) value of 19501 is shown in the `IPv4 Label` field with a `Label Type` field showing the action associated with that label, which is a pop action.

Output 14-5: Label-IPv4 RIB-OUT at R1

```
A:admin@R1# show router bgp routes 192.0.2.1/32 label-ipv4 hunt
 --- [snip] ---
-------------------------------------------------------------------------------
RIB Out Entries
-------------------------------------------------------------------------------
Network        : 192.0.2.1/32
Nexthop        : 192.0.2.1
Path Id        : None
To             : 192.0.2.10
 --- [snip] ---
IPv4 Label     : 19501                  Label Type     : SR_POP
Lbl Allocation : PER-PREFIX
Origin         : IGP
AS-Path        : No As-Path
Route Tag      : 0
Neighbor-AS    : n/a
Orig Validation: NotFound
Source Class   : 0                      Dest Class     : 0
Prefix SID     : index 1, originator-srgb [19500/500]
```

A received IPv4 labeled unicast BGP route containing a Prefix-SID is installed in the route-table and the tunnel-table and can have ECMP next-hops or a Fast Reroute backup next-hop. They can be used for any service that supports labeled BGP transport. Prior to being used however, a labeled unicast IPv4 route with a Prefix-SID attribute must be resolved in the same way as a labeled unicast IPv4 route without a Prefix-SID attribute. Broadly, there are two cases: non-adjacent next-hops, and adjacent next-hops. Non-adjacent next-hops need to resolve the next-hop of the labeled unicast IPv4 route to a tunnel with an endpoint that corresponds to the same next-hop. This is simply because the route contains a label that would not be understood by the receiving routers adjacent downstream neighbours. Adjacent next-hops that set next-hop-self have no requirement to resolve next-hops to a tunnel. Essentially the BGP Prefix-SID becomes the active/top-of-stack label. This is because the labeled unicast route contains a label (BGP Prefix-SID) that was advertised by the receiving routers downstream neighbour. This type of environment would be typical for inter-domain route exchange where ASBRs peer in EBGP using interface addresses, or an IGP-free Data Centre where spines and leaves peer in EBGP using interface addresses. As AS64496 also has SR for IS-IS enabled it is selected to resolve the labeled BGP routes, and this is enabled in the BGP **next-hop-resolution** context with a **resolution-filter** as shown in *Output 14-6*.

Output 14-6: Next-Hop Resolution for Labeled BGP Routes

```
    router "Base" {
        bgp {
            next-hop-resolution {
                labeled-routes {
                    transport-tunnel {
                        family label-ipv4 {
                            resolution filter
                            resolution-filter {
                                sr-isis true
                            }
                        }
                    }
                }
            }
        }
    }
```

Note: A device like a control-plane-only Route-Reflector that is not in the data-path can set **route-table-install** to **false** and **next-hop-resolution** for **labeled-routes** to **rr-use-route-table true** to resolve BGP next-hops.

Router R3 is configured to peer in IBGP with the AS64496 Route-Reflector, and in EBGP with router Ry in AS100. When a prefix is learned through labeled unicast IPv4 BGP, it is possible to add a Prefix-SID or to modify the received Prefix-SID value using an **import** policy before re-advertising the same route to other peers. This might be useful to avoid label conflicts where an external domain advertises Prefix-SID values that are already in-use within the internal domain. It is again accomplished using the same **sr-label-index** command as an **action** output, but as the route is already populated in the Label-IPv4 RIB no **rib-management** configuration is required. To illustrate how this is achieved R3 modifies the Prefix-SID value received from R1 before advertising the same prefix to Ry in AS100. Recall that R1 uses index 1, which with an SRGB start-label of 19500 yields a label value of 19501. Using the policy shown in *Output 14-7* R3 modifies the received label index value to 401.

Output 14-7: Modifying a Received Prefix-SID Value

```
    policy-options {
        prefix-list "R1" {
            prefix 192.0.2.1/32 type exact {
            }
        }
         policy-statement "Prefix-SID-Rewrite" {
            entry 10 {
                from {
                    prefix-list ["R1"]
                }
                to {
                    protocol {
                        name [bgp-label]
                    }
                }
                action {
                    action-type accept
                    sr-label-index {
                        value 401
                    }
                }
            }
        }
```

Adding or modifying a Prefix-SID to a received prefix must be implemented as an **import** policy. The above policy is therefore applied on R3's IBGP session towards the AS64496 Route-Reflector.

Output 14-8: Applying the Import Policy for Prefix-SID Modification

```
    router "Base" {
        bgp {
            neighbor "192.0.2.10" {
                group "IBGP-Core"
                import {
                    policy ["Prefix-SID-Rewrite"]
                }
            }
        }
    }
```

The pertinent parts of the prefix advertised to Ry are shown in *Output 14-9*. The next-hop is the interface address of the EBGP peering session between R3 and Ry, and as the peers are adjacent next-hops there is no requirement to resolve next-hops to a tunnel. The Prefix-SID contains the modified Prefix-SID label-index of 401, which with an SRGB start-label of 19500 gives a label value of 19901. The `Label Type` field shows the programmed action at R3, which is an SR label swap.

If required, the BGP Prefix-SID can be prevented from propagating outside of an SR domain using the **block-prefix-sid** command at BGP, group, or neighbour level. This command will remove the Prefix-SID attribute from all routes sent and received to and from the peers included in the scope of the command. Without it, the attribute will be propagated unconstrained.

Output 14-9: BGP Prefix SID Advertised to Ry in AS100

```
A:admin@R3# show router bgp routes 192.0.2.1/32 label-ipv4 hunt | match "RIB Out" post-
lines 24
RIB Out Entries
-------------------------------------------------------------------------------
Network        : 192.0.2.1/32
Nexthop        : 192.168.172.41
Path Id        : None
To             : 192.168.172.42
Res. Nexthop   : n/a
 --- [snip] ---
IPv4 Label     : 19901                  Label Type     : SR_SWAP
Lbl Allocation : PER-PREFIX
 --- [snip] ---
Prefix SID     : index 401, originator-srgb [19500/500]
```

To allow for bidirectional connectivity router Ry advertises a labeled unicast BGP route for its prefix 172.17.0.149 to R3, which is advertised internally in AS64496. The Prefix-SID attribute contains a label-index of 149 and an originator SRGB of 95000, which equates to an IPv4 label value of 19649. Router R1 resolves the BGP next-hop to an SR IS-IS tunnel.

Output 14-10: Ry Prefix with Prefix-SID Received at R1

```
A:admin@R1# show router bgp routes 172.17.0.149/32 label-ipv4 detail

===============================================================================
BGP LABEL-IPV4 Routes
===============================================================================
Original Attributes

Network        : 172.17.0.149/32
Nexthop        : 192.0.2.3
Path Id        : None
From           : 192.0.2.10
Res. Nexthop   : 192.0.2.3 (ISIS Tunnel)
Local Pref.    : 100                    Interface Name : NotAvailable
 --- [snip] ---
IPv4 Label     : 19649
Flags          : Used Valid Best IGP Group-Best In-TTM In-RTM
Route Source   : Internal
AS-Path        : 100
```

```
Route Tag       : 0
Neighbor-AS     : 100
Orig Validation: NotFound
Source Class    : 0                     Dest Class     : 0
Add Paths Send : Default
RIB Priority    : Normal
Last Modified   : 00h03m00s
Prefix SID      : index 149, originator-srgb [95000/500]
```

As the labeled unicast BGP prefix for Ry is resolved and valid, R1 installs a route-table entry and a tunnel-table entry for it. This may now be used for BGP and services as required.

Output 14-11: R1's Tunnel-Table for Ry

```
A:admin@R1# show router tunnel-table 172.17.0.149/32

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
172.17.0.149/32       bgp       MPLS  262151    12     192.0.2.3      1000
-------------------------------------------------------------------------------
```

Note: A labeled unicast route carrying a BGP Prefix-SID cannot appear as the top segment in an SR Policy otherwise the policy will not be correctly resolved.

When using BGP Prefix-SIDs there is a need to be mindful of SID conflicts. When the label-index of an advertised BGP Prefix-SID is combined with the SRGB start-label to derive a local label value that label may have already been allocated to another application, hence a conflict exists. Any existing conflicts will be shown in the CLI output of the labeled BGP route attribute flags field as a `ConflictSID`. SR-OS will address these type of conflicts in the following manner:

– If the conflict is with another BGP route for another prefix, all of the conflicting BGP routes are advertised with a normal labeled unicast BGP label taken from the dynamic label range.
– If the conflict is with an IGP route and BGP is not attempting to redistribute that IGP prefix as a labeled unicast IPv4 or IPv6 prefix containing an **sr-label-index** action set to **prefer-igp true**, the IGP route takes priority and the BGP route is advertised with a normal labeled unicast BGP label from the dynamic range.
– If the conflict is with an IGP route, and BGP is attempting to redistribute that IGP route as a labeled unicast IPv4 or IPv6 prefix with a route-policy that has the **sr-label-index** action set to **prefer-igp true**, this is not considered a conflict and BGP uses the IGP-advertised label to derive the BGP label-index value. This effectively stitches the BGP SR tunnel to the IGP SR tunnel.

BGP Egress Peer Engineering

BGP Egress Peer Engineering (EPE) allows operators to influence how traffic is routed between their Autonomous System (AS) boundary and external BGP peers. The egress border router where the BGP-EPE traffic steering functionality is implemented is called a BGP-EPE-enabled border router. The BGP-EPE-enabled router within the domain advertises specific MPLS labels for specific BGP peers or specific adjacencies to BGP peers, which are programmed as a label-switch to implicit-null such that any allocated labels are popped before forwarding traffic towards a given peer. Ingress routers then select the appropriate BGP-EPE-enabled router and peer/adjacency based on the desired exit point, and forward traffic towards the destination with the appropriate MPLS label stack.

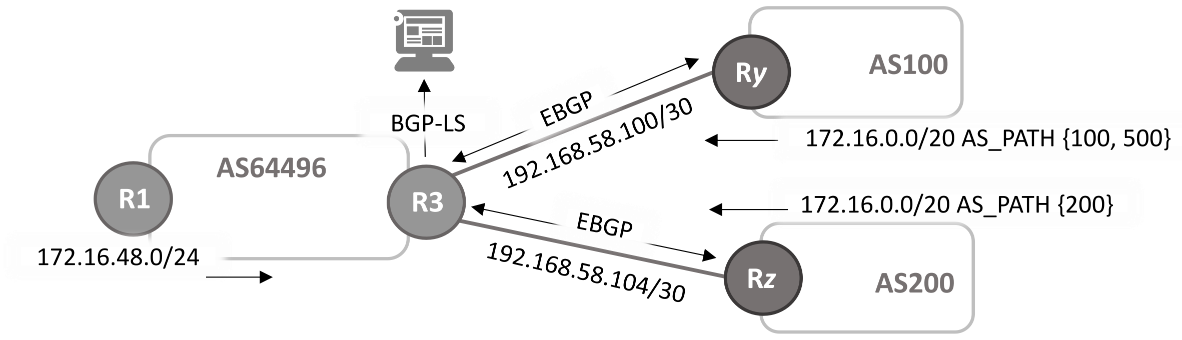
There are two methods through which BGP EPE can be achieved, a centralised approach and a distributed approach:

- The centralised approach is very much SR centric. The BGP-EPE-enabled router assigns peer Node-SIDs and peer Adj-SIDs in the form of MPLS labels, and then advertises those SIDs in BGP-LS such that they can be learned by a centralised controller. The centralised controller is thereafter responsible for programming the relevant SR label stack at the ingress router(s) to influence the exit point for a given prefix or prefixes.
- In the distributed approach the BGP-EPE-enabled router assigns MPLS labels to BGP peers and adjacencies and advertises those prefixes and associated labels into its own AS using labeled unicast BGP (BGP-LU). Ingress routers then use those prefixes to resolve Next-Hops of external prefixes using a recursive lookup to BGP.

This section covers both approaches. Although the distributed approach using BGP-LU does not use SR, the rationale for covering it is predominantly so that the reader can make a direct comparison between the two approaches with their associated advantages and disadvantages. In addition, it may be that while an external prefix learnt in BGP resolves to a BGP-LU route advertised by a BGP-EPE-enabled router, that BGP-LU may in turn resolve to SR.

To illustrate the use of BGP-EPE the topology in *Figure 14-3* is used for both the centralised and distributed approaches. Router R3 belongs to AS64496 and peers externally with Ry in AS100 and Rz in AS200, both of which advertise prefix 172.16.0.0/20. Router Ry advertises prefix 172.16.0.0/20 with AS_PATH {100, 500}, and Rz advertises prefix 172.16.0.0/20 with AS_PATH {200}. As a result of the shorter AS_PATH length, router R3 prefers the path via Rz and AS200, but the operator of AS64496 wants traffic for prefix 172.16.0.0/20 to be steered via router Ry in AS100.

Figure 14-3: BGP-EPE Test Topology



The basic configuration at R3 for external peering with Ry and Rz is not shown within this section for clarity, but it is worth stating that the IP interfaces towards those routers need to be base router interfaces in order to correctly install the label switch to implicit null. They cannot belong to an IES or VPRN service. Other than that, it is a very basic BGP peering configuration.

Centralised BGP-EPE

The centralised approach to BGP-EPE [65] allows operators to extend the use of SR's source-routing capabilities beyond the Autonomous System (AS) boundary and influence how traffic is forwarded towards directly attached BGP peers. It allows a centralised controller to program any egress peer policy at ingress border routers or at hosts within the domain. The centralised controller is called the BGP-EPE controller. As previously outlined, the egress border router where the BGP-EPE traffic steering functionality is implemented is called a BGP-EPE-enabled border router. The input policy programmed at an ingress border router or at a source is called a BGP-EPE policy.

A BGP-EPE-enabled router has a number of BGP peering segments (or BGP peering SIDs) that enable a source-routed inter-domain path to be realised. There are three variants, the PeerNode SID, PeerAdj SID, and the PeerSet SID. The PeerNode SID is a segment describing a peer. The PeerAdj SID is a segment describing a link to a peer. A peerSet SID is a segment describing a link or a node that is part of a set. This would typically be used to represent multiple adjacencies to the same peer.

An ingress border router of an AS may then impose a list of segments to steer a flow along a selected path within the AS towards a selected egress border router, and then through a specific peer. At a minimum, a BGP-EPE policy consists of two segments; the Node-SID of the selected egress border router followed by the BGP peering segment for the chosen peer or peering interface. To enable a centralised EPE controller to compute the path of the BGP-EPE policy, the BGP-EPE-enabled router advertises the BGP peering SIDs in BGP-LS towards the BGP-EPE controller. Thereafter the BGP-EPE controller is able to program source-routes from the ingress router to external peers using BGP SR Policy, PCEP/SR-TE LSPs, Netconf, or static configuration.

For example, consider a router R1 in AS1 that peers externally with router Rx and Ry in AS2, and router Rz in AS3. R1 has single-hop EBGP peering with Rx and Ry, and a multihop EBGP session to Rz as there are two links between R1 and Rz over which traffic is load-balanced. Router R1 allocates the following BGP peering segments:

- A peerNode segment for each of Rx, Ry, and Rz (Rx: 1001, Ry: 1002, Rz: 1003)
- A PeerAdj segment for each recursing interface to a multihop peer (link 1 to Rz: 1050, link 2 to Rz: 1051)
- A PeerSet segment to the set of peers Rx and Ry (Rx/Ry: 1091)

Router R1's Node-SID within AS1 is 5001. Some examples of BGP-EPE policies that could be used by ingress routers within AS1 to influence traffic flows are as follows:

- Prefer all traffic for <Prefix/Length> to use router Rx in AS2: {5001, 1001}
- Prefer all traffic for <Prefix/Length> to use any router in AS2: {5001, 1091}
- Prefer all traffic for <Prefix/Length> to load-balance on any link to Rz in AS3: {5001, 1003}
- Prefer all traffic for <Prefix/Length> to use link 1 to Rz in AS3: {5001, 1050}
- Prefer all egress traffic to use weighted load-balancing between AS2 and AS3, with AS2 receiving 80% of egress traffic and AS3 receiving 20% of egress traffic: Segment-list 1 {5001, 1091, weight 8}, Segment-List 2 {5001, 1003, weight 2}.

The criteria for traffic steering at the ingress router can be coarse or granular. Coarse traffic steering might for example include all egress traffic. Granular traffic steering may match selective IP prefixes for steering, in which case the Color Extended Community could be added by the egress router and advertised internally into the AS.

SR-OS supports EPE with support for the PeerNode SID defining an EBGP/IBGP peer, and the PeerAdj SID defining a particular interface towards a peer. The PeerNode SID and PeerAdj SID labels have the same forwarding semantics as the Adj-SID as they install an ILM operation with a label swap to implicit-null.

Output 14-12 shows the necessary configuration to enable centralised BGP-EPE at R3. The command **egress-peering-engineering** is configured under the **bgp** context and put into an **admin-state** of **enable** to trigger the router to advertise PeerNode and PeerAdj SIDs in BGP-LS towards a controller. The **egress-engineering** command under each group or neighbour context triggers the allocation of SID values for those peers. Labels for PeerNode and PeerAdj SIDs are dynamically allocated without the requirement for an SRLB.

Output 14-12: Centralised BGP-EPE Configuration at R3

```
    router "Base" {
        bgp {
            egress-peer-engineering {
                admin-state enable
            }
            group "External-AS100" {
                egress-engineering {
                    admin-state enable
                }
            }
            group "External-AS200" {
                egress-engineering {
                    admin-state enable
                }
            }
            neighbor "192.168.58.102" {
                group "External-AS100"
                }
            }
            neighbor "192.168.58.106" {
                group "External-AS200"
            }
        }
    }
```

The resulting PeerNode SIDs and PeerAdj SIDs are shown in *Output 14-13*. PeerNode SIDs (labels) 524258 and 524256 together with PeerAdj SIDs 524259 and 524257 are allocated to Rx (AS100) and Ry (AS200) respectively. All allocated labels are programmed with an out label of 3 indicating a switch to implicit-null. If multiple PeerAdj SIDs to the same peer exist, SR-OS supports ECMP by default.

Output 14-13: Allocated PeerNode and PeerAdj SIDs

```
A:admin@R3# tools dump router segment-routing tunnel
===============================================================================
Legend: (B) - Backup Next-hop for Fast Re-Route
        (D) - Duplicate
label stack is ordered from top-most to bottom-most
===============================================================================
------------------------------------------------------------------------------+
 Prefix                                                                       |
 Sid-Type        Fwd-Type       In-Label  Prot-Inst(algoId)                   |
                 Next Hop(s)                         Out-Label(s) Interface/Tunnel-ID |
------------------------------------------------------------------------------+
 --- [snip] ---
 Node            Orig/Transit   524256    BGP-EPE-0
                 192.168.58.106                           3          External-AS200
 Adjacency       Transit        524257    BGP-EPE-0
                 192.168.58.106                           3          External-AS200
 Node            Orig/Transit   524258    BGP-EPE-0
                 192.168.58.102                           3          External-AS100
 Adjacency       Transit        524259    BGP-EPE-0
                 192.168.58.102                           3          External-AS100
```

A new BGP Link State Protocol ID (BGP) and new Node/Link Descriptor TLVs are introduced [66] to allow BGP-LS Link NLRI to represent the BGP peering and associated peering segments (PeerNode SID, PeerAdj SID). *Output 14-14* shows the PeerNode and PeerAdj SIDs advertised in BGP-LS by R3 for the BGP peer to Ry in AS100, truncated to show only the relevant parts. The first entry is the PeerAdj SID with label 524259, while the second entry is the PeerNode SID with label 524258. If the next-hop to a given peer goes down, the associated ILM entry for the peering SID(s) will be removed and the associated BGP-LS update withdrawn. If a BGP neighbour goes down, the system will withdraw BGP-LS Link NLRI advertising the peer SIDs. However, the same SID values are retained locally and are then re-advertised when the neighbour comes back up. Allocated SID values for BGP peers are not persistent.

Output 14-14: BGP Peering Segments Advertised in BGP-LS by R3

```
A:admin@R3# show router bgp routes bgp-ls hunt
 --- [snip] ---
Network:
 Type          : LINK-NLRI
 Protocol      : BGP                    Identifier     : 0x0
 Local Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  BGP Router Id      : 192.0.2.3
 Remote Node descriptor:
  Autonomous System  : 0.0.0.100
  Link State Id      : 0
  BGP Router Id      : 172.29.153.112
 Link descriptor:
  Local ID           : 7
  Remote ID          : 0
  IPV4 Interface Addr: 192.168.58.101
  IPV4 Neighbor Addr : 192.168.58.102
--- [snip] ---
-------------------------------------------------------------------------------
Link State Attribute TLVs :
 Peer Adjacency Segment Identifier (Adj-SID) :  flags 0xc0 weight 0 label
524259
-------------------------------------------------------------------------------

Network:
 Type          : LINK-NLRI
 Protocol      : BGP                    Identifier     : 0x0
 Local Node descriptor:
  Autonomous System  : 0.0.251.240
  Link State Id      : 0
  BGP Router Id      : 192.0.2.3
 Remote Node descriptor:
  Autonomous System  : 0.0.0.100
  Link State Id      : 0
  BGP Router Id      : 5172.29.153.112
 Link descriptor:
  IPV4 Interface Addr: 192.168.58.101
  IPV4 Neighbor Addr : 192.168.58.102
 --- [snip] ---
-------------------------------------------------------------------------------
Link State Attribute TLVs :
 Peer Node Segment Identifier (Node-SID) :  flags 0xc0 weight 0 label 524258
-------------------------------------------------------------------------------
```

The controller is used to push an SR policy to router R1. The color of the policy is 900 and the endpoint is 192.0.2.3 (R3). There are two segments in the segment-list, segment 1 is R3's Node-SID 14003, while segment 2 is the PeerNode SID to Ry in AS100. The policy is resolved-up and is therefore active and operational.

Note: Although not an issue in this test topology, note that a BGP-EPE label cannot appear as the top segment in an SR Policy otherwise the policy will not be correctly resolved.

Output 14-15: SR Policy for BGP-EPE

```
A:admin@R1# show router segment-routing sr-policies bgp end-point 192.0.2.3 color 900

===============================================================================
SR-Policies Path
===============================================================================
-------------------------------------------------------------------------------
Active          : Yes                   Owner           : bgp
Color           : 900
Head            : 0.0.0.0               Endpoint Addr   : 192.0.2.3
RD              : 34567                 Preference      : 100
BSID            : 20002
```

```
TunnelId         : 917521                Age              : 10
Origin ASN       : 64496                 Origin           : 192.0.2.254
NumReEval        : 0                     ReEvalReason     : none
NumActPathChange: 0                      Last Change      : 12/16/2021 13:59:43
Maintenance Policy: N/A

Path Segment Lists:
Segment-List     : 1                     Weight           : 1
S-BFD State      : Down                  S-BFD Transitio*: 0
Num Segments     : 2                     Last Change      : 11/29/2021 12:47:59
  Seg 1 Label    : 14003                 State            : resolved-up
  Seg 2 Label    : 524258                State            : N/A
```

R3 attaches a Color Extended Community of 01:900 to the prefix 172.16.0.0/20 when advertising the prefix internally into AS64496. As the Color advertised with the prefix matches that of the SR Policy, and the next-hop of the prefix matches the SR Policy endpoint, R1 is able to resolve the prefix to the SR policy. Although not shown for conciseness, this resolution of the IPv4 prefix does require that BGP **next-hop-resolution** for IPv4 **shortcut-tunnels** is enabled with a **resolution-filter** that permits **sr-policy**.

Output 14-16: Prefix 172.16.0.0/20 Resolved to SR-Policy

```
A:admin@R1# show router route-table 172.16.0.0/20

===============================================================================
Route Table (Router: Base)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age        Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
172.16.0.0/20                                 Remote  BGP       01h19m40s  170
       192.0.2.3 (tunneled:SR-Policy:917521)                        0
-------------------------------------------------------------------------------
```

As a result of the BGP best path selection algorithm at R3, it will prefer the path to 172.16.0.0/20 through Rz in AS200 due to the shorter AS_PATH length. However, the use of this SR Policy will override that and steer traffic to that destination through Ry in AS100. This can easily be validated by initiating a ping from R1's advertised prefix 172.31.48.0/24 towards 172.16.0.0/20 and confirming that the packet count on the interface to AS100 is incrementing accordingly.

Output 14-17: Incrementing Packet Statistics Towards Ry in AS100

```
*A:R3# show router interface "External-AS100" statistics | match expression "Tx Pkts|Tx
V4 Pkts"
Tx Pkts           : 5755                 Tx Bytes          : 490676
Tx V4 Pkts        : 5755                 Tx V4 Bytes       : 490676
Mpls Tx Pkts      : 0                    Mpls Tx Bytes     : 0
```

Note: It's worth noting that the term 'centralised approach' is used rather loosely. In this section the BGP-EPE policy applied at the ingress router is programmed by a centralised controller, but it is equally possible that a statically configured SR policy at the ingress router could program exactly the same parameters as a centralised controller. It's worth noting however that peer Node-SIDs and Adj-SIDs are not persistent, and a static policy would not react to a change of SID.

Distributed BGP-EPE

At the time of writing this feature is supported only on FP-based platforms of FP3 generation or better. It is not currently supported on 7250 IXR generation one or two platforms. Please check Release Notes for an updated list of 7250 IXR unsupported features.

The distributed approach to BGP-EPE [70] relies on BGP-LU and recursive resolution of BGP next-hops to BGP. Typically when an egress router advertises routes received from an external peer into its own domain in IBGP it sets the next-hop to itself to allow ingress routers in the domain to resolve the next-hop to an IGP route (or MPLS tunnel). With distributed BGP-EPE, routes received from external peers have their next-hop unchanged when advertising into IBGP. The BGP-EPE-enabled egress router then advertises BGP-LU routes for the adjacent peers interface address (for single-hop EBGP) or loopback address (for multi-hop EBGP). This allows ingress routers in the domain to resolve next-hops for BGP routes to BGP-LU routes.

For example, consider a hypothetical router R1 in AS1 that peers externally with router Rx in AS2, Ry in AS3, and Rz in AS4. R1 has single-hop EBGP peering with Rx and Ry, and a multihop EBGP session to Rz as there are two links between R1 and Rz over which traffic is load-balanced. The IP addresses of those peers are as shown in *Table 14-1*:

Table 14-1: IP Addressing

Node	Loopback	Link Addresses
R1	192.0.2.1/32	N/A
Rx	192.0.2.100/32	R1 (192.168.1.1/30) to Rx (192.168.1.2/30)
Ry	192.0.2.101/32	R1 (192.168.1.5/30) to Ry (192.168.1.6/30)
Rz	192.0.2.102/32	R1 (192.168.1.9/30) to Rz (192.168.1.10/30) link 1
		R1 (192.168.1.13/30) to Rz (192.168.1.14/30) link 2

For single-hop EBGP with Rx and Ry, the following BGP routes are advertised:

- Rx advertises an IPv4 BGP route {172.16.0.0/20} to R1 with a BGP next-hop of 192.168.1.2. When R1 advertises this route into AS1 in IBGP it does not modify the next-hop address, and simply leaves it unchanged.
- For reachability to Rx, R1 advertises a BGP-LU route {192.168.1.2, label 1001} into AS1 with a BGP next-hop of 192.0.2.1. R1 programs an MPLS forwarding state of pop and forward to 192.168.1.2 for this advertised label.
- Ry advertises an IPv4 BGP route {172.16.0.0/20} to R1 with a BGP next-hop of 192.168.1.6. When R1 advertises this route into AS1 in IBGP it does not modify the next-hop address, and simply leaves it unchanged.

- For reachability to Ry, R1 advertises a BGP-LU route {192.168.1.6, label 1002} into AS1 with a BGP next-hop of 192.0.2.1. R1 programs an MPLS forwarding state of pop and forward to 192.168.1.6 for this advertised label.

For multi-hop EBGP with Rz, the following BGP routes are advertised:

- Rz advertises an IPv4 BGP route {172.16.0.0/20} to R1 with a BGP next-hop of 192.0.2.102. When R1 advertises this route into AS1 in IBGP it does not modify the next-hop address, but simply leaves it unchanged. Note that R1 has static routes to 192.0.2.102/32 with next-hops pointing to each of the remote IP addresses of link 1 (192.168.1.10) and link 2 (192.168.1.14) to enable load-balancing over those underlying links. Note that the IBGP routers in AS1 should support the BGP Add-Paths extension so that R1 can advertise all of the paths to this BGP next-hop (i.e. link 1 and link 2).
- For reachability to Rz link 1, R1 advertises a BGP-LU route {192.0.2.102/32, label 1003} with a BGP next-hop of 192.0.2.1. R1 appends the Add-Path Path-ID 1 to different this route from the link 2 route advertisement, and programs an MPLS forwarding state of pop and forward to 192.168.1.10 for this advertised label.
- For reachability to Rz link 2, R1 advertises a BGP-LU route (192.0.2.102/32, label 1004} with a BGP next-hop of 192.0.2.1. R1 appends the Add-Path Path-ID 2 to different this route from the link 1 route advertisement, and programs an MPLS forwarding state of pop and forward to 192.168.1.14 for this advertised label.
- For load-balancing towards Rz, R1 advertises a BGP-LU route {192.0.2.102/32, label 1005} with a BGP next-hop of 192.0.2.1. R1 appends the Add-Path Path-ID 3 to different this route from the link 1 route and link 2 route advertisements, and programs an MPLS forwarding state of pop and load-balance to {192.168.1.10, 192.168.1.14} for this advertised label.

For simplicity, assume AS1 is running SR, and router R1's Node-SID is 5001. Some examples of BGP-EPE policies that could be used by ingress routers within AS1 to influence traffic flows are as follows:

- Prefer all traffic for <172.16.0.0/20> to use router Rx in AS2: {5001, 1001}
- Prefer all traffic for <172.16.0.0/20> to use router Ry in AS3: {5001, 1002}
- Prefer all traffic for <172.16.0.0/20> to use link 2 to Rz in AS4: {5001, 1004}
- Prefer all traffic for <172.16.0.0/20> to load-balance across link 1 and link 2 to Rz in AS4: {5001, 1005}.

When using centralised BGP-EPE, policies are instantiated at ingress routers through a process of defining a segment-list in an SR policy to the desired egress peer, and traffic is subsequently steered into the policy by attaching the relevant Color community to advertised service routes. One of the implications of this approach is that the preferred exit point is explicitly defined with a sequence of segments without any reliance on the BGP best-path selection algorithm at the ingress router. When using distributed BGP-EPE the process is notably different. It is a process of resolving the BGP service route to a BGP-LU route and then recursively resolving the BGP-LU route to the underlying transport technology (which could be SR or LDP for example). In the above example, assuming that Add-Path is present an ingress router would receive three paths for prefix {172.16.0.0/20} via {Rx, AS2}, {Ry, AS3}, and {Rz, AS4}, and each of those paths would have a different next-hop that in turn resolved to a different BGP-LU route with its associated label. When the preferred path is selected, the imposed label stack for that path provides the instructions on how the packet should be forwarded by downstream routers. However, it is still necessary to use conventional mechanisms to influence which path is the preferred path. For example, suppose the BGP best-path selection algorithm at an ingress router selects {Rx, AS2} as the preferred path for prefix {172.16.0.0/20} but the requirement is to forward traffic to {Ry, AS3} then local policy must be used to modify the best-path selection, such as the use of Local Preference. Thereafter, the relevant MPLS label stack imposed by the ingress router will ensure that downstream routers adhere to that decision. In summary, paths for a given destination need to be made visible to an ingress router (through the use of Add-Path), and thereafter local policy is used at that router to modify the outcome of the BGP best-path selection algorithm to the desired exit point.

Referring again to the BGP-EPE test topology in *Figure 14-3*, the necessary configuration to enable distributed BGP-EPE at R3 is shown in *Output 14-18*. The **egress-peer-engineering-label-unicast true** command can be enabled at either the BGP group or neighbour level and will trigger the generation of a BGP-LU route for the /32 or /128 prefix belonging the EPE peer. It is enabled both towards Ry in AS100 (192.168.58.102) and Rz in AS200 (192.168.58.106) at group level. Note that **add-paths** is enabled for both the **ipv4** and **label-ipv4** address families. It is required for **ipv4** to ensure that R3 advertises prefix 172.16.0.0/20 from both next-hops (Ry and Rz) each with different Path-IDs, therefore allowing maximum path visibility within AS64496. It is required for **label-ipv4** when multihop EBGP is used to allow the egress router to advertise the BGP peering address multiple times for each underlying link used to reach that peer (each with a different label). It is therefore not explicitly required in the test topology as multihop EBGP is not in use but is included for completeness.

Output 14-18: Distributed BGP-EPE Configuration at R3

```
    router "Base" {
        bgp {
            add-paths {
                ipv4 {
                    send 2
                    receive true
                }
                label-ipv4 {
                    send 2
                    receive true
                }
            }
            group "External-AS100" {
                egress-peer-engineering-label-unicast true
            }
            group "External-AS200" {
                egress-peer-engineering-label-unicast true
            }
            neighbor "192.168.58.102" {
                group "External-AS100"
            }
            neighbor "192.168.58.106" {
                group "External-AS200"
            }
        }
    }
```

Output 14-19 shows the prefix IPv4 BGP prefix 172.16.0.0/20 advertised by R3 into AS64496 in IBGP. The first entry for prefix 172.16.0.0/20 is with next-hop 192.168.58.102 (Ry), AS_PATH {100, 500}, and Path Id 2. The second entry for prefix 172.16.0.0/20 is with next-hop 192.168.58.106 (Rz), AS_PATH {200}, and Path Id 1. Without the presence of Add-Paths, R3 would have only advertised a single path with a next-hop of 192.168.58.106 (Rz) owing to the shorter AS_PATH length. Add-Paths ensures that both paths are advertised as unique NLRI by appending the Path Id, which avoids path suppression and increases path visibility within AS64496.

Output 14-19: Prefix 172.16.0.0/20 Advertised by R3 into IBGP

```
*A:R3# show router bgp routes 172.16.0.0/20 hunt | match expression "Network|Nexthop|Path
Id|AS-Path"
Network        : 172.16.0.0/20
Nexthop        : 192.168.58.102
Path Id        : 2
Res. Nexthop   : n/a
AS-Path        : 100 500
Network        : 172.16.0.0/20
Nexthop        : 192.168.58.106
Path Id        : 1
Res. Nexthop   : n/a
AS-Path        : 200
```

Output 14-20 shows the BGP-LU routes advertised by R3, truncated to show only the attributes pertinent to BGP-EPE. Network 192.168.58.102/32 is peer Ry in AS100 and is advertised with label value 524262 and Path Id 2, while network 192.168.58.106/32 is peer Rz in AS200 and is advertised with label value 524263 and Path Id 1. (The Path Id's used here have no relationship to the Path ID's used with the IPv4 route above. Since they are entirely different address families, the Path ID's can simply be re-used.) Both routes have a next-hop of R3 (192.0.2.3) and a label allocation of BGP-EPE programmed with a swap action (to implicit-null).

Output 14-20: BGP-LU Routes for Ry and Rz Advertised by R3

```
*A:R3# show router bgp routes label-ipv4 hunt | match expression "Network|Nexthop|Path
Id|IPv4 Label|Lbl Allocation"
Network        : 192.168.58.102/32
Nexthop        : 192.0.2.3
Path Id        : 2
Res. Nexthop   : n/a
IPv4 Label     : 524262                 Label Type     : SWAP
Lbl Allocation : BGP-EPE
Network        : 192.168.58.106/32
Nexthop        : 192.0.2.3
Path Id        : 1
Res. Nexthop   : n/a
IPv4 Label     : 524263                 Label Type     : SWAP
Lbl Allocation : BGP-EPE
```

Ingress routers in AS64496 will receive both unlabeled BGP routes (172.16.0.0/20) and labeled BGP routes (192.168.58.102/32 and 192.168.58.106/32). It is therefore necessary to ensure that the BGP **next-hop-resolution** is correctly configured for both families to allow for the recursive next-hop resolution:

- Unlabeled BGP routes (in this case **ipv4**) require a **shortcut-tunnel** with a **resolution-filter** that allows **bgp**. In other words, unlabeled BGP routes can resolve to (labeled) BGP routes.
- Labeled routes (in this case **label-ipv4**) require a **transport-tunnel** with a **resolution-filter** that uses the underlying MPLS transport to reach the egress BGP-EPE-enabled router. In this example, the underlying MPLS transport protocol **sr-isis**. In other words, BGP-LU routes resolve to existing MPLS tunnels between ingress and egress routers within AS64496.

With the relevant BGP next-hop-resolution in place, *Output 14-21* shows the tunnel-table entries at R1 for the BGP tunnels to Ry in AS100 (192.168.58.102/32) and Rz in AS200 (192.168.58.106/32). Both entries have a next-hop of R3 (192.0.2.3). These tunnels will be used to resolve unlabeled BGP routes with the associated next-hops.

Output 14-21: BGP Tunnel-Table Entries at R1

```
A:admin@R1# show router tunnel-table protocol bgp

===============================================================================
IPv4 Tunnel Table (Router: Base)
===============================================================================
Destination           Owner     Encap TunnelId  Pref   Nexthop        Metric
   Color
-------------------------------------------------------------------------------
192.168.58.102/32     bgp       MPLS  262165    12     192.0.2.3      1000
192.168.58.106/32     bgp       MPLS  262164    12     192.0.2.3      1000
-------------------------------------------------------------------------------
```

Due to the presence of Add-Path in AS64496, router R1 will receive two paths for prefix 172.16.0.0/20, one from external peer Ry with AS_PATH {100, 500}, and one from external peer Rz with AS_PATH {200}. As a result of the shorter AS_PATH, the BGP best-path selection algorithm at R1 will prefer the path to Rz, but the requirement is to send traffic for prefix 172.16.0.0/20 to be steered via Ry in AS100. The policy in *Output 14-22* is therefore applied as an import policy at R1. The policy simply matches the prefix 172.16.0.0/20 with next-hop address 192.168.58.102 (Ry) and sets the Local Preference attribute to a value of 200. As this is a higher Local Preference than the default 100, the path via Ry will be preferred.

Output 14-22: Import Policy at R1

```
    policy-options {
        prefix-list "External-172.16/20" {
            prefix 172.16.0.0/20 type exact {
            }
        }
        policy-statement "EPE-Prefix-172.16/20" {
            entry 10 {
                from {
                    prefix-list ["External-172.16/20"]
                    next-hop {
                        ip-address 192.168.58.102
                    }
                }
                action {
                    action-type accept
                    local-preference 200
                }
            }
        }
    }
```

Output 14-23 shows the BGP paths for prefix 172.16.0.0/20 at R1 after the import policy has been applied, and as can be seen the path via Ry in AS100 (192.168.58.102) with Local Preference 200 is valid, used, and best. When forwarding traffic towards this destination prefix router R1 will impose the label stack of {Node-SID-R3, BGP-LU-label-Ry} from top to bottom. The Node-SID-R3 will ensure traffic simply follows the shortest IGP path through AS64496 towards R3. At R3 that label will be popped to expose the BGP-LU-label-Ry, which will again be popped and forwarded towards Ry in AS100 without an IP lookup.

Output 14-23: BGP Routes for Prefix 172.16.0.0/20 at R1

```
A:admin@R1# show router bgp routes 172.16.0.0/20
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP IPv4 Routes
===============================================================================
Flag  Network                                            LocalPref   MED
      Nexthop (Router)                                   Path-Id     IGP Cost
      As-Path                                                        Label
-------------------------------------------------------------------------------
u*>i  172.16.0.0/20                                      200         None
      192.168.58.102                                     54          1000
      100 500                                                        -
*i    172.16.0.0/20                                      100         None
      192.168.58.106                                     53          1000
      200                                                            -
-------------------------------------------------------------------------------
Routes : 2
```

The forwarding path from R1 to prefix 172.16.0.0/20 can be easily validated by initiating a ping from R1's advertised prefix 172.31.48.0/24 towards 172.16.0.0/20 and confirming that the packet count on the interface to AS100 is incrementing accordingly.

Output 14-24: Incrementing Packet Statistics Towards Ry in AS100

```
*A:R3# show router interface "External-AS100" statistics | match expression "Tx Pkts|Tx
V4 Pkts"
Tx Pkts          : 118                   Tx Bytes          : 12103
Tx V4 Pkts       : 118                   Tx V4 Bytes       : 12103
Mpls Tx Pkts     : 0                     Mpls Tx Bytes     : 0
```

Statically Assigned Adjacency-SIDs

The default behaviour is to automatically assign Adj-SID labels from the dynamic label range, although Adj-SIDs labels can be statically assigned for individual adjacencies if required. When Adj-SIDs are assigned from the dynamic label range they are non-persistent and may change values in the event of a system restart. However, when Adj-SIDs are statically assigned they are persistent and will survive a system restart. Once a static Adj-SID is assigned to an interface the default dynamic Adj-SID is deleted.

If static Adj-SIDs are used, the assigned values can be either locally unique to an SR router or globally unique. In theory, the use of globally unique Adj-SID values could potentially reduce the size of the label stack that a headend needs to impose to target a remote Adj-SID. For example, if locally unique Adj-SID values are used then in order to target a remote Adj-SID would require that the headend imposes the Node-SID of the remote node followed by the Adj-SID of the targeted link on that node. If globally unique values are used, there is theoretically no requirement for the Node-SID to be imposed and just the Adj-SID would suffice since it is globally unique. Unfortunately, this is not something that is currently supported in SR-OS when SR-MPLS is used. However, even when locally unique Adj-SID values are used there are still some potential benefits of using statically assigned Adj-SIDs. For example, assume that an SR Policy is instantiated on a headend by another router (possibly the egress router) and that SR Policy targets a remote Adj-SID. If at some later point in time the router owning that remote Adj-SID undergoes a restart that value may very likely change if the dynamic label range is used. If it does change, that SR Policy could potentially become a traffic black-hole without some other form of failure detection mechanism, or traffic could simply be mis-forwarded. If statically assigned Adj-SIDs are used, the interruption would be temporary for the period of the restart. Another potential advantage is that operational communities can become familiar with values assigned to given interfaces if a well-defined structure is used for assignment.

Statically assigned Adj-SIDs are taken from a locally-configured Segment Routing Local Block (SRLB). In SR-MPLS SRLBs are reserved label blocks used for specific local purposes such as SR Policy BSIDs, Adjacency Set SIDs, and assignment of static Adjacency SIDs. A dedicated SRLB is required per-application and has local significance only. Since the SRLB has local significance, the same range could be used on all SR routers in the domain for operational ease. Ranges for each SRLB are taken from the dynamic label range. Initially the SR Local Block is created using the **reserved-label-block** command within the **mpls-labels** context. Within the context of that local block, the **start-label** and **end-label** define the range of the block.

Output 14-25: Creation of SR Local Block for Static Adj-SID Assignment

```
    router "Base" {
        mpls-labels {
            reserved-label-block "Adj-SID" {
                start-label 24000
                end-label 24999
            }
        }
    }
```

Once the SRLB is defined it is dedicated to the specific application, which in this case is static Adj-SIDs that are owned and advertised by the relevant IGP instance. The SRLB is therefore assigned to the **segment-routing** context within that IGP instance using the **srlb** command. Once the **srlb** is assigned, static values are configured under each {**isis|ospf**} interface using the **ipv4-adjacency-sid label** command.

Output 14-26: Assignment of SRLB and Static Adj-SID

```
    router "Base" {
        isis 0 {
            segment-routing {
                srlb "Adj-SID"
            }
            interface "link-to-R2" {
                ipv4-adjacency-sid {
                    label 24102
                }
            }
        }
    }
```

Output 14-27 shows the advertised IS-IS LSP after the static Adj-SID value has been assigned. It is truncated to show only the Adj-SID Sub-TLV for the link on which the static value was used. There are two Adj-SID Sub-TLVs, one for IPv4 and one for IPv6. As the static value was only assigned to the IPv4 Adj-SID, the IPv6 Adj-SID remains allocated from the dynamic label range. The Adj-SID Sub-TLV for IPv4 however, shows the value 24102 as configured. It also has the 'P' flag set to indicate that this Adj-SID value is persistent.

Output 14-27: Advertised Adj-Sid Sub-TLV with Static Value

```
A:admin@R1# show router isis database R1.00-00 level 2 detail
---- [snip] ----
TLVs :
---- [snip] ----
  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 149
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
---- [snip] ----
    Adj-SID: Flags:v4BVLP Weight:0 Label:24102
    Adj-SID: Flags:v6BVL Weight:0 Label:524282
---- [snip] ----
```

Dual Protected/Unprotected Adjacency SIDs

By default SR-OS advertises Adj-SIDs as unprotected until **loopfree-alternate** is enabled under the IGP, after which SR-OS advertises Adj-SIDs as protected (B-flag set). There may be instances however, where an operator wishes to offer protection to some SR-TE LSPs but not to others. That is, upon failure of a given link, SR-TE LSPs with protection are restored and routed away from the failure while LSPs without protection simply remain down until either the link is restored or the associated LSP **retry-timer** expires (default, after which the system will attempt to re-establish the LSP via an alternate unprotected path.

Support for dual protected/unprotected Adj-SIDs is enabled using the **allocate-dual-sids** command within the **segment-routing adjacency-sid** context. If a protected or unprotected adjacency is already configured, an additional dynamic protected, or unprotected Adj-SID is instantiated and advertised for the interface.

Output 14-28: Protected/Unprotected Dual-SIDs

```
    router "Base" {
        isis 0 {
            segment-routing {
                adjacency-sid {
                    allocate-dual-sids true
                }
            }
        }
    }
```

With the above configuration applied *Output 14-29* shows the resultant IS-Neighbours TLV. As the interface is dual-stack, there are four Adj-SIDs advertised, two for IPv4 and two for IPv6. The first two entries have the F-flag (Address-Family) unset denoting they are IPv4. The last two entries have the F-flag set denoting they are IPv6. For each address family, one entry has the B-flag (Backup) set indicating it is protected, and one entry does not, indicating that it is unprotected. Note that because the allocate-dual-sids command is enabled at IGP level, protected and unprotected Adj-SIDs will be advertised for every adjacency. *Output 14-29* merely provides an example of one.

Output 14-29: IS Neighbour TLV with Dual Adj-SIDs

```
A:admin@R1# show router isis database R1.00-00 detail level 2

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

--- [SNIP] ---

  TE IS Nbrs   :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 157
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
    MaxLink BW: 10000000 kbps
    Resvble BW: 10000000 kbps
    Unresvd BW:
        BW[0] : 10000000 kbps
        BW[1] : 10000000 kbps
        BW[2] : 10000000 kbps
        BW[3] : 10000000 kbps
        BW[4] : 10000000 kbps
        BW[5] : 10000000 kbps
        BW[6] : 10000000 kbps
        BW[7] : 10000000 kbps
    Admin Grp : 0x0
    TE Metric : 100
    Adj-SID: Flags:v4BVL Weight:0 Label:524267
    Adj-SID: Flags:v4VL Weight:0 Label:524283
    Adj-SID: Flags:v6BVL Weight:0 Label:524266
    Adj-SID: Flags:v6VL Weight:0 Label:524282
```

To validate the use of unprotected Adj-SIDs, the following topology is used, with all routers belonging to a single IS-IS Level 2 area. System addresses are 192.0.2.x/24, where ‘x’ is denoted by the router number (for example, R1 is 192.0.2.1, R2 is 192.0.2.2, etc). All routers have TI-LFA enabled and are advertising both protected and unprotected Adj-SIDs.

```
R1--------R2--------R3
 |          |          |
 R4--------R5-------R6
<------IS-IS  L2 ------>
```

An SR-TE LSP is created between R1 and R6 that has a **path-computation-method** of **local-cspf** and is configured with a **local-sr-protection** set to **none**, meaning the system will calculate the path using only unprotected adjacencies (excluding protected adjacencies) and will fail to compute the CSPF if it cannot do so. Once the LSP is established it takes the path R1-R4-R5-R6 as shown in *Output 14-30*, where the path is shown as transiting 192.0.2.4 (R4) and 192.0.2.5 (R5) before reaching the destination 192.0.2.6 (R6).

Output 14-30: Path for R1-to-R6 SR-TE LSP

```
*A:R1# show router mpls sr-te-lsp "R1-to-R6" path detail | match expression "Admin
State|PathCompMethod|LocalSrProt|Actual Hops|A-SID"

Admin State       : Up                      Oper State        : Up
PathCompMethod    : local-cspf              OperPathCompMethod: local-cspf
LocalSrProt       : none                    Oper LocalSrProt  : none
Actual Hops       :
    192.168.0.10(192.0.2.4)(A-SID)                Record Label        : 524274
 -> 192.168.0.22(192.0.2.5)(A-SID)                Record Label        : 524264
 -> 192.168.0.26(192.0.2.6)(A-SID)                Record Label        : 524262
```

Each hop of the path is computed using unprotected Adj-SIDs, and this can be validated by reconciling the Adj-SID labels associated with each hop of the SR-TE LSP path above with the Adj-SIDs advertised by each hop. This is shown in *Output 14-31*, which consists of three separate CLI outputs. The first output shows R1’s advertised Adj-SIDs for its IS Neighbour R4 and shows that the unprotected Adj-SID with B-flag unset is 524274, which reconciles with the first hop of the SR-TE LSP path. The second output shows R4’s advertised Adj-SIDs for its IS Neighbour R5 and shows that the unprotected Adj-SID with B-flag unset is 524264, which reconciles with the second hop of the SR-TE LSP path. The third and final output shows R5’s advertised Adj-SIDs for its IS Neighbour R6 and shows that the unprotected Adj-SID with B-flag unset is 524262, which again reconciles with the final hop of the SR-TE LSP path. In short, all of the hops of the SR-TE LSP are computed with unprotected Adj-SIDs.

Output 14-31: Dual-SIDs Advertised by R1, R4, and R5

```
*A:R1# show router isis database R1.00-00 level 2 detail | match expression "TE IS Nbrs|Nbr
|Flags:v4"
  TE IS Nbrs   :
    Nbr   : R4.00
    Adj-SID: Flags:v4BVL Weight:0 Label:524276
    Adj-SID: Flags:v4VL Weight:0 Label:524274

*A:R1# show router isis database R4.00-00 level 2 detail | match expression "TE IS Nbrs|Nbr
|Flags:v4"
  TE IS Nbrs   :
    Nbr   : R5.00
    Adj-SID: Flags:v4BVL Weight:0 Label:524279
    Adj-SID: Flags:v4VL Weight:0 Label:524264

*A:R1# show router isis database R5.00-00 level 2 detail | match expression "TE IS Nbrs|Nbr
|Flags:v4"
  TE IS Nbrs   :
    Nbr   : R6.00
    Adj-SID: Flags:v4BVL Weight:0 Label:524274
    Adj-SID: Flags:v4VL Weight:0 Label:524262
```

With the SR-TE LSP from R1 to R6 routed along the path R1-R4-R5-R6, the R1-R4 link is failed at the R4 side by disabling the port. The sequence of events is captured at R1 and the pertinent parts of it are shown in *Debug 14-1*. The events within this output can be broadly summarised as follows:

- The interface to “link-to-R4” is declared down at 16:43:06 after failure of BFD and IS-IS (events 1512 and 1513).
- At 16:43:21 the LSP R1-to-R6 is set to operationally down as it uses an unprotected Adj-SID and the LSP retry-timer is started. The next attempt to re-establish the LSP will be made in 28 seconds (events 1520, 1525, and 1528).
- At 16:43:48 the retry timer expires, and a retry is attempted. A CSPF is requested (events 1529, 1530, and 1531).
- At 16:43:49 the CSPF is returned that routes the LSP along the path R1-R2-R5-R6. The SR-TE LSP is declared operationally up (events 1610, 1613, 1615, and 1619).

Debug 14-1: Failure Sequence for Unprotected SR-TE LSP from R1 to R6

```
1512 2022/08/09 16:43:06.163 BST minor: VRTR #2061 Base 192.168.0.10
BFD: Local Discriminator 2 BFD session on node 192.168.0.10 is down due to noHeartBeat

1513 2022/08/09 16:43:06.163 BST warning: ISIS #2045 Base VR:  1 ISIS (0) Adjacency state
Adjacency  status  changed  to  down  for  interface:  link-to-R4,  for  level:  l2,  LSP-id:
1920.0000.2004.00-00

1520 2022/08/09 16:43:21.223 BST minor: DEBUG #2001 Base MPLS
MPLS: LSP
Set operational state for LSP R1-to-R6 to Down, previous state is Up

1525 2022/08/09 16:43:21.223 BST warning: MPLS #2010 Base VR 1:
LSP R1-to-R6 is operationally disabled ('shutdown') because noPathIsOperational

1528 2022/08/09 16:43:21.223 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Started retry timer for LspPath R1-to-R6::empty(LspId 62978). Next retry attempt will be
after 28sec.

1529 2022/08/09 16:43:48.973 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Retry timer expired for LspPath R1-to-R6::empty(LspId 62978)

1530 2022/08/09 16:43:48.973 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
```

```
Initiate setup for LspPath R1-to-R6::empty(LspId 62978) (Retry attempt 1)

1531 2022/08/09 16:43:48.973 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
Create CSPF request for LspPath R1-to-R6::empty(LspId 62978)

1610 2022/08/09 16:43:48.973 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
CSPF Path returned by TE for LspPath R1-to-R6::empty(LspId 62978)
From 192.0.2.1  To 192.0.2.6
Hop 1 -> Label 524281 IpAddr 192.168.0.2 LinkId 0 UpstreamIpAddr 192.168.0.1 UpstreamLinkId
0 RtrAddr 192.0.2.2 Strict
Hop  2  ->  Label  524265  IpAddr  192.168.0.14  LinkId  0  UpstreamIpAddr  192.168.0.13
UpstreamLinkId 0 RtrAddr 192.0.2.5 Strict
Hop  3  ->  Label  524262  IpAddr  192.168.0.26  LinkId  0  UpstreamIpAddr  192.168.0.25
UpstreamLinkId 0 RtrAddr 192.0.2.6 Strict

1613 2022/08/09 16:43:49.153 BST minor: DEBUG #2001 Base MPLS
MPLS: SR-TE
SR-TE tunnel is up for LspPath R1-to-R6::empty(LspId 62978)

1615 2022/08/09 16:43:49.153 BST minor: DEBUG #2001 Base MPLS
MPLS: LSP Path
Set operational  state  for LspPath R1-to-R6::empty(LspId 62978)  to Up,  previous  state  is
Down

1619 2022/08/09 16:43:49.153 BST minor: DEBUG #2001 Base MPLS
MPLS: LSP
Set operational state for LSP R1-to-R6 to Up, previous state is Down
```

Finally, the operational state of the R1-to-R6 SR-TE LSP and its post-recovery path can be displayed. Here the path is shown as transiting 192.0.2.2 (R2) and 192.0.2.5 (R5) before reaching the destination 192.0.2.6 (R6).

Output 14-32: R1-to-R6 SR-TE LSP After R1-R4 Link Failure

```
*A:R1#  show  router  mpls  sr-te-lsp  "R1-to-R6"  path  detail  |  match  expression  "Admin
State|PathCompMethod|LocalSrProt|Actual Hops|A-SID"

Admin State       : Up                      Oper State        : Up
PathCompMethod    : local-cspf              OperPathCompMethod: local-cspf
LocalSrProt       : none                    Oper LocalSrProt  : none
Actual Hops       :
    192.168.0.2(192.0.2.2)(A-SID)                 Record Label        : 524281
 -> 192.168.0.14(192.0.2.5)(A-SID)                Record Label        : 524265
 -> 192.168.0.26(192.0.2.6)(A-SID)                Record Label        : 524262
```

Note: As described in this sub-section SR-TE LSPs that use unprotected Adj-SIDs will not be protected upon failure and will remain in an operationally down state until the LSP **retry-timer** expires, after which the system will attempt to re-establish the LSP. If however **sr-te-resignal resignal-on-igp-event** is enabled, and the failure is visible to the headend the LSP will attempt to re-establish immediately.

Adjacency Sets

An adjacency set is a bundle of two or more adjacencies grouped together to form a set that is represented by a common Adj-SID. When traffic arrives at a router with the Adj-Set SID present, the router will load-balance traffic across the member links of the adjacency set.

An Adjacency Set can be either parallel or non-parallel. A parallel adjacency set is where the links forming the bundle originate on one node and terminate on another common node. With non-parallel links the originating node bundles links of the adjacency set together, but the far end of those links may terminate on different nodes.

When adjacency sets are used a common Adj-SID is advertised for every link in the set, sometimes referred to as the Adj-Set SID, and another Adj-SID is advertised for each of the adjacencies that forms part of the set. A weight factor can be associated with the Adj-SID advertised with each adjacency. The weight is used to enable a weighted load-balancing function. For example, if a node advertises Adj-SID 1000 for link 1 with weight 1, and Adj-SID 1000 for link 2 with weight 2 then packets arriving with Adj-SID 1000 will be load-balanced on a 1:2 ratio between link 1 and link 2. SR-OS does not currently support weighted load-balancing for adjacency sets.

Adj-SIDs used for adjacency sets must come from a locally-configured Segment Routing Local Block (SRLB). To avoid repetition the SRLB '*Adj-SID*' configured in *Output 14-25* and assigned to **segment-routing** in *Output 14-26* is used to demonstrate the configuration steps to enable the creation of an adjacency set. The **adjacency-set** is created within the same **segment-routing** context and is assigned an integer value in the range 1-4294967295. Within that context options exist to define the characteristics of the adjacency set. SR-OS supports either IPv4 or IPv6 adjacency sets, not dual-stack. The **family** command is therefore used to select the appropriate address family, with the default being IPv4. If an interface is dual-stack then an adjacency set can be created for each address family. The **parallel** command is used to indicate whether the adjacency set is parallel or non-parallel, with the default being **parallel** set to **true**. The **advertise** command is used to determine if the adjacency set is advertised into the IGP. Only parallel adjacency sets can be advertised into the IGP. When non-parallel adjacency sets are used the set is not advertised into the IGP because the use of load-balancing over the adjacency set could potentially result in packets being forwarded to a downstream router with label below the adjacency set label being a label that the downstream router did not assign. Hence the use of non-parallel links requires a level of coordination as to what packets are forwarded to what neighbour to ensure that this does not happen. When parallel adjacency sets are used the default for **advertise** is **true**. One final command in the **adjacency-set** context, not shown in the example below, is the **sid** command. By default, Adj-Set SIDs are allocated from the dynamic label range. The **sid** command can optionally be used if statically assigned Adj-Set SIDs are required.

Output 14-33: Creation of an Adjacency Set

```
    router "Base" {
        isis 0 {
            segment-routing {
                srlb "Adj-SID"
                adjacency-set 1 {
                    family ipv4
                    parallel true
                    advertise true
                }
            }
        }
```

Once the adjacency set has been configured the member adjacencies forming the set can be assigned to it as illustrated in *Output 14-34*. Note that only point-to-point interfaces can be assigned to an adjacency set, and up to 32 interfaces can form part of the same adjacency set.

Output 14-34: Assignment of Interfaces to the Adjacency Set

```
    router "Base" {
        isis 0 {
            interface "first-link-to-R2" {
                adjacency-set 1 { }
            }
            interface "second-link-to-R2" {
                adjacency-set 1 { }
            }
        }
    }
```

Output 14-35 shows the advertised IS-IS LSP after the two links above have been assigned to the adjacency set. It is truncated to show only the relevant parts of the Extended IS Reachability TLV for the two links. As can be observed from the interface addresses, both links are dual-stack IPv4/IPv6 and as a result each link has an Adj-SID Sub-TLV for IPv4 indicated by `Flags:v4BVL`, and an Adj-SID Sub-TLV for IPv6 indicated by `Flags:v6BVL`. There is also a third Adj-SID Sub-TLV for each link which is the Adj-Set SID and can be identified as such by the presence of the S flag (Set Flag). If OSPF were used, this would be represented by the presence of the G flag (Group Flag).

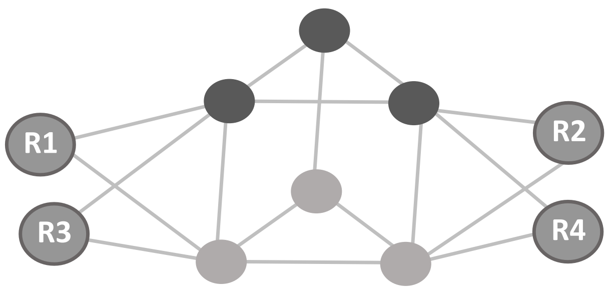
Output 14-35: Advertising Adjacency Sets

```
A:admin@R1# show router isis database R1.00-00 level 2 detail
---- [snip] ----
TLVs :
---- [snip] ----
  TE IS Nbrs    :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 156
    IF Addr   : 192.168.0.1
    IPv6 Addr : 2001:db8:33ac:2200::4:1
    Nbr IP    : 192.168.0.2
    Nbr IPv6  : 2001:db8:33ac:2200::4:2
---- [snip] ----
    Adj-SID: Flags:v4BVL Weight:0 Label:524268
    Adj-SID: Flags:v6BVL Weight:0 Label:524267
    Adj-SID: Flags:v4VLS Weight:0 Label:524265
  TE IS Nbrs    :
    Nbr   : R2.00
    Default Metric  : 100
    Sub TLV Len     : 125
    IF Addr   : 192.168.0.65
    IPv6 Addr : 2001:db8:33ac:2200::4:41
    Nbr IP    : 192.168.0.66
    Nbr IPv6  : 2001:db8:33ac:2200::4:42
---- [snip] ----
    Adj-SID: Flags:v4BVL Weight:0 Label:524276
    Adj-SID: Flags:v6BVL Weight:0 Label:524266
    Adj-SID: Flags:v4VLS Weight:0 Label:524265
---- [snip] ---
```

Anycast SID

An Anycast segment or Anycast-SID enforces the ECMP-aware shortest-path forwarding towards the closest node of the anycast set. It does not reference a particular node but rather references all nodes belonging to that set, and all nodes of that set advertise the same prefix with the same Prefix-SID value. SR routers thereafter resolve the Anycast Prefix-SID to the closest of the routers in the Anycast set advertising the prefix using IGP distance. The use of Anycast SID can be useful in numerous scenarios, such as providing a level of disjointness, or providing a redundancy mechanism. Consider the example in *Figure 14-4*, which depicts a dual-plane core topology. The routers in the upper-plane advertise prefix 192.0.2.251/32 with Prefix-SID 1051, while the routers in the lower-plane advertise prefix 192.0.2.252/32 with Prefix-SID 1052. R2's Node-SID is 200, while R4's Node-SID is 400. A customer of the network operator requires diversity between two sites, one from R1 to R2, and a second from R3 to R4. To achieve this, R1 can impose a label stack of {1051, 200} to influence traffic to route over the upper-plane. Equally, R3 can impose a label stack of {1052, 400} to influence traffic to route over the lower plane. Of course, the paths are not strictly disjoint, but rather the example is a best-effort disjointness. If the link between R1 and the top plane should fail for example, traffic from R1 towards R2 would need to route through the bottom plane before being routed back via the top plane.

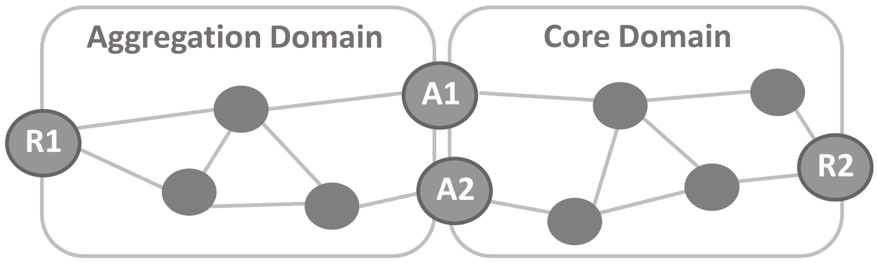
Figure 14-4: Disjointness with Anycast SID



A key point to note when using Anycast SID with SR-MPLS is that the use of absolute SID values (discussed in chapter 2) is important. Router R1 may impose a label stack of {1051, 200} and send a packet to the closest SR-router in the Anycast set of the upper plane, but when the closest SR router of the Anycast set removes the active label (the Anycast SID), the next label in the stack needs to be a label that it advertised and has programmed in its ILM. If indexing is used, this cannot be guaranteed.

Another example of using Anycast SID is for redundancy. For example, assume a case where inter-domain connectivity is required and there are redundant ABRs/ASBRs spanning the two domains. In *Figure 14-5* there is a core domain and an aggregation domain that are distinct IGP instances. There are two ABRs/ASBRs, A1 and A2 that span both IGP instances and participate in both. If A1 and A2 advertise the same prefix and Prefix-SID into each domain, they offer a redundant egress point towards the adjacent domain.

Figure 14-5: Redundancy with Anycast SID



In SR-OS a Prefix-SID can be assigned to the primary address of any network interface that is configured to be a **loopback** interface. *Output 14-36* provides an example of Anycast SID configuration using IS-IS. A **loopback** interface is configured with a **primary** address of 192.0.2.252/32, which is referenced in **isis** as a **passive** interface. A Prefix-SID is then assigned as an **index** or an explicit **label** value. The **clear-n-flag** allows the user to not set the Node-SID flag in the IS-IS or OSPF Prefix-SID sub-TLV. Some implementations of Anycast SID are known to perform a check on received Prefix-SIDs and drop duplicate Prefix-SID sub-TLVs with the Node-SID flag set. SR-OS does not perform such a check and accepts duplicate Prefix-SIDs with the Node-SID flag set or unset and will resolve the SID to the closest IGP owner.

Output 14-36: Anycast SID Configuration

```
    router "Base" {
        interface "Anycast-Loopack" {
            loopback
            ipv4 {
                primary {
                    address 192.0.2.252
                    prefix-length 32
                }
            }
        }
        isis 0 {
            interface "Anycast-Loopack" {
                admin-state enable
                passive true
                ipv4-node-sid {
                    label 14252
                    clear-n-flag true
                }
            }
        }
    }
```

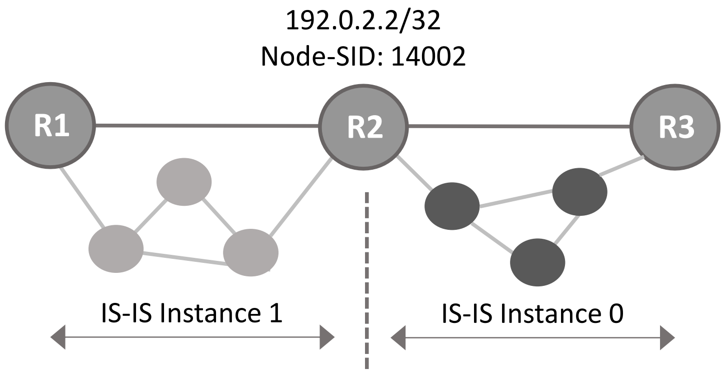
Shared SID

In a seamless-MPLS network domain border routers may participate in multiple instances of IS-IS or OSPF. When enabling SR in this kind of environment and assigning a SID to an interface, SR-OS enforces Node-SID uniqueness amongst IGP instances by default. That is, if a SID is assigned to an interface as described in chapters 3 and 4, that interface and SID cannot be assigned to multiple IGP instances. Routers which participate in multiple IGP instances will include the system interface in multiple areas, and for reasons of operational simplicity it may be preferable to advertise the same IP address and Node-SID into multiple instances. To allow for this SR-OS also supports the concept of a shared SID, which removes the requirement for per-instance SID uniqueness and allows SIDs to be re-used between IGP instances.

The shared SID needs to be shared on the same interface across IGP instances. That is, SID 's' cannot be assigned to interface 'x' in IS-IS and interface 'y' in OSPF, and also needs to be assigned to a common address family (IPv4 or IPv6) within each instance. In addition, a shared SID also has to be either a shared SID or a non-shared SID – it cannot be both. If in one instance a SID is assigned to an interface in a non-shared manner as described chapters 3 and 4, it cannot also be configured as a shared SID in a second instance.

As the most likely application for using shared SID is a seamless MPLS environment, I could make a very elaborate test environment to illustrate its use. However, as shared SID is a very simple concept, I will reciprocate with the simple topology shown in *Figure 14-6*. Router R2 participates in IS-IS instance 1 towards R1, and also in IS-IS instance 0 towards R3. Both instances are Level 2 and use multi-instance IS-IS [69] using the **standard-multi-instance true** command within each IS-IS instance. SR is enabled on all routers and the SRLB in use is 12000-19999. R2's Node-SID is 14002, and the objective is that this SID will be shared in both IS-IS instances.

Figure 14-6: Shared SID Test Topology



Chapters 3 and 4 described two modes of operation when assigning labels to SR for use with SR-MPLS, the global option and the per-instance option. The global option uses the entire SRGB, whereas the per-instance option divides the SRGB into non-overlapping sub-ranges assigned to each instance. When a SID is shared between instances, all instances must share the same SRLB. In other words, each instance must use the **global** command when assigning labels to SR. *Output 14-37* shows the relevant configuration at router R2 for both IS-IS instance 0 and instance 1.

Output 14-37: Global SRLB Designation at R2

```
    router "Base" {
        isis 0 {
            segment-routing {
                prefix-sid-range {
                    global
                }
            }
        }
        isis 1 {
            segment-routing {
                prefix-sid-range {
                    global
                }
            }
        }
    }
```

To allocate one or more shared SIDs the **segment-routing sr-mpls** context contains a **prefix-sids** sub-context that requires the assignment of a valid and configured interface as a keyword. In *Output 14-38* applied at R2, the system interface (192.0.2.2/32) is used, within which an **ipv4-sid** or **ipv6-sid** can be allocated as either an absolute **label** value or an **index**. Whichever value is used, the Prefix-SID is always advertised in SR-OS as an index value. The **node-sid true** command allows to set or unset the Node-SID flag (N-flag) in the Prefix-SID advertisement. If unset, the interface may be advertised as an Anycast SID. For completeness, *Output 14-38* also shows the configuration of the system interface in both IS-IS instances simply to highlight that neither contains a Node-SID allocation as this is now the function of the **sr-mpls prefix-sid**. If both an **sr-mpls prefix-sid** and a regular IGP **node-sid** are simultaneously configured, the IGP will advertise the IGP **node-sid** value. In other words, the IGP configuration will override the configuration in the **segment-routing sr-mpls** context.

Output 14-38: Shared SID Configuration at R2

```
    router "Base" {
        isis 0 {
            interface "system" {
                passive true
                level-capability 2
            }
        }
        isis 1 {
            interface "system" {
                passive true
                level-capability 2
            }
        }
        segment-routing {
            sr-mpls {
                prefix-sids "system" {
                    node-sid true
                    ipv4-sid {
                        label 14002
                    }
                }
            }
        }
    }
```

Output 14-39 shows R2's IS-IS LSP advertised into instance 0 towards R3, truncated to show its system interface 192.0.2.1/32 within the Extended IP Reachability TLV. The Prefix-SID sub-TLV shows an index of 2002, which with an SRGB start-label of 12000 equates to an absolute label value of 14002 as configured. The flags field indicates that the prefix is a Node-SID, and that PHP is not in use. As the **prefix-attributes-tlv true** command is used within the IS-IS context, the Extended IP Reachability TLV also contains a Prefix Attributes sub-TLV which again has the Node flag (N-flag) set, together with the E-flag to indicate that the router is Entropy Label Capable (ELC).

Output 14-39: IS-IS Instance 0 LSP with Extended IP Reachability TLV

```
A:admin@R2# show router isis 0 database R2.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 0 Database (detail)
===============================================================================

Displaying Level 2 database
-------------------------------------------------------------------------------
 --- [snip] ---
  TE IP Reach   :
    Default Metric  : 0
    Control Info:   S, prefLen 32
    Prefix   : 192.0.2.2
    Sub TLV   :
      AttrFlags: NE
      Prefix-SID Index:2002, Algo:0, Flags:NnP
 --- [snip] ---
```

Similarly, *Output 14-40* shows R2's IS-IS LSP advertised into instance 1 towards R1, again truncated to show its system interface 192.0.2.1/32 within the Extended IP Reachability TLV. As can be observed, the same SID is advertised into both instances and is therefore considered a shared SID.

Output 14-40: ISIS Instance 1 LSP with Extended IP Reachability TLV

```
A:admin@R2# show router isis 1 database R2.00-00 level 2 detail

===============================================================================
Rtr Base ISIS Instance 1 Database (detail)
===============================================================================

Displaying Level 2 database
-------------------------------------------------------------------------------
 --- [snip] ---
  TE IP Reach   :
    Prefix   : 192.0.2.2
    Sub TLV   :
      AttrFlags: NE
      Prefix-SID Index:2002, Algo:0, Flags:NnP
 --- [snip] ---
```

Finally, R2's SR tunnel database shows the advertised SID (14002) as an IGP-Shared SID belonging to the base algorithm (algorithm 0). As expected for a Node-SID the forwarding behaviour is to terminate packets with SID 14002 as the active label. As a result, there is no out-label or outgoing interface entries.

Output 14-41: R2's SR Tunnel Entry for Shared SID

```
*A:R2# tools dump router segment-routing tunnel
===============================================================================
Legend: (B) - Backup Next-hop for Fast Re-Route
        (D) - Duplicate
label stack is ordered from top-most to bottom-most
===============================================================================
-------------------------------------------------------------------------------+
 Prefix                                                                        |
 Sid-Type        Fwd-Type       In-Label  Prot-Inst(algoId)                    |
                 Next Hop(s)                       Out-Label(s) Interface/Tunnel-ID |
-------------------------------------------------------------------------------+
 --- [snip] ---
 192.0.2.2
 Node            Terminating    14002     IGP-Shared-0
 --- [snip] ---
```

Using Entropy Label

Load-balancing is the ability to balance traffic through a network using multiple paths. It is widely deployed by operators because it brings numerous benefits. Most notably, it provides a high level of resilience with pre-programmed backup paths, and it provides interim upgrade steps before moving to the next higher speed link (for example, multiples of 10GigE before moving to 100GigE) which in turn eases capacity planning. In general, there are two techniques used to achieve load-balancing, Equal Cost Multi-Path (ECMP) and Link Aggregation Groups (LAG). ECMP is implemented at the IP layer and means that the IGP shortest cost path between a source and destination node has several equal cost paths. LAG is implemented at the Ethernet layer and allows two or more member links to appear as a single virtual Layer 3 interface. In both cases, packets are load-balanced across the available links on a per-flow basis to avoid jitter and reordering issues for that flow. To make sure that all packets of a given flow are forwarded over the same link, routers use match criteria within the packet headers as input to a hash function.

When the input to the hash function uses Layer 3 and Layer 4 fields such as source and destination IP addresses and port numbers, the input to the hash function is typically very granular and this yields a high variance in hash output with the result that traffic is load-balanced evenly across available links. (This is not always true of course, for example where IP tunnelling such as the GPRS Tunnelling Protocol (GTP), or IPSec is used.) However, this granular hash input information is typically only available to ingress LSRs. In an MPLS network transit LSRs generally only have access to the MPLS label stack for hash input, and frequently this results in poor variance of the hash input with the result that traffic is poorly distributed across the available links.

Quite simply, the more granular the input to the hash function is, the higher the probability of achieving an even distribution across multiple links. The insertion of an Entropy Label [52] at an ingress LSR provides a mechanism to load-balance traffic flows at transit LSRs that only use the MPLS label stack as hash input. Each flow identified at an ingress LSR is assigned a different entropy label value that forms part of the MPLS label stack. It consists of the insertion of an Entropy Label (EL) that is not signalled, preceded by an Entropy Label Indicator (ELI). The purpose of the ELI is to ensure that the egress LSR can unambiguously distinguish between entropy labels and application labels. The ELI uses the special-purpose MPLS label value of 7 and is inserted after the relevant MPLS transport label(s) and before any service labels. The use of entropy label therefore results in two labels being inserted in the MPLS label stack; the ELI followed by the EL. For clarity it's worth stating that the imposition of the EL does not consume any label resources at the ingress LSR.

Due to hardware differences between FP4-based platforms and the Broadcom-based 7250-IXR there are some differences in the process of enabling Entropy Label, as well as the granularity with which it can be applied to services. FP-based platforms provide the ability to support Entropy Label on a per-service basis. The 7250-IXR mandates that Entropy Label is enabled at a system-wide level, and it must be enabled at that level before any Entropy Label configuration can be applied elsewhere. Once applied at the system level, the use of Entropy Label is applicable to all services, with no per-service option to enable or disable it. In addition, the use of Entropy Label on the 7250-IXR platforms is mutually exclusive with DSCP transparency for Layer 3 services. When **entropy-label** is placed into an **admin-state** of **enable** within the **fp options** context, **dscp-transparency** must be placed into an **admin-state** of **disable** before the configuration can be successfully committed. The impact of this is that if **entropy-label** is enabled on an SR tunnel used by a VPRN service for BGP next-hop resolution, DSCP bits for VPRN traffic will not be preserved after MPLS EXP marking at network egress. *Output 14-42* shows the necessary configuration to enable Entropy Label at system level on the 7250-IXR.

Output 14-42: Enabling the use of Entropy Label on 7250-IXR Platforms

```
    system {
        fp {
            options {
                qos {
                    dscp-transparency {
                        admin-state disable
                    }
                }
                mpls {
                    entropy-label {
                        admin-state enable
                    }
                }
            }
        }
    }
```

The process for enabling Entropy Label for shortest path SR tunnels and SR-TE LSPs is common on all SR-OS platforms, however, an ingress LSR cannot insert EL's for packets going into a given label switched path (LSP) unless an egress LSR has indicated that it is capable of processing ELs on that LSP. This can be either through Entropy Label Capability (ELC) signalling or through explicit configuration. Chapters 3 and 4 described how routers indicate their ELC in IS-IS and OSPF for use with SR, and although SR-OS will advertise its

ELC, it does not currently process received ELC advertised through IS-IS or OSPF to determine if an egress LSR is Entropy Label capable. Explicit configuration is therefore required, and this implemented at the headend using the **entropy-label override-tunnel-elc** command within the relevant IGP instance. When set to **true**, this command instructs the router to ignore any ELC advertisements received through IGP advertisements and instead assume that the whole domain supports processing of Entropy Labels.

Output 14-43: Explicit Configuration of ELC

```
    router "Base" {
        isis 0 {
            entropy-label {
                override-tunnel-elc true
            }
        }
    }
```

Once the Entropy Label capability has been configured, the use of Entropy Labels can be enabled within the IGP segment-routing context using the command **entropy-label true**. Once enabled, this command allows the use of Entropy Label on shortest path SR tunnels.

Output 14-44: Enabling EL/ELI Insertion for Shortest Path SR

```
    router "Base" {
        isis 0 {
            segment-routing {
                entropy-label true
            }
        }
    }
```

For SR-TE LSPs, the use of Entropy Labels can be enabled at the base MPLS level. Within the **entropy-label** context when **sr-te** is set to **true** all SR-TE LSPs inherit this setting by default. However, it is equally possible to enable or disable the use of **entropy-label** on a per-LSP basis. Entropy Label behaviour at both levels is user-configurable, but SR-TE LSPs must transition through a down/up state before any applied changes are adopted.

Output 14-45: Enabling EL/ELI Insertion for SR-TE LSPs

```
    router "Base" {
        mpls {
            entropy-label {
                sr-te true
            }
            lsp "R1-to-R3" {
                entropy-label true
            }
        }
    }
```

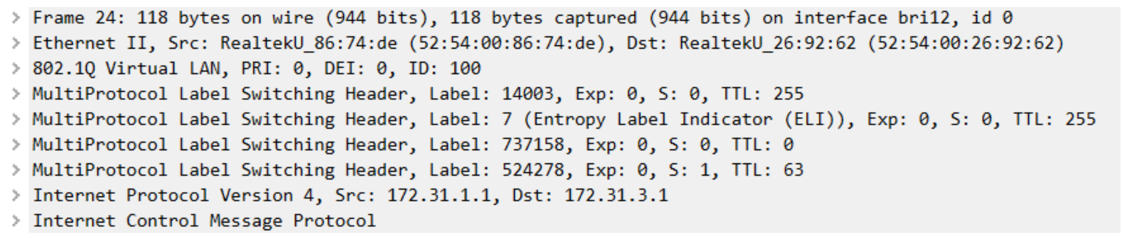
On 7250 IXR platforms the configuration covered to this point is sufficient to enable the use of Entropy Label. For FP-based Nokia platforms insertion of EL/ELI is configured on a per-service basis, therefore the per-service configuration is an additional step. *Output 14-46* shows the required configuration for an Epipe service and a VPRN service. For the Epipe service, **entropy-label** is enabled under the **spoke-sdp** context. For a VPRN service, **entropy-label** is enabled at the top level within the VPRN context.

Output 14-46: Per-Service Entropy Label Configuration with FP-based Platforms

```
    service {
        epipe "two" {
            spoke-sdp 2003:2 {
                entropy-label
            }
        }
        vprn "one" {
            entropy-label true
        }
    }
```

Figure 14-7 shows the insertion of ELI/EL on an SR-ISIS LSP. The top label 14003 represents the Node-SID for the destination, and this is followed by the ELI with label value 7. The next label in the stack is the EL with label value 737158, and finally the service label 524278 is bottom of stack.

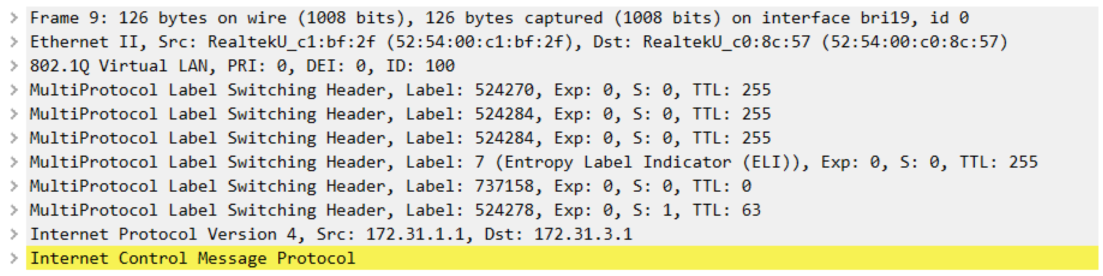
Figure 14-7: ELI/EL with SR-ISIS LSP



```
> Frame 24: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_86:74:de (52:54:00:86:74:de), Dst: RealtekU_26:92:62 (52:54:00:26:92:62)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 14003, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 7 (Entropy Label Indicator (ELI)), Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 737158, Exp: 0, S: 0, TTL: 0
> MultiProtocol Label Switching Header, Label: 524278, Exp: 0, S: 1, TTL: 63
> Internet Protocol Version 4, Src: 172.31.1.1, Dst: 172.31.3.1
> Internet Control Message Protocol
```

Figure 14-8 shows the insertion of ELI/EL on an SR-TE LSP with strict hops using Adj-SIDs. The first three labels (524270, 524284, 524284) are the Adj-SIDs expressing the path. This is followed by the ELI and EL, and finally the service label 524278 is bottom of stack. A point to note with SR-TE LSPs is to ensure that configured maximum label stack is large enough to accommodate the additional two labels. In the example below a label-stack-size of 6 is the minimum requirement.

Figure 14-8: ELI/EL with SR-TE LSP



```
> Frame 9: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_c1:bf:2f (52:54:00:c1:bf:2f), Dst: RealtekU_c0:8c:57 (52:54:00:c0:8c:57)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 524270, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 7 (Entropy Label Indicator (ELI)), Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 737158, Exp: 0, S: 0, TTL: 0
> MultiProtocol Label Switching Header, Label: 524278, Exp: 0, S: 1, TTL: 63
> Internet Protocol Version 4, Src: 172.31.1.1, Dst: 172.31.3.1
> Internet Control Message Protocol
```

SR Statistics and Streaming Telemetry

It is frequently desirable to be able to know the amount of data that is being passed through a given LSP or set of LSPs. This requirement may be triggered by a need to record statistics over a medium-to-long-term period to help with capacity planning and potential changes to a network topology. Alternatively, it may be triggered by a requirement to provide information such as LSP throughput and/or link delay to a centralised controller for the purpose of ensuring that a given LSP remains placed on a path that meets its objectives.

SR-OS provides the ability to generate statistics for IGP SIDs, SR-TE LSPs, and SR Policies, and these statistics can be obtained from CLI output, SNMP MIBs, or YANG state. Typically, these statistics would be pulled from the node with a medium-level frequency, for example every 15 minutes or every hour, but in the case of LSPs or SR Policies that are controlled by a PCE there may a need a requirement to provide these results in a timelier manner. For these SR-TE LSPs and SR Policies the option therefore exists to push the statistics northbound using streaming telemetry.

This section covers the configuration requirements for the generation of statistics and how they can be viewed. It then looks at the use of gRPC streaming telemetry for two use-cases, both of which are relevant to the use of a centralised controller. Firstly, the use of streaming telemetry to provide SR-TE LSP and SR Policy throughput statistics, which enables a centralised controller to ascertain if the programmed path has sufficient capacity. Secondly, although not directly related to SR, the use of streaming telemetry to push single-hop link latency measurements northbound. If the centralised controller knows the measured link latency for every link in the network, it can not only accurately compute shortest path latency results, but also take corrective action should the latency on a given path increase to the point where it needs to be moved onto a new path.

Statistics Collection

At the time of writing the SR statistics covered in this section are supported only on FP-based platforms. Statistics for SR-TE LSPs are supported on 7250 IXR generation two platforms but are not available on 7250 IXR generation one platforms. Please check Release Notes for an updated list of 7250 IXR unsupported features.

When statistics are enabled, the system allocates a statistic index to each entity for which statistics should be collected, which is responsible for maintaining and updating the statistics count. The resources for these statistics indices are not infinite, and equally the allocation of them is non-deterministic in nature. If the system is unable to allocate a statistics index due to resource consumption it raises an alarm and will automatically retry the allocation of that index. If the system is subsequently able to allocate an index a second notification is raised. SR-OS can support statistics for both ingress and egress data path directions. On ingress indices are allocated to each programmed ILM entry for Node-SID or Adj-SID statistics as configured. On egress, indices are allocated to each of the NHLFEs for the received Node-SIDs, and for the NHLFEs corresponding to local and received Adj-SIDs. A single index can be shared between primary and backup paths for a given Node-SID or Adj-SID. If statistics are disabled, any allocated indices are releases and counters cleared.

Unlike RSVP-TE statistics, SR statistics are not FC-aware and simply provide an aggregate throughput in packets and octets. The statistics are accessible through CLI output, SNMP, and are modelled in Nokia YANG state. They cannot however be written to a local accounting file.

IGP SID Statistics

SR-OS provides the ability to generate statistics for Node-SIDs, Adj-SIDs, and Adj-Set SIDs in both the ingress and egress data path directions. Within the **segment-routing** context of the relevant OSPF or IS-IS instance, there are sub-contexts for both **ingress-statistics** and **egress-statistics**. Either or both can be enabled, and both ingress and egress have the option to enable statistics generation for **node-sid**, **adj-sid**, and **adj-set**. In this example, **egress-statistics** are enabled for **node-sid** providing the ability to generate statistics for shortest path SR paths.

Output 14-47: Enabling IGP SID Statistics

```
    router "Base" {
        isis 0 {
            segment-routing {
                egress-statistics {
                    node-sid true
                }
            }
        }
    }
```

An example of the available statistics for Node-SID are shown in *Output 14-48*. As illustrated in the example, arguments exist to display statistics for a selected Node-SID prefix and a Flex-Algorithm identifier. If required, statistics can be cleared using the command “clear router isis sid-egress-stats <sid-type>”.

Output 14-48: Statistics Output for Egress Node-SID

```
A:admin@R1# show router isis sid-stats node ip-prefix-prefix-length 192.0.2.3/32 algo 0

===============================================================================
Rtr Base ISIS Instance 0 Sid Statistics
===============================================================================
Ingress Label      : 14003              Type              : node
Prefix             : 192.0.2.3/32
Algorithm          : 0
Ingress Oper State: disabled            Egress Oper State : enabled
Ingress Octets     : 0                  Egress Octets     : 4153240
Ingress Packets    : 0                  Egress Packets    : 7987

-------------------------------------------------------------------------------
Sid count : 1
```

SR-TE LSP Statistics

SR-TE LSP statistics are enabled at the LSP level of an ingress LSR. If used with primary/secondary paths the system includes statistics for the primary and an aggregate of all paths.

Output 14-49: Enabling Statistics for SR-TE LSPs

```
    router "Base" {
        mpls {
            lsp "R1-to-R3" {
                egress-statistics {
                    admin-state enable
                }
            }
        }
    }
```

An example of the output of the per-LSP statistics is shown below. Again, these statistics can be cleared if required using the command "clear router mpls sr-te-lsp-egress-stats <lsp-name>"

Output 14-50: SR-TE LSP Statistics Output

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R3 egress-stats

===============================================================================
SR-TE LSP Egress Statistics
===============================================================================
-------------------------------------------------------------------------------
LSP Name : R1-to-R3
-------------------------------------------------------------------------------
Admin State       : Up

Path Name         : empty
StatsOperState    : Up
Aggregate Pkts    : 4862                Aggregate Octets : 2528240

Total for all paths
Tot.Aggr Pkts     : 4862                Tot.Aggr Octets  : 2528240
```

The above configuration yields basic aggregate packets and octets, but it is also possible to enable rate-based statistics for SR-TE LSPs to allow for the generation of a packet rate in packets per second and a bit rate in megabits per second. To allow for the generation of rate-based statistics the configuration shown in *Output 14-49* needs to be extended to include an accounting policy and to enable statistics collection, which is shown in *Output 14-51*. The **accounting-policy** must be configured to generate the **combined-mpls-srte-lsp-egress** records, and as it cannot be written to an accounting file, it needs a **destination** of **null**. The **accounting-policy** also has a **collection-interval** command that determines the frequency with which the rates are calculated. It has a default of 5 minutes, which incidentally is also the minimum collection interval. The **accounting-policy** is then put into an **admin-state** of **enable** and referenced within the SR-TE LSP **egress-statistics** context for PCC-initiated SR-TE LSPs. In addition to this, **collect-stats** is also set to **true**. For PCE-initiated SR-TE LSPs, the **accounting-policy** and **collect-stats** commands are enabled under the **lsp-template** (discussed in chapter 6 and explicitly shown in *Output 6-40*).

Output 14-51: Configuration for Rate-Based SR-TE LSP Statistics

```
    log {
        accounting-policy 1 {
            admin-state enable
            record combined-mpls-srte-egress
            destination {
                null
            }
        }
    }
    router "Base" {
        mpls {
            lsp "R1-to-R3" {
                egress-statistics {
                    admin-state enable
                    collect-stats true
                    accounting-policy 1
                }

            }
        }
    }
```

Output 14-52 shows the output of the egress statistics command with rate-based statistics enabled. Rate-based statistics are provided as an aggregate of all paths of the LSP and are not included for each path of the LSP if primary/secondary LSPs paths are used. The packet rate (pps) and bit rate (Mbps) are shown on the bottom line.

Output 14-52: SR-TE Statistics Output Including Rate-Based Statistics

```
A:admin@R1# show router mpls sr-te-lsp R1-to-R3 egress-stats

===============================================================================
SR-TE LSP Egress Statistics
===============================================================================
-------------------------------------------------------------------------------
LSP Name : R1-to-R3
-------------------------------------------------------------------------------
Collect Stats    : Enabled            Accting Plcy.    : 1
Admin State      : Up

Path Name        : empty
StatsOperState   : Up
Aggregate Pkts   : 256045             Aggregate Octets : 385735118

Total for all paths
Tot.Aggr Pkts    : 256045             Tot.Aggr Octets  : 385735118
Packet Rate(pps) : 200                Bit Rate(Mbps)   : 2
```

SR Policy Statistics

Statistics for SR Policies are enabled within the **segment-routing sr-policies** context to allow for the fact that some or all SR Policies may be learnt through BGP and will not have a configuration context through which statistics could be enabled.

Output 14-53: Enabling Statistics for SR Policy

```
    router "Base" {
        segment-routing {
            sr-policies {
                egress-statistics {
                    admin-state enable
                }
            }
        }
    }
```

The per-policy statistics are available using the command shown in *Output 14-54*, where each policy is identified through a combination of color and endpoint. If multiple segment-lists exist for a given SR Policy, statistics will be collected for each one individually. Statistics can be cleared using the command "clear router segment-routing sr-policies egress-statistics color <number> end-point <address>" as required.

Output 14-54: Statistics Output for SR Policy

```
A:admin@R1# show router segment-routing sr-policies egress-statistics color 100 end-point
192.0.2.3

===============================================================================
SR-Policies Egress Statistics
===============================================================================

Egress Statistics:

Color           : 100                   Endpoint Addr   : 192.0.2.3
Segment-List    : 1
TunnelId        : 917510                BSID            : 20001
Pkt Count       : 135569                Octet Count     : 71580432


Egress Statistics:

Color           : 100                   Endpoint Addr   : 192.0.2.3
Segment-List    : 2
TunnelId        : 917510                BSID            : 20001
Pkt Count       : 117329                Octet Count     : 61949712
```

Streaming Telemetry

Streaming telemetry is becoming the de-facto standard for providing operational data at higher frequencies. The gRPC Network Management Interface (gNMI) is based on the Google Remote Procedure Call (gRPC) framework and is used to manage network devices. It supports a number of operations such as Get, Set, Capabilities, and Subscribe. gRPC runs over HTTP/2 as its transport protocol and uses Transport Layer Security (TLS) for encryption and integrity. HTTP/2 is designed to make applications run faster and more robust than previous HTTP versions because it is fully multiplexed, meaning it can send multiple requests for data in parallel over a single TCP connection.

When a client wishes to receive information about some particular state data, it creates a subscription using the Subscribe RPC containing a *SubscribeRequest* message. The subscription contains a series of elements, where each element represents a YANG data tree node name. A path is actually formed by concatenating both a *prefix* and one or more paths, however, the default value of the prefix is null, and when null is used a full path needs to be specified. If a non-null prefix is used, the absolute path is comprised of the concatenation of the prefix and the list of path elements. This can be useful if a subscribe request is for multiple leaves of the same branch in the YANG state data model. For example, assume the required data is as follows:

- path state/port/ethernet/statistics/in-octets
- path state/port/ethernet/statistics/out-octets

Rather than listing the full path twice, it is possible to concatenate prefix and path as follows:

- prefix state/port/ethernet/statistics
- path in-octets
- path out-octets

The Subscribe RPC also specifies a desired subscription mode. There are three modes that define the longevity of the subscription:

- ONCE: defines a subscription that returns one-off data.
- POLL: is a subscription that utilises a stream to periodically request a set of data.
- STREAM: is a long-lived subscription that streams data according to the triggers specified within the individual subscription's mode. The stream subscription mode may be one of ON_CHANGE, SAMPLE, or TARGET_DEFINED.

A SAMPLE path mode subscription requires the defined data be returned every *sample_interval*. When a subscription is defined to be ON_CHANGE, data updates are only sent when the value of the data item changes. For ON_CHANGE subscriptions, the server first generates updates for all paths that match the subscription paths. After the initial set of updates, updated values are only transmitted when their value changes. When a client uses the TARGET_DEFINED mode, it is left up to the target (router) to determine the best type of subscription on a per-leaf basis. A leaf may contain for example an incrementing counter so the SAMPLE mode would be appropriate, whilst another leaf may contain the operational state, which is event driven so an ON_CHANGE subscription would be more appropriate. All three STREAM modes are supported in SR-OS for streaming telemetry subscriptions.

When a subscription is successfully established to the gRPC server, *SubscribeResponse* messages are returned from the server containing update notifications about the paths that are subscribed to. Each update is represented as a path and value pair. The path represents the data tree path being subscribed to, and the value represents the value of that data tree path, where the encoding can be either JavaScript Object Notation (JSON), bytes, or protobuf. When JSON or bytes are used the value of the leafs are encoded as typed values. Protobuf however is an open-source data format that uses a binary encoding format for both path and value, where each leaf in the YANG data tree is encoded as a binary index. The intention is to increase the efficiency of the payload, but it does mean that the relevant decoding information needs to be understood by the gRPC client. In SR-OS the protobuf encoded YANG files are distributed with each SR-OS software release and are specific to each release. They can also be found at https://github.com/nokia/7x50_protobufs.

As previously described, gRPC uses HTTP/2 with TLS for encryption. The gRPC server in SR-OS can operate with or without TLS. It is generally recommended that TLS is used in production environments, but the use of TLS requires an exchange of X.509 digital certificates to provide authenticity and encryption keying material. Whilst the TLS handshake itself uses asymmetric keys for encryption, it allows for confidential generation of a shared key to allow for subsequent symmetric encryption. In general, only server-side authentication is used, however, TLS does allow this to be extended to two-way authentication where the client certificate is also authenticated. When an SR-OS device functions as a gRPC server only server-side authentication is performed. Since the purpose of this document is to describe how to implement Segment Routing in SR-OS, detailing the steps required to install a valid X.509 digital certificate for use with TLS would appear to be an unnecessary distraction. That said, given that a reader may wish to implement streaming telemetry for use with Segment Routing it would appear somewhat remiss to exclude that information. As a reasonable compromise, the steps required to install an X.509 certificate for use with TLS are described in Appendix A, while this section will assume that the necessary certificates are installed and available for use with TLS.

Configuring gRPC with TLS

The configuration shown in *Output 14-55* is applied to enable the use of TLS for gRPC using a certificate obtained from the CA and imported into SR-OS using the procedures outlined in Appendix A. Note that the **cert-profile certificate-file** and **key-file** refer to the server certificate and key imported into the cf3:/system-pki/ directory as described within that appendix. The **server-cipher-list** contains a list of negotiated and acceptable ciphers and authentication algorithms for the TLS session establishment. The **server-tls-profile** references both the **cert-profile** and the **server-cipher-list** and is placed into an **admin-state** of **enable**.

Output 14-55: TLS Configuration

```
    system {
        security {
            tls {
                cert-profile "grpc-tls-certificates" {
                    admin-state enable
                    entry 1 {
                        certificate-file "tls-cert"
                        key-file "tls-key"
                    }
                }
                server-cipher-list "grpc-cipher-list" {
                    cipher 4 {
                        name tls-rsa-with3des-ede-cbc-sha
                    }
                    cipher 5 {
                        name tls-rsa-with-aes128-cbc-sha
                    }
                    cipher 6 {
                        name tls-rsa-with-aes256-cbc-sha
                    }
                    cipher 7 {
                        name tls-rsa-with-aes128-cbc-sha256
                    }
                    cipher 8 {
                        name tls-rsa-with-aes256-cbc-sha256
                    }
                }
                server-tls-profile "grpc-tls-profile" {
                    admin-state enable
                    cert-profile "grpc-tls-certificates"
                    cipher-list "grpc-cipher-list"
                }
            }
        }
    }
```

The next step is to configure a user that is allowed to access the system using gRPC. After the TLS handshake has completed the gRPC user is authenticated in the usual manner, which allows for assignment of the relevant user profile and access rights. In this case, the gRPC user must have access to **grpc**.

Output 14-56: gRPC User Access Privileges

```
    system {
        security {
            user-params {
                local-user {
                    user "grpc" {
                        password <password>
                        access {
                            grpc true
                        }
                        console {
                            member ["default"]
                        }
                    }
                }
            }
        }
    }
```

The **grpc** context provides the option to either use unsecure gRPC over native TCP by selecting the **allow-unsecure-connection** option, or when TLS is required the relevant **tls-server-profile** is referenced. The two options are mutually exclusive. The **grpc** context must also be put into an **admin-state** of **enable**. Note that in SR-OS the gRPC service runs on port 57400 by default and is non-configurable.

Output 14-57: Enabling gRPC

```
    system {
        grpc {
            admin-state enable
            tls-server-profile "grpc-tls-profile"
        }
    }
```

The status of the imported certificate can be verified as shown in *Output 14-58*. In general, the status flags provide a reasonably clear summary of any issues that may exist.

Output 14-58: Verify TLS Status

```
A:admin@R1# show system security tls cert-profile "grpc-tls-certificates"

===============================================================================
Certificate Profile Entry "grpc-tls-certificates"
===============================================================================
Id  Certificate File Name     Key File Name             Status Flags
-------------------------------------------------------------------------------
1   tls-cert                  tls-key
===============================================================================
```

Streaming SR Statistics

Once statistics gathering is configured on the router and gRPC is enabled, streaming telemetry can be used to push SR-TE LSP and SR Policy throughput statistics northbound. An example application of its use would be to provide a centralised controller with near-real-time data such that the controller can make informed decisions about the current placement of the path given its throughput requirements, and whether an optimisation may be necessary.

To subscribe to SR-TE LSP or SR Policy the following YANG state paths must be used:

- **SR-TE LSP**: /state/router[router-name=Base]/mpls/statistics/lsp-egress[lsp-name=<name>]/octets
- **SR Policy**: /state/router[router-name=Base]/segment-routing/sr-policies/sr-policy-egress

The example shown in *Output 14-59* uses a Nokia gRPC client known as gNMIc, but any gRPC client compliant with gNMI version 0.7.0 could be used. It shows a STREAM-mode subscription with a path mode of TARGET_DEFINED to an SR-TE LSP at router R1 which has an SR-TE LSP named R1-to-R3. The output shows two update responses from the SR-OS gRPC server interspaced by a period of 10 seconds. Each update contains a timestamp, the prefix/path that is subscribed to, and the corresponding value of that path in octets. The example is very directed in as much as it targets a specific SR-TE LSP using the explicit LSP and path names. Both could however be replaced with a wildcard '*' to have the gRPC server return statistics for all SR-TE LSPs on the target device. Although not shown in the output, after the first update message is received the gRPC server sends a *sync_response* true notification to the client to inform it that the full contents of the stream have been transmitted (or sync'ed) once. This subscription will last until the client terminates it.

Output 14-59: Sample-Mode Subscription to SR-TE LSP

```
[root@centos8-dot32 ejbca]# gnmic -a 192.0.2.1:57400 --tls-ca EJBCAcert.pem -u grpc -p
Nokia4gnmi subscribe -e bytes --path /state/router[router-name=Base]/mpls/statistics/lsp-
egress[lsp-name=R1-to-R3]/lsp-path[path-name=empty]/octets
{
  "source": "192.0.2.1:57400",
  "subscription-name": "default",
  "timestamp": 1638871179352434620,
  "time": "2021-12-07T09:59:39.35243462Z",
  "prefix":    "state/router[router-name=Base]/mpls/statistics/lsp-egress[lsp-name=R1-to-
R3]/lsp-path[path-name=empty]",
  "updates": [
    {
      "Path": "octets",
      "values": {
        "octets": 883584920
      }
    }
  ]
}

{
  "source": "192.0.2.1:57400",
  "subscription-name": "default",
  "timestamp": 1638871189351298043,
  "time": "2021-12-07T09:59:49.351298043Z",
  "prefix":    "state/router[router-name=Base]/mpls/statistics/lsp-egress[lsp-name=R1-to-
R3]/lsp-path[path-name=empty]",
  "updates": [
    {
      "Path": "octets",
      "values": {
        "octets": 883585106
      }
    }
  ]
}
```

The active gRPC subscriptions on a given node can be viewed in SR-OS using the command "show system telemetry grpc subscription" which shows every subscription together with a subscription identifier. Using this subscription identifier, the command can then be further extended to include the paths that each subscription has subscribed to as shown in *Output 14-60*.

Output 14-60: Active Telemetry Subscriptions

```
A:admin@R1# show system telemetry grpc subscription paths 5

===============================================================================
Telemetry gRPC subscription
===============================================================================
Subscription id       : 5
User                  : grpc
Destination           : 192.168.254.34
Port                  : 57814
Subscription mode     : stream
Encoding              : json
Notification count    : 14
Context count         : 14
Notification bundling : Disabled

-------------------------------------------------------------------------------
Paths
-------------------------------------------------------------------------------
Path                  : /state/router[router-name=Base]/mpls/statistics/lsp-
                        egress[lsp-name=R1-to-R3]/lsp-path[path-name=empty]/
                        octets
Path mode             : target-defined
Sample interval       : 10000 ms
Finished samples      : 14
Deferred samples      : 0
Total collection time : 14 ms
Min collection time   : 1 ms
Avg collection time   : 1 ms
Max collection time   : 1 ms
-------------------------------------------------------------------------------
No. of paths          : 1
```

Debugging and troubleshooting protocols that use encryption to exchange messages is often difficult, simply because they are encrypted. SR-OS does however provide some useful debug output for gRPC exchanges that can be useful for troubleshooting purposes. I have used this debug multiple times to identify why a given gRPC connection over TLS isn't being correctly established and always found it useful.

Output 14-61: gRPC Debug Commands in SR-OS

```
*A:R1# show debug
debug
    system
        grpc
            client all
            type gnmi-subscribe
        exit
    exit
exit
```

Streaming Link Delay

Streaming telemetry can be used to push single-hop link latency with near-real-time latency values from either Ethernet CFM (Y.1731), STAMP (TWAMP-Light), or MPLS-DM probes. The streaming of link delay results uses the OAM Performance Management (OAM-PM) framework in SR-OS to provide the foundations for streaming. To that end, OAM-PM test sessions are configured and activated to generate the delay results and make them available for subscriptions. Delay test results are not maintained on the node for streamed data. Rather, the result is a single-shot statistic that is overwritten every time a sample window closes.

Note: Link delay can be advertised northbound towards a controller using BGP-LS, and this is fully supported in SR-OS. It does however require that ASLAs are in use, and Flex-Algorithm is configured and operational. The use of both ASLAs and Flex-Algorithm are described in Chapter 10.

For the purpose of illustration STAMP/TWAMP-Light is used to generate the link delay measurements for a single link between routers R1 and R2. Router R1 will be the test initiator while R2 will be the reflector, and there are configuration requirements at both. *Output 14-62* shows the required configuration at the initiator R1. Link delay results use a **delay-template** within the **streaming** context to define measurement parameters. Within the **delay-template** the **sample-window** is the amount of time the results of individual test probes are measured for before a value is made available for streaming and is configured in seconds. The **window-integrity** command is used to specify a percentage of probe samples that must be present in the measurement interval for that window to be considered valid. If the number of probe samples falls below the value configured in the **window-integrity**, the results are made available but are reported as suspect. The **fd-avg** command is used for the Frame Delay Average measurement and the **ifdv-avg** is used to calculate the Inter-Frame Delay Variation. Both have options for **forward**, **backward**, or **round-trip**, and the round-trip value is selected in this example. If an initiator and responder are present at both ends of a given link, then a **forward** direction measurement would potentially be considered more appropriate.

The test **session** is used to set the configuration parameters of the test probes and is configured with a **session-type** of **proactive** so that the test will start immediately after it has been enabled. It then allows for the selection of IP, Ethernet, or MPLS as a test family, with IP (STAMP/TWAMP-Light) being the family used in this example. The **ip** context allows for the configuration of test parameters such as **source** and **destination** addresses, **routing-instance**, **ttl**, and **destination-udp-port**, with port 862 being the default as defined for STAMP. The **twamp-light** context is assigned a **test-id** which is an integer value in the range [0-2147483647] and references the previously configured **delay-template**. The **interval** command specifies the frequency of individual test probes and is a value in milliseconds with the default being one second. With a configured **interval** of 1 second and a **sample-window** of 10 seconds, there are clearly 10 probes per **sample-window**.

Output 14-62: Link Delay Streaming

```
    oam-pm {
        session "link-R1-to-R2" {
            description "STAMP OAM-PM Latency Probe R1-R2 Link"
            session-type proactive
            ip {
                destination 192.168.0.2
                destination-udp-port 862
                fc af
                router-instance "Base"
                source 192.168.0.1
                ttl 1
                twamp-light {
                    admin-state enable
                    test-id 1
                    interval 1000
                    delay-template "stream-template-1"
                }
            }
        }
        streaming {
            delay-template "stream-template-1" {
                admin-state enable
                sample-window 10
                window-integrity 90
                fd-avg round-trip { }
                ifdv-avg round-trip { }
            }
        }
    }
```

To respond to STAMP probes a session-reflector or responder must be enabled, and *Output 14-63* shows the configuration put in place at router R2. The reflector configuration is in the base routing instance under **twamp-light**. A **reflector** context contains parameters such as the **udp-port** on which it must listen (and which must align with the sender UDP port), and a **prefix** to constrain the list of IP addresses to which it will respond. Finally, the **reflector** must be put into an **admin-state** of **enable**.

Output 14-63: TWAMP/STAMP Reflector

```
    router "Base" {
        twamp-light {
            reflector {
                admin-state enable
                description "STAMP Link Measurement Reflector"
                udp-port 862
                prefix 192.168.0.0/24 {
                }
            }
        }
    }
```

From a gRPC perspective the subscription is an ON_CHANGE STREAM-mode subscription. The following YANG state branches are used for the ON_CHANGE subscription for each of the Ethernet, IP, and MPLS-DM test families respectively. Note that keys can be wildcarded to allow for a single subscription to target multiple paths:

```
/state/oam-pm/session[session-
name=*]/ethernet/dmm/statistics/streaming/metric[metric-id=*]/newest-
closed[direction=*]

/state/oam-pm/session[session-name=*]/ip/twamp-
light/statistics/delay/streaming/metric[metric-id=*]/newest-closed[direction=*]

/state/oam-pm/session[session-name=*]/mpls/dm/statistics/streaming/metric[metric-
id=*]/newest-closed[direction=*]
```

Table 14-2 lists the available options for the keys used in the delay subscription path.

Table 14-2: Keys used in Delay Subscription Paths

Field	Description
session	Name of the OAM-PM session
test-Type	twamp-light | dmm | dm
metric	fd-avg | ifdv-avg
direction	forward | backward | round-trip

Output 14-64 shows an example of the ON_CHANGE STREAM-mode subscription for link-R1-to-R2, again using the gNMIc gRPC client. The output shows a single update notification from the gRPC server containing the four values that are returned for each completed sample window: close-time, sample-count, suspect, and delay. The close-time indicates the time the sample was closed in UTC, while the sample-count indicates the number of samples used to compute the delay, which in this case is 10. The suspect value is set to false if the number of valid probes in the **sample-window** is equal to or greater than the **window-integrity** percentage. Finally, there is a delay value computed over the **sample-window** in microseconds.

Output 14-64: Delay Streaming Update

```
[root@centos8-dot32 ejbca]# gnmic -a 192.0.2.1:57400 --tls-ca EJBCAcert.pem -u grpc -p
Nokia4gnmi subscribe --path /state/oam-pm/session[session-name=link-R1-to-R2]/ip/twamp-
light/statistics/delay/streaming/metric[metric-id=fd-average]/newest-
closed[direction=round-trip] --stream-mode on-change
{
  "source": "192.0.2.1:57400",
  "subscription-name": "default",
  "timestamp": 1638886310352714322,
  "time": "2021-12-07T14:11:50.352714322Z",
  "prefix":                  "state/oam-pm/session[session-name=link-R1-to-R2]/ip/twamp-
light/statistics/delay/streaming/metric[metric-id=fd-average]/newest-
closed[direction=round-trip]",
  "updates": [
    {
      "Path": "close-time",
      "values": {
        "close-time": "2021-12-07T14:11:49.0Z"
      }
    },
    {
      "Path": "sample-count",
      "values": {
        "sample-count": 10
      }
    },
    {
      "Path": "suspect",
      "values": {
        "suspect": false
      }
    },
    {
      "Path": "delay",
      "values": {
        "delay": 21891
      }
    }
  ]
}
```

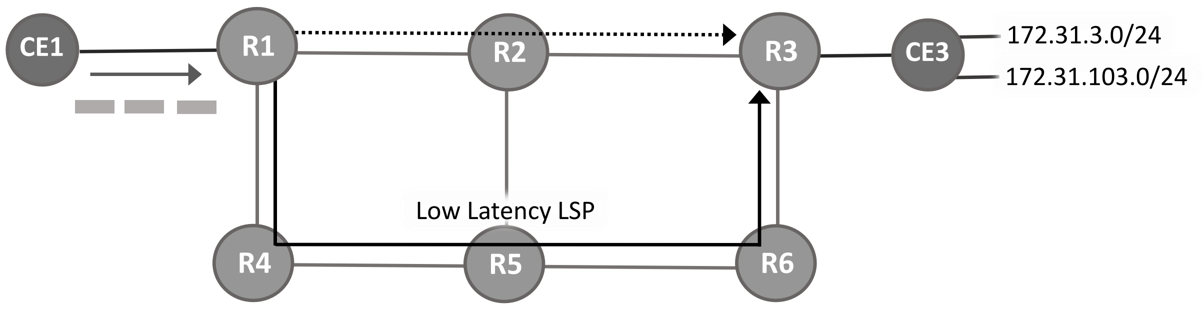
Traffic Steering for SR-TE LSPs with Admin-Tags

When using SR Policies at an ingress router, the Color Extended Community can be used to automatically steer traffic to a given SR Policy. If a BGP or service route is received containing a Color Extended Community with a value corresponding to a valid local SR policy, and the endpoint of that policy matches the next-hop of the BGP/service route, traffic is forwarded into the associated policy. SR Policies therefore provide an inherent mechanism to steer traffic into a particular policy. When using SR-TE LSPs however, this is not the case; there is no inherent way to associate traffic with a given LSP. To provide this capability, SR-OS supports the concept of *LSP tagging*. LSP tagging provides a prefix-based method of traffic-steering and is somewhat analogous to the Color approach used with SR Policy.

SR-TE LSPs can be configured with one or more administrative tags that allow the system to resolve particular BGP routes to a given SR-TE LSP or set of SR-TE LSPs. These BGP routes include BGP labelled unicast, BGP shortcuts, and VPRN/EVPN with auto-bind-tunnel.
Example applications for using administrative tags include steering traffic onto SR-TE LSPs with certain properties such as high throughput or low latency, or steering traffic onto LSPs that avoid certain geographies. Like the concept of Color in an SR Policy, it can potentially be considered an approach for 3GPP 5G slicing.

To demonstrate the use and configuration requirements for LSP tagging the setup in *Figure 14-9* is used. A VPRN service with an auto-bind-tunnel resolution-filter of SR-TE is established between R1 and R3, and two SR-TE LSPs are instantiated from R1 to R3. The first LSP simulates a shortest path route through the network and takes the route R1-R2-R3. The second LSP simulates a low latency path and routes through R1-R4-R5-R6-R3. CE3 has two IPv4 prefixes that it advertises to R3. Prefix 172.31.3.0/24 is general traffic that can take the shortest path route. Prefix 172.31.103.0/24 however hosts a voice application that has stringent latency requirements.

Figure 14-9: Test Topology for LSP Tagging



The first configuration task is to create one or more admin-tags within the **routing-options** context. The **admin-tag** is used to color one or more LSPs with certain characteristics. Up to 256 admin-tags are supported per system. In the example of *Output 14-65* I have created a single **admin-tag** named 'low-latency'.

Output 14-65: Creation of Administrative Tags

```
    routing-options {
        admin-tags {
            admin-tag "low-latency" { }
            }
        }
    }
```

The next step is to configure one or more admin-tags on one or more SR-TE LSPs for the purpose of tunnel selection. Note that LSP tagging does not deal with tunnel placement, only traffic steering. Up to four admin-tags may be configured for each SR-TE LSP. In this example the **admin-tag** is assigned to the low latency SR-TE LSP that routes R1-R4-R5-R6-R3 and aptly named "R1-to-R3-Low-Latency". For PCE-initiated SR-TE LSPs, the same **admin-tag** command can be configured within the **lsp-template** context

Output 14-66: Assignment of Admin-Tags to SR-TE LSPs

```
    router "Base" {
        mpls {
            lsp "R1-to-R3-Low-Latency" {
                admin-tag "low-latency" { }
                }
            }
        }
    }
```

The next step in the configuration process is to create a **route-admin-tag-policy**. The purpose of the **route-admin-tag-policy** is to include or exclude one or more of the admin-tags. It is possible to create a list of **include** and **exclude** statements and for both to co-exist. In this example however, the **route-admin-tag-policy** simply includes the low-latency **admin-tag**.

Output 14-67: Creation of Route-Admin-Tag-Policy

```
    routing-options {
        admin-tags {
            route-admin-tag-policy "low-latency-policy" {
                include "low-latency" { }
            }
        }
    }
```

The final stage in the configuration process is to match the appropriate received routes and assign those prefixes to an admin-tag-policy with the required characteristics. Recall from *Figure 14-9* that CE3 advertises prefix 172.31.3.024 for general traffic and prefix 172.31.103.0/24 for traffic with low latency requirements. To adopt a similar methodology for colouring of prefixes to that used with SR Policies, when R3 advertises prefix 172.31.103.0/24 into VPN-IPv4 it also attaches a Color Extended Community of color:01:100. *Output 14-68* shows the relevant **vrf-import** policy at router R1. The **policy-statement** '*vrf-one-import*' has two entries. Entry 10 matches the community value color:01:100, assigns it to **admin-tag-policy** 'low-latency-policy', and then moves it to the next-entry for further evaluation. The assignment of the prefix to an **admin-tag-policy** means that only LSPs matching the criteria within that policy will be used to resolve the BGP next-hop. Entry 20 then simply checks if the Route-Target Extended Community for VRF 'one' is present in the prefix, and if it is the route is accepted for import into the VRF.

Output 14-68: Assignment of VPN Prefixes to Admin-Tag-Policies

```
    policy-options {
        community "vrf-one" {
            member "target:64496:1" { }
        }
        community "vrf-one-low-latency" {
            member "color:01:100" { }
        }
        policy-statement "vrf-one-import" {
            entry 10 {
                from {
                    community {
                        name "vrf-one-low-latency"
                    }
                }
                action {
                    action-type next-entry
                    admin-tag-policy "low-latency-policy"
                }
            }
            entry 20 {
                from {
                    community {
                        name "vrf-one"
                    }
                }
                action {
                    action-type accept
                }
            }
        }
    }
```

To summarise the process at the receiving router:

i. Prefix 172.31.103.0/24 from CE3 is assigned to **admin-tag-policy** 'low-latency-policy within R1's VPRN **vrf-import** policy.

ii. The **admin-tag-policy** 'low-latency-policy' has a single statement to **include** admin-tag low-latency.
iii. The **admin-tag** low-latency is assigned to the SR-TE LSP named 'R1-to-R3-Low-Latency' that routes via R1-R4-R5-R6-R3.

In short, the BGP next-hop for prefix 172.31.103.0/24 must be resolved using the SR-TE LSP 'R1-to-R3-Low-Latency' since this is the only LSP that matches the criteria specified in the admin-tag-policy.

To verify that the traffic steering is functioning as expected, CE1 offers traffic towards prefix 172.31.103.0/24. Egress statistics are enabled on the LSP 'R1-to-R3-Low-Latency" and the aggregate packet rate is verified as incrementing.

Output 14-69: Egress Statistics for SR-TE LSP 'R1-to-R3-Low-Latency'

```
A:admin@R1# show router mpls sr-te-lsp "R1-to-R3-Low-Latency" egress-stats

===============================================================================
SR-TE LSP Egress Statistics
===============================================================================
-------------------------------------------------------------------------------
LSP Name : R1-to-R3-Low-Latency
-------------------------------------------------------------------------------
Admin State      : Up

Path Name        : empty
StatsOperState   : Up
Aggregate Pkts   : 2479              Aggregate Octets : 674288

Total for all paths
Tot.Aggr Pkts    : 2479              Tot.Aggr Octets  : 674288
```

SR Operations and Maintenance (OAM)

Detecting MPLS Data-Plane Failures [50] describes a simple and effective mechanism to detect data-plane failures in MPLS LSPs. It specifies a probe message known as an MPLS Echo Request and models the test methodology on the ping/traceroute paradigm. Ping, or LSP ping is used for end-to-end connectivity checks. Traceroute, or LSP traceroute, is used for hop-by-hop fault localisation and path tracing.

The MPLS Echo Request contains a base header followed by information about the Forwarding Equivalence Class (FEC) whose MPLS path is being verified. In ping mode the packet should reach the end of the path, at which point it is sent to the control plane of the egress LSR to verify whether it is indeed an egress for the FEC. In traceroute mode the packet is sent to the control plane of each transit LSR, which performs various checks to confirm that it is indeed a transit LSR for this path.

The MPLS Echo Request is a UDP packet with a destination IP address in the 127/8 range for IPv4, and that same range embedded in an IPv4-mapped IPv6 address for IPv6. Routing of the MPLS Echo Request is based solely in the imposed label stack, hence this destination IP address is never used in a forwarding decision. The contents of the Echo Request and Reply are shown in *Figure 14-10*. The version number is currently 1. The message type is set to 1 for Echo Request and 2 for Echo Reply. The Reply Mode instructs the destination how to respond to Echo Request with the Echo Reply, and in SR-OS is always via an IPv4/IPv6 UDP packet. Return Codes and Return Subcodes allow the destination to indicate how it processed the Echo Request, and a set of pre-defined Return Code values are contained in section 3.1 of [50]. The Return Subcode uses the notation <RSC> and contains the point in the label stack where processing was terminated. If the RSC is 0, no labels were processed, otherwise the packet was label switched at depth <RSC>. The Return Code is always set to zero in the Echo Request. The Sender's Handle has no semantics and is filled by the sender and returned unchanged allowing the sender to correlate requests and replies. The Sequence Number is incremental and is used for detection of missing replies. The base header is followed by one or more TLVs, and two of note are the Target FEC Stack TLV and the Downstream Detailed Mapping (DDMAP) TLV which are discussed further below.

Figure 14-10: MPLS Echo Request/Reply Header

Version Number		Global Flags	
Message Type	Reply Mode	Return Code	Rtn Subcode
Sender's Handle			
Sequence Number			
Timestamp Sent (seconds)			
Timestamp Sent (seconds fraction)			
Timestamp Received (seconds)			
Timestamp Received (seconds fraction)			
TLVs (variable)... Target FEC Stack, DDMAP, etc			

The Target FEC Stack TLV contains a list of sub-TLVs, where each sub-TLV has an IANA allocated Sub-Type to indicate the type of FEC being tested, such as LDP IPv4 Prefix, and RSVP IPv4 LSP. The base specification in [50] included Sub-Types to cover known MPLS protocols at the time of writing but did not contain a Sub-Type for SR. The base specification was therefore extended in [51] to include Target FEC Stack Sub-TLVs for three new Sub-Types, the IPv4 IGP-Prefix Segment ID, the IPv6 IGP-Prefix Segment ID, and the IGP-Adjacency Segment ID. The format for the IPv4 and IPv6 IGP-IPv4 Prefix Segment ID is identical apart from the size of the address field and is shown in *Figure 14-11*. The Protocol field is used to instruct the responder to validate the FEC with either OSPF (1), IS-IS (2), or any protocol (0).

Figure 14-11: IPv4/IPv6 IGP-Prefix Segment ID Sub-TLV

IPv4 or IPv6 Prefix		
Prefix Length	Protocol	Reserved

The IGP-Adjacency Segment ID sub-TLV contains local and remote interface IDs for the link to which the Adj-SID is bound. It also contains Advertising and Receiving Node Identifier fields that when set to 1 carry either a 32-bit OSPF Router ID and when set to 2 carry a 48-

bit IS-IS System ID. If set to zero, the field is 32-bits and must be set to zero. The Adjacency Type field is used to indicate whether the adjacency is IPv4, IPv6, or an unnumbered interface, and also if the adjacency is a parallel adjacency. The protocol field serves the same function as previously described for the IPv4/IPv6 IGP-Prefix Segment ID.

Figure 14-12: Adjacency Segment ID Sub-TLV

Adj. Type	Protocol	Reserved
Local Interface ID (4 or 16 octets)		
Remote Interface ID (4 or 16 octets)		
Advertising Node Identifier (4 or 16 octets)		
Receiving Node Identifier (4 or 16 octets)		

The Downstream Detailed Mapping (DDMAP) TLV is sent in an Echo Request and is a request that the DDMAP objects be included in the Echo Reply. It is not returned in an Echo Reply if the responding router is the egress LSR for the FEC and is therefore not used for LSP ping, but rather only LSP traceroute. When returned by an LSR, a DDMAP object should be returned for each interface over which this FEC could be forwarded. The MTU field is the size (in octets) of the largest MPLS frame that could be forwarded on the interface to the downstream LSR. The Address Type field is used to identify whether the downstream interface is IPv4 or IPv6 and numbered or unnumbered. The downstream address and downstream interface address are self-evident, and the size of these fields is determined by the value of the Address Type field. Return Codes and Return Subcodes serve the same function as the Echo Request/Reply header, which have already been discussed.

Figure 14-13: Downstream Detailed Mapping TLV

MTU	Address Type	DS Flags
Downstream Address (4 or 16 octets)		
Downstream Interface Address (4 or 16 octets)		
Return Code	Rtn Subcode	Sub-TLV Length
List of Sub-TLVs		

The DDMAP TLV obsoletes its predecessor the Downstream Mapping (DSMAP) TLV in [50], however, the DSMAP is still supported in SR-OS and is used by default unless explicitly specified otherwise. Optionally it is possible to make DDMAP the default using the command **configure test-oam mpls-echo-request-downstream-map ddmap**.

The DDMAP TLV can contain one or more sub-TLVs which include:

- FEC Stack Change Sub-TLV: Included when the responding node has an Echo Reply where the downstream node has a different FEC Stack than the FEC Stack received

in the Echo Request. This would be the case for example where SR-LDP interworking or any other form of LSP stitching is used.
- Label Stack Sub-TLV: Contains the set of labels in the label stack as it would have appeared if the responding router were forwarding the packet through the interface.
- Multipath Data Sub-TLV: Contains multipath information including labels or addresses that will exercise this particular downstream neighbour. SR-OS does not currently support exercising ECMP paths using LSP traceroute for SR.

Some examples follow showing how to execute LSP ping and LSP traceroute in SR-OS for shortest path SR LSPs, SR-TE LSPs, and SR Policies. The relevant CLI commands and outputs are shown, as well as corresponding packet captures to help solidify understanding of the theory that has been previously described.

Output 14-70 shows an LSP Ping through a shortest path SR LSP from R1 to R3 that routes R1-R2-R3. An LSP ping for a shortest path SR tunnel requires the imposition of a single label onto the MPLS Echo Request, that being the Node-SID of the destination. As shown in the output, the responder replies with an Echo Request containing a Return Code of 3 (`rc=3`) meaning that the replying router is an egress for the FEC at stack-depth <RSC>, where RSC is egress router.

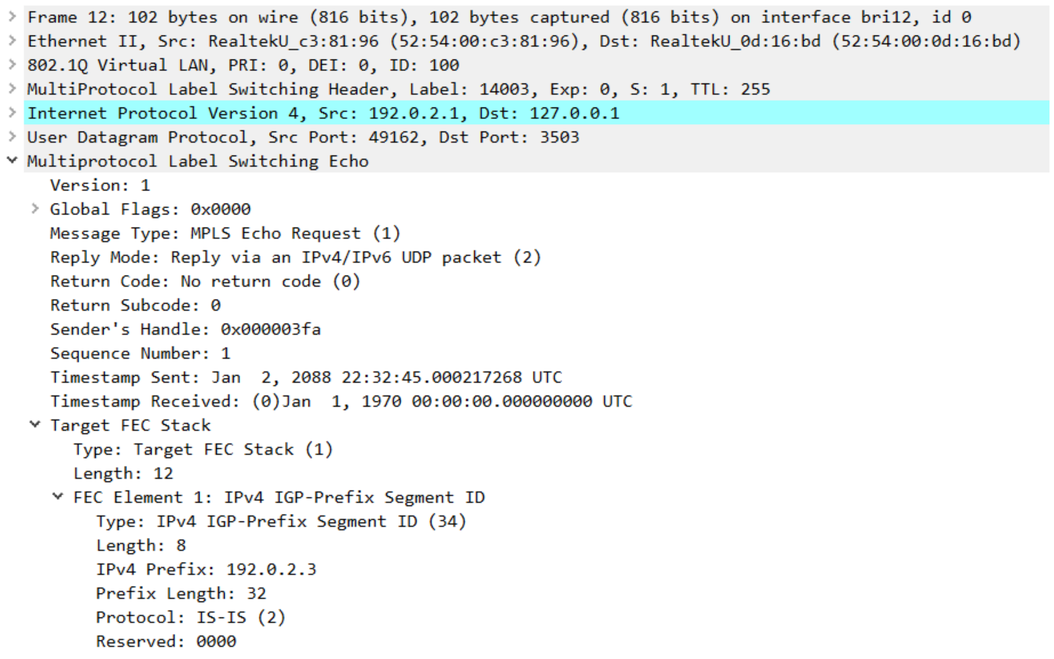
Output 14-70: LSP Ping for SR-ISIS LSP

```
*A:R1# oam lsp-ping sr-isis prefix 192.0.2.3/32 detail
LSP-PING 192.0.2.3/32: 80 bytes MPLS payload
Seq=1, send from intf link-to-R2, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=203ms rc=3 (EgressRtr)

---- LSP 192.0.2.3/32 PING Statistics ----
1 packets sent, 1 packets received, 0.00% packet loss
round-trip min = 203ms, avg = 203ms, max = 203ms, stddev = 0.000ms
```

Figure 14-14 shows the corresponding packet capture for the LSP ping as the packet egresses R1. A single MPLS label of 14003 is imposed on the packet, which is the Node-SID of R3. The base header indicates that the message is an MPLS Echo Request, and requests a response using an IPv4/IPv6 UDP packet. Return Code and Return Subcode are set to zero in the Echo Request as they are only populated by the responder. The Target FEC Stack has a FEC Element (sub-TLV) of IPv4 IGP-Prefix Segment ID, with an IPv4 address of 192.0.2.3 (R3).

Figure 14-14: LSP Ping for Shortest Path SR



```
> Frame 12: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface bri12, id 0
> Ethernet II, Src: RealtekU_c3:81:96 (52:54:00:c3:81:96), Dst: RealtekU_0d:16:bd (52:54:00:0d:16:bd)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 14003, Exp: 0, S: 1, TTL: 255
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 49162, Dst Port: 3503
v Multiprotocol Label Switching Echo
    Version: 1
  > Global Flags: 0x0000
    Message Type: MPLS Echo Request (1)
    Reply Mode: Reply via an IPv4/IPv6 UDP packet (2)
    Return Code: No return code (0)
    Return Subcode: 0
    Sender's Handle: 0x000003fa
    Sequence Number: 1
    Timestamp Sent: Jan  2, 2088 22:32:45.000217268 UTC
    Timestamp Received: (0)Jan  1, 1970 00:00:00.000000000 UTC
  v Target FEC Stack
      Type: Target FEC Stack (1)
      Length: 12
    v FEC Element 1: IPv4 IGP-Prefix Segment ID
        Type: IPv4 IGP-Prefix Segment ID (34)
        Length: 8
        IPv4 Prefix: 192.0.2.3
        Prefix Length: 32
        Protocol: IS-IS (2)
        Reserved: 0000
```

Output 14-71 shows an LSP Ping through an SR-TE LSP from R1 to R3 that routes R1-R4-R5-R6-R3 using strict hops with Adj-SIDs. An LSP ping for an SR-TE LSP requires that all the SIDs that constitute the SR-TE LSP are pushed onto the MPLS Echo Request to ensure that it verifies the data-path. From a router output perspective however, the response is similar to that of the LSP ping through an SR-ISIS LSP. The Return Code is 3 meaning that the replying router is an egress for the FEC at stack-depth <RSC>, where RSC is egress router.

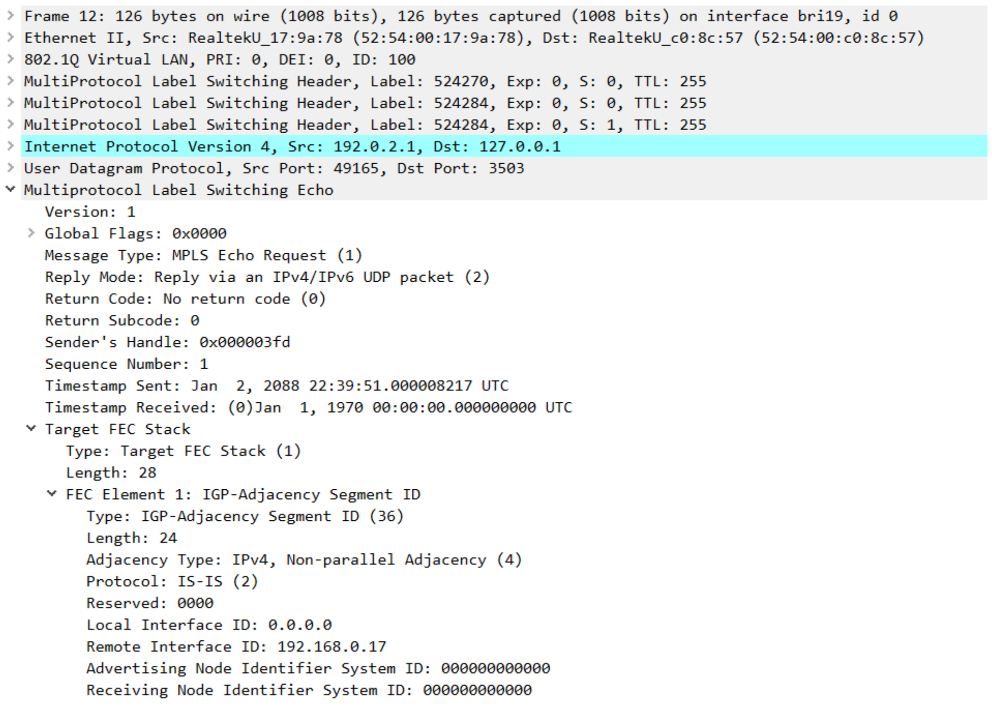
Output 14-71: LSP Ping for SR-TE LSP

```
*A:R1# oam lsp-ping sr-te "R1-to-R3" detail
LSP-PING R1-to-R3: 96 bytes MPLS payload
Seq=1, send from intf link-to-R4, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=205ms rc=3 (EgressRtr)

---- LSP R1-to-R3 PING Statistics ----
1 packets sent, 1 packets received, 0.00% packet loss
round-trip min = 205ms, avg = 205ms, max = 205ms, stddev = 0.000ms
```

The corresponding packet capture for the LSP ping through the SR-TE LSP is shown in *Figure 14-11*. There are three MPLS labels imposed representing the Adj-SIDs of the links R4-R5, R5-R6, and R6-R3 respectively. The Target FEC Stack TLV contains a single FEC Element of IGP-Adjacency Segment ID, with an IP address of 192.168.0.17, which is R3's interface address for the R6-R3 link.

Figure 14-15: LSP Ping for SR-TE LSP



```
> Frame 12: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_17:9a:78 (52:54:00:17:9a:78), Dst: RealtekU_c0:8c:57 (52:54:00:c0:8c:57)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 524270, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 0, TTL: 255
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 1, TTL: 255
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 49165, Dst Port: 3503
v Multiprotocol Label Switching Echo
    Version: 1
  > Global Flags: 0x0000
    Message Type: MPLS Echo Request (1)
    Reply Mode: Reply via an IPv4/IPv6 UDP packet (2)
    Return Code: No return code (0)
    Return Subcode: 0
    Sender's Handle: 0x000003fd
    Sequence Number: 1
    Timestamp Sent: Jan  2, 2088 22:39:51.000008217 UTC
    Timestamp Received: (0)Jan  1, 1970 00:00:00.000000000 UTC
  v Target FEC Stack
      Type: Target FEC Stack (1)
      Length: 28
    v FEC Element 1: IGP-Adjacency Segment ID
        Type: IGP-Adjacency Segment ID (36)
        Length: 24
        Adjacency Type: IPv4, Non-parallel Adjacency (4)
        Protocol: IS-IS (2)
        Reserved: 0000
        Local Interface ID: 0.0.0.0
        Remote Interface ID: 192.168.0.17
        Advertising Node Identifier System ID: 000000000000
        Receiving Node Identifier System ID: 000000000000
```

Output 14-72 gives an example of an LSP Traceroute through an SR-TE LSP from R1 to R3 that routes R1-R4-R5-R6-R3. Note that the **oam lsp-trace** command includes an optional argument of **downstream-map-tlv ddmap** to ensure a DDMAP TLV is used as opposed to a legacy DSMAP TLV. An LSP Traceroute for an SR-TE LSP again requires that all the SIDs that constitute the SR-TE LSP are pushed onto the MPLS Echo Request to ensure that it verifies the data-path. In the output, there are two responses from each transit LSR, 192.0.2.4 (R4), 192.0.2.5 (R5), and 192.0.2.6 (R6), and a single response from the destination R3. Looking at the two responses from R4, the first one shows a Return Code of 3 indicating that the replying router is an egress for the FEC at stack-depth <RSC>, where RSC is 4. This is because R4 is actually the egress router for this segment. R4's second response shows a Return code of 8 indicating that the packet was label switched at stack-depth <RSC>, where RSC is 3. R4's Echo Request also includes a DDMAP TLV showing the downstream address, interface address, and MRU. It also contains a Label Stack sub-TLV showing the set of labels that it would forward through the interface to R5.

The responses from R5 and R6 are similar to that of R4, but the stack-depth in the <RSC> is reduced to account for the reduced number of imposed labels as the packet progresses. The last response is from R3 with a Return Code of 3 as it is the egress router.

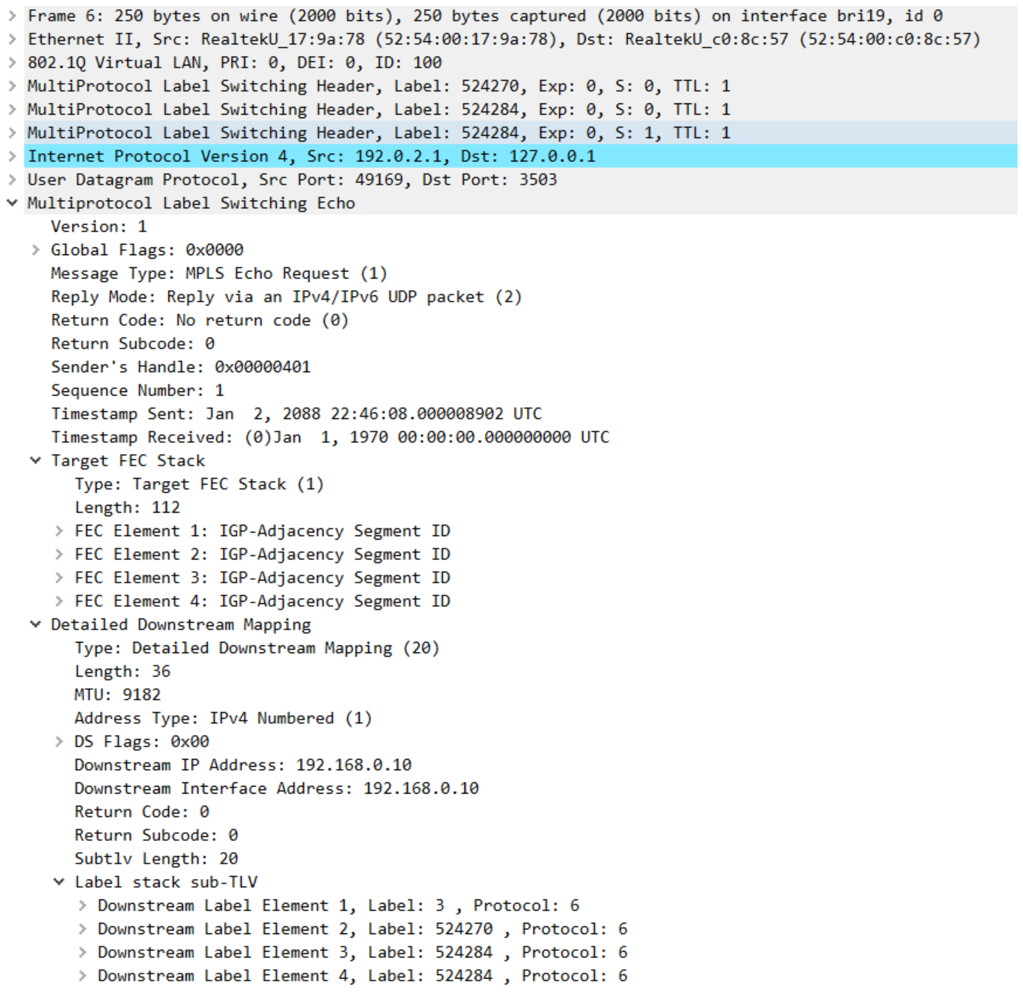
Output 14-72: LSP Traceroute for SR-TE LSP

```
*A:R1# oam lsp-trace sr-te "R1-to-R3" downstream-map-tlv ddmap detail
lsp-trace to R1-to-R3: 0 hops min, 0 hops max, 220 byte packets
1  192.0.2.4  rtt=201ms rc=3(EgressRtr) rsc=4
1  192.0.2.4  rtt=202ms rc=8(DSRtrMatchLabel) rsc=3
     DS 1: ipaddr=192.168.0.22 ifaddr=192.168.0.22 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
           label[2]=524284 protocol=6(ISIS)
           label[3]=524284 protocol=6(ISIS)
2  192.0.2.5  rtt=203ms rc=3(EgressRtr) rsc=3
```

```
2  192.0.2.5  rtt=203ms rc=8(DSRtrMatchLabel) rsc=2
     DS 1: ipaddr=192.168.0.26 ifaddr=192.168.0.26 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
           label[2]=524284 protocol=6(ISIS)
3  192.0.2.6  rtt=204ms rc=3(EgressRtr) rsc=2
3  192.0.2.6  rtt=204ms rc=8(DSRtrMatchLabel) rsc=1
     DS 1: ipaddr=192.168.0.17 ifaddr=192.168.0.17 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
4  192.0.2.3  rtt=205ms rc=3(EgressRtr) rsc=1
```

Figure 14-16 shows the corresponding packet capture as the Echo Request egresses R1. There are three MPLS labels imposed representing the Adj-SIDs of the links R4-R5, R5-R6, and R6-R3 respectively. Note that unlike the LSP Ping captures in *Figure 14-14* and *Figure 14-15* the TTL of the MPLS packets are 1 to ensure that each transit LSR processes the packet. The Target FEC Stack contains four FEC Elements of type IGP-Adjacency Segment ID, representing the links R1-R4, R4-R5, R5-R6, and R6-R3 from top to bottom.

Figure 14-16: MPLS Traceroute for SR-TE LSP



```
> Frame 6: 250 bytes on wire (2000 bits), 250 bytes captured (2000 bits) on interface bri19, id 0
> Ethernet II, Src: RealtekU_17:9a:78 (52:54:00:17:9a:78), Dst: RealtekU_c0:8c:57 (52:54:00:c0:8c:57)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
> MultiProtocol Label Switching Header, Label: 524270, Exp: 0, S: 0, TTL: 1
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 0, TTL: 1
> MultiProtocol Label Switching Header, Label: 524284, Exp: 0, S: 1, TTL: 1
> Internet Protocol Version 4, Src: 192.0.2.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 49169, Dst Port: 3503
v Multiprotocol Label Switching Echo
    Version: 1
  > Global Flags: 0x0000
    Message Type: MPLS Echo Request (1)
    Reply Mode: Reply via an IPv4/IPv6 UDP packet (2)
    Return Code: No return code (0)
    Return Subcode: 0
    Sender's Handle: 0x00000401
    Sequence Number: 1
    Timestamp Sent: Jan  2, 2088 22:46:08.000008902 UTC
    Timestamp Received: (0)Jan  1, 1970 00:00:00.000000000 UTC
  v Target FEC Stack
      Type: Target FEC Stack (1)
      Length: 112
    > FEC Element 1: IGP-Adjacency Segment ID
    > FEC Element 2: IGP-Adjacency Segment ID
    > FEC Element 3: IGP-Adjacency Segment ID
    > FEC Element 4: IGP-Adjacency Segment ID
  v Detailed Downstream Mapping
      Type: Detailed Downstream Mapping (20)
      Length: 36
      MTU: 9182
      Address Type: IPv4 Numbered (1)
    > DS Flags: 0x00
      Downstream IP Address: 192.168.0.10
      Downstream Interface Address: 192.168.0.10
      Return Code: 0
      Return Subcode: 0
      Subtlv Length: 20
    v Label stack sub-TLV
      > Downstream Label Element 1, Label: 3 , Protocol: 6
      > Downstream Label Element 2, Label: 524270 , Protocol: 6
      > Downstream Label Element 3, Label: 524284 , Protocol: 6
      > Downstream Label Element 4, Label: 524284 , Protocol: 6
```

Support for LSP Ping and Trace is also available with SR Policies. Using LSP Ping and LSP Trace with SR Policy results in the imposition of all associated Node-SIDs and Adj-SIDs programmed for the path of the selected segment-list. *Output 14-73* shows an LSP ping through an SR Policy from R1 to R3 that routes R1-R4-R5-R6-R3 using strict hops with Adj-SIDs. The construction of the MPLS Echo Request for an SR Policy is almost identical to that when using SR-TE or SR-ISIS, with one notable exception. As previously described, the Target FEC Stack TLV contains a list of sub-TLVs, where each sub-TLV has an IANA allocated Sub-Type to indicate the type of FEC being tested. In the case of SR Policy however, only the first SID in the segment-list needs to be resolved in order for the path to be considered valid. That is, the headend has no visibility of the type of each segment beyond the first segment in the segment-list. For that reason, the Target FEC Stack TLV identifies each segment in the segment-list as a Nil FEC Type. The Nil FEC Type is described in [50] and allows a Target FEC Stack sub-TLV to be added to the Target FEC Stack for labels that have no explicit FEC associated with them.

Output 14-73: LSP Ping with SR Policy

```
*A:R1# oam lsp-ping sr-policy color 100 endpoint 192.0.2.3 detail
LSP-PING color 100 endpoint 192.0.2.3: 76 bytes MPLS payload
Seq=1, send from intf link-to-R4, reply from 192.0.2.3
       udp-data-len=32 ttl=255 rtt=204ms rc=3 (EgressRtr)

---- LSP color 100 endpoint 192.0.2.3 PING Statistics ----
1 packets sent, 1 packets received, 0.00% packet loss
round-trip min = 204ms, avg = 204ms, max = 204ms, stddev = 0.000ms
```

An LSP Trace through an SR Policy defines the policy to be tested using the **color** and **endpoint**. It again uses the Nil FEC Type in the Target FEC Stack TLV, and as can be seen in the example, the Label Stack sub-TLV in the DDMAP TLV has a protocol type of unknown for every label below the first label. This is again because the headend has no visibility of the type of each segment beyond the first segment in the segment-list.

Output 14-74: LSP Trace with SR Policy

```
*A:R1#  oam  lsp-trace  sr-policy  color  100  endpoint  192.0.2.3  downstream-map-tlv  ddmap
detail
lsp-trace to color 100 endpoint 192.0.2.3: 0 hops min, 0 hops max, 160 byte packets
1  192.0.2.4  rtt=202ms rc=3(EgressRtr) rsc=4
1  192.0.2.4  rtt=202ms rc=8(DSRtrMatchLabel) rsc=3
     DS 1: ipaddr=192.168.0.22 ifaddr=192.168.0.22 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
           label[2]=524284 protocol=0(Unknown)
           label[3]=524284 protocol=0(Unknown)
2  192.0.2.5  rtt=203ms rc=3(EgressRtr) rsc=3
2  192.0.2.5  rtt=203ms rc=8(DSRtrMatchLabel) rsc=2
     DS 1: ipaddr=192.168.0.26 ifaddr=192.168.0.26 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
           label[2]=524284 protocol=0(Unknown)
3  192.0.2.6  rtt=204ms rc=3(EgressRtr) rsc=2
3  192.0.2.6  rtt=204ms rc=8(DSRtrMatchLabel) rsc=1
     DS 1: ipaddr=192.168.0.17 ifaddr=192.168.0.17 iftype=ipv4Numbered MRU=9182
           label[1]=3 protocol=6(ISIS)
4  192.0.2.3  rtt=204ms rc=3(EgressRtr) rsc=1
```

On-Demand Next-Hop SR-TE LSP

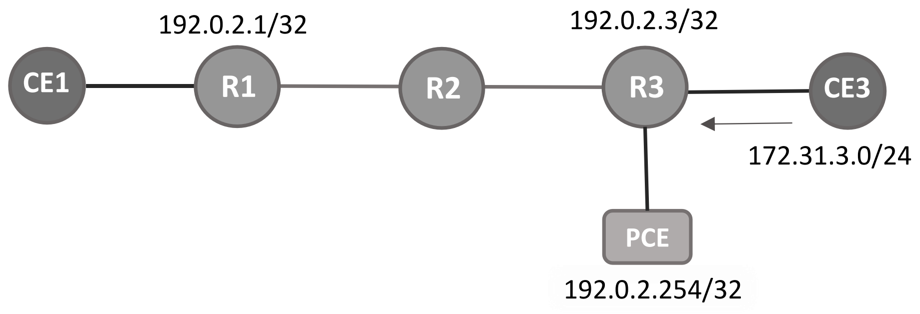
SR-OS supports the concept of an Auto-LSP feature that can also be used to dynamically create SR-TE LSPs. The characteristics of the automatically created SR-TE LSP are derived from an LSP template that can take one of three forms, **p2p-sr-te-mesh**, **p2p-sr-te-one-hop**, and **p2p-sr-te-on-demand**:

- An LSP template of type **p2p-sr-te-mesh** allows for the creation of a mesh of SR-TE LSPs where the destinations of those SR-TE LSPs are dictated by one or more prefix-lists. If the router discovers a router in the traffic engineering database with a router ID matching an entry in the prefix-list, it triggers the instantiation of an SR-TE LSP to that destination.
- An LSP template of type **p2p-sr-te-one-hop** creates a set of one-hop SR-TE LSPs to each of a given routers directly connected IGP adjacencies using the Adj-SID associated with that interface.
- An LSP template of type **p2p-sr-te-on-demand** allows for a partial mesh of tunnels to be created only when they are needed and are known as On-Demand Next-Hop SR-TE LSPs.

The latter is discussed here. The intent of On-Demand Next-Hop SR-TE LSP is to simplify provisioning. For some of lower-end Nokia platforms such as some of the 7250 IXR variants that have limited hardware resource, it could also potentially help to conserve resources by only creating SR-TE LSPs when they are needed. They are dynamically created by the system and triggered by importing a BGP route when a route **policy-statement** contains an **action** statement of **create-mpls-tunnel**. When a BGP route is associated with that action, if an SR-TE LSP does not already exist to that next-hop it is dynamically created and can be computed by either PCE, local-CSPF, or using the hop-to-label translation approach. On-Demand Next-Hop SR-TE LSPs use **admin-tags** to allow certain BGP routes to be associated with a set of SR-TE LSPs that meet certain criteria, such as low-latency, or high-throughput. This allows for dynamic color-aware SR-TE LSP placement. On-Demand Next-Hop SR-TE LSPs can be triggered by BGP routes from the VPN-IPv4, VPN-IPv6, EVPN, Label-IPv4, Label-IPv6, IPv4, and IPv6 address families.

To illustrate the use of On-Demand Next-Hop SR-TE LSPs a VPRN service is created between routers R1 and R3. CE3 advertises IPv4 prefix 172.31.3.0/24 to R3, which is advertised to R1 as a VPN-IPv4 prefix. The prefix will then trigger the creation of a dynamic SR-TE LSP at R1 which is computed and subsequently controlled by a PCE.

Figure 14-17: Test Topology for On-Demand Next-Hop SR-TE LSPs



The use of admin-tags has been described earlier in this chapter in the section 'Traffic Steering for SR-TE LSPs with Admin-Tags', but to reiterate, they can be used to steer traffic onto SR-TE LSPs with certain properties such as high throughput or low latency, or steering traffic onto LSPs that avoid certain geographies. In this example though, the SR-TE LSP path will simply be computed using IGP metric and therefore the **admin-tag** associated with that or those SR-TE LSPs is aptly named 'igp-shortest-path'. The **route-admin-tag-policy** is used to include or exclude one or more **admin-tags**, and again in this simple example the policy 'igp-shortest-path-policy' has an **include** statement for the 'igp-shortest-path' **admin-tag**.

Output 14-75: Admin-Tag and Admin-Tag-Route-Policy

```
    routing-options {
        admin-tags {
            admin-tag "igp-shortest-path" { }
            route-admin-tag-policy "igp-shortest-path-policy" {
                include "igp-shortest-path" { }
            }
        }
    }
```

Within the **mpls** context **an lsp-template** is created with a type of **p2p-sr-te-on-demand**. Most of the SR-TE parameters within the **lsp-template** have been described in chapter 6, with the **pce-control** and **path-computation-method** commands indicating that LSPs will use the PCE for path computation and delegation. The other notable commands are the **vprn-auto-bind** command which is set to **true** to indicate that any dynamically created LSPs may be used for VPRN auto-bind-tunnel purposes, and the **admin-tag** command to group any dynamically created SR-TE LSPs into the policy "igp-shortest-path". The **auto-lsp** command within the **mpls** context also requires configuration and references the previously configured **lsp-template** name.

Output 14-76: LSP-Template and Auto-LSP Configuration

```
    router "Base" {
        mpls {
            lsp-template "on-demand-next-hop" {
                admin-state enable
                type p2p-sr-te-on-demand
                default-path "empty"
                vprn-auto-bind true
                pce-report true
                path-computation-method pce
                pce-control true
                max-sr-labels {
                    label-stack-size 5
                    additional-frr-labels 2
                }
                admin-tag "igp-shortest-path" { }
            }
            auto-lsp "on-demand-next-hop" {
            }
        }
    }
```

The final piece of the configuration is the **policy-statement**, which in this case is referenced in the VPRN **vrf-import** command. The match criteria (or **from** criteria) in this example references all routes imported into the VRF, but an equally likely use-case is to use the Color

Extended Community discussed in chapter 7 to create more granular application of prefixes to groups of SR-TE LSPs with different characteristics. The **action** command assigns the previously configured **admin-tag-policy** and has the **create-mpls-tunnel** command set to true. It is worth stating that the **policy-statement** may contain multiple entries, each with different match criteria and a different **admin-tag-policy** assigned. In this case SR-OS will create multiple sets of SR-TE LSPs to a given next-hop, each computed with the set of constraints associated with the **admin-tag**.

Output 14-77: Route-Policy with Create-MPLS-Tunnel

```
    policy-options {
        policy-statement "vrf-one-import" {
            entry 10 {
                from {
                    community {
                        name "vrf-one"
                    }
                }
                action {
                    action-type accept
                    admin-tag-policy "igp-shortest-path-policy"
                    create-mpls-tunnel true
                }
            }
        }
    }
```

With the above configuration applied at router R1, when any route is imported matching the **from** criteria, the following will happen:

- The route is matched in the route-import policy
- An **admin-tag** named 'igp-shortest-path' is applied.
- A trigger causes MPLS to create an SR-TE LSP to the BGP next-hop of the route that forms part of the set of LSPs bounded by the assigned **admin-tag**.

The test VPRN is configured to have the **auto-bind-tunnel resolution-filter** set to match both **sr-te** and **sr-isis**. The former is of course required in order to use the dynamically created SR-TE LSPs for next-hop resolution. The latter is simply as a form of backstop in the event there is some delay in signalling and establishing the SR-TE LSP so that the prefix can initially resolve to **sr-isis** and when the **sr-te** LSP is established it will move to that owing to the tunnel-table preference. CE3 thereafter advertises prefix 172.31.3.0/24, which is received at router R1 as a VPN-IPv4 route and imported into the test VRF. *Output 14-78* shows the truncated sequence of events at R1 starting with the reception of the VPN-IPv4 prefix 64496:1:172.31.3.0/24 from R3 (192.0.2.3) in frame 142. This triggers R1 to initiate a PCReq message to the PCE for a tunnel to destination R3 in frame 143. The name assigned to the LSP takes the form <auto-lsp-name>-<destination>-<tunnel-ID>::<path-name> and is therefore given the name 'on-demand-next-hop-192.0.2.3-671750::empty'. The PCReq message is immediately followed in frame 145 by a PCRep message containing a computed path, after which R1 sends a PCRpt message indicating that the path is operationally up and active, and delegates the LSP to the PCE. The route is thereafter resolved to the SR-TE LSP in the VPRN route-table.

Output 14-78: Auto-LSP Creation

```
142 2021/12/17 08:58:03.536 GMT minor: DEBUG #2001 Base Peer 1: 192.0.2.10
Peer 1: 192.0.2.10: UPDATE
Peer 1: 192.0.2.10 - Received BGP UPDATE:
    Withdrawn Length = 0
    Total Path Attr Length = 89
    Flag: 0x90 Type: 14 Len: 32 Multiprotocol Reachable NLRI:
        Address Family VPN_IPV4
        NextHop len 12 NextHop 192.0.2.3
        172.31.3.0/24 RD 64496:1 Label 524278 (Raw label 0x7fff61)
 --- [snip] ---

143 2021/12/17 08:58:03.761 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REQUEST ][017 20:10:04.400]
  Svec :{numOfReq 1}{nodeDiverse F linkDiverse F srlgDiverse F}
    Request: {id 4 PST(SegRt) 1 srcAddr 192.0.2.1 destAddr 192.0.2.3}
             {PLspId 4 tunnelId:16389 lspId: 35840 lspName on-demand-next-hop-192.0.2.3-
671750::empty}
 --- [snip] ---

145 2021/12/17 08:58:03.992 GMT minor: DEBUG #2001 Base PCC
PCC: [RX-Msg: REPLY ][017 20:10:04.630]
Peer 192.0.2.254
  Request: {id 4} {} Response has calculated path
        {{Total Paths: 1}}
          Path: {PST(SegRt) 1}
 --- [snip] ---

146 2021/12/17 08:58:04.131 GMT minor: DEBUG #2001 Base PCC
PCC: [TX-Msg: REPORT ][017 20:10:04.770]
Peer 192.0.2.254
  Report : {srpId:0 PST(SegRt):1 PLspId:4 lspId: 35840 tunnelId:16389}
    {Sync 0 Rem 0 AdminState 1 OperState 2 Delegate 1 Create 0}
    {srcAddr  192.0.2.1  destAddr  192.0.2.3  extTunnelId  ::  pathName  on-demand-next-hop-
192.0.2.3-671750::empty}
    {Binding Type: 0 Binding Val : 0}
 --- [snip] ---
```

The SR-TE LSP does not contain any explicit configuration on the node but has the usual operational show commands. It will remain in place until the last BGP route with a next-hop to that destination is withdrawn, after which it will be removed, and any remaining state cleaned up. In the case of a PCE controlled SR-TE LSP, the PCC (R1) will send a PCRpt message with the Remove-flag set to 1 instructing the PCE to remove any state for this path from its database.

Output 14-79: Auto SR-TE LSP Operational State

```
A:admin@R1# show router mpls sr-te-lsp

===============================================================================
MPLS SR-TE LSPs (Originating)
===============================================================================
LSP Name                                          Tun     Protect   Adm  Opr
  To                                              Id      Path
-------------------------------------------------------------------------------
on-demand-next-hop-192.0.2.3-671750               16389   N/A       Up   Up
  192.0.2.3
-------------------------------------------------------------------------------
LSPs : 1
```

The Replication Segment (Tree-SID)

An SR policy can be used to define a point-to-multipoint (P2MP) tree. A P2MP tree uses a variant of the unicast SR policy described in chapter 7 and spans from a root node to a set of leaf nodes via intermediate replication nodes. Each hop in the tree allocates a replication segment, which are stitched together to form the distribution tree. A P2MP SR policy can be considered analogous to a single multicast provider P-tunnel. The P2MP SR policy uses an MPLS data-plane and instructions are programmed using a replication segment. The replication segment is a forwarding entity with an incoming label, together with a set of outgoing interfaces and their associated labels.

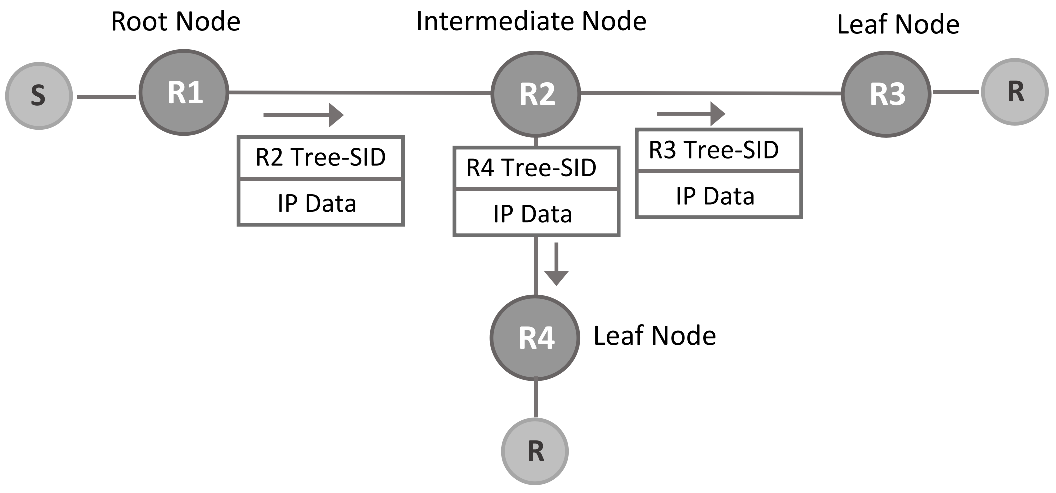
Overview

A P2MP SR policy is identified using the tuple {Root ID, Tree-ID}, where Root ID is the address of the Root node, and Tree-ID is a 32-bit integer value. The P2MP policy is instantiated at the root node and contains one or more candidate paths and a set of leaf nodes. Each candidate path defines the downstream hop(s) for the multipoint distribution tree and contains the relevant replication segment at that downstream hop to complete the next stage of moving from the root towards the leaves of the tree. A downstream node could be a leaf node, an intermediate node, or both which is referred to as a bud node. A P2MP policy does not contain forwarding information for the P2MP LSP. That information is encoded in the replication segment, and as such intermediate nodes and leaf nodes only require a replication policy specifying the instructions to execute for that particular segment.

The replication segment is identified by the tuple {Root ID, Tree-ID, path-instance ID}. which is instantiated in the forwarding plane at the root node, intermediate replication nodes, and leaf nodes. Ingress traffic at non-Root nodes is associated with a replication segment based upon the identification of an incoming replication-SID, or Tree-SID. In this respect, the Tree-SID effectively performs the same function as the Binding SID does for a unicast SR policy. When a packet is received containing a Tree-SID as the active segment, it is mapped to the relevant replication policy which contains the forwarding instructions that the node should execute for that Tree-SID. In an SR-MPLS network the Tree-SID is of course represented as an MPLS label, where the MPLS label range is defined in an SRLB at each Tree-SID capable router. The Tree-SID may be different at the replication and leaf nodes, but if possible it is recommended that the same Tree-SID for a given P2MP tree should be used network-wide for operational ease.

A packet steered into a P2MP tree is replicated by the Tree-SID at the root node to each downstream node in the replication segment, with each replicated packet containing the Tree-SID at the next downstream node. The basic concept is shown in *Figure 14-18*, where a packet is forwarded from root node R1 with instructions to downstream nodes how to deal with this packet in the form of a Tree-SID. Although the schematic shows the Tree-SID with the value of each downstream node in the multipoint distribution tree, the actual value can (and ideally should) be the same at each hop. It is this value that instructs the downstream node how to process the packet, which is why is why it is synonymous with a BSID.

Figure 14-18: Replication Segment



It is possible for two or more P2MP trees to share a replication segment at a root or intermediate replication nodes. If a replication segment is shared, all of its downstream replication segments are shared. Therefore, the leaf nodes reached through a shared replication segment must be either congruent or a subset of the leaf nodes of the P2MP trees that share the segment. In addition, as the leaf nodes share the same replication segment for two or more P2MP trees there must be some way to demultiplex packets and correctly forward them downstream, such as an upstream-assigned MPLS inner label used with Multicast VPN for example (although not currently used in SR-OS). It is also possible for part of the P2MP distribution tree to be based on unicast SR forwarding. For example, assume a topology of R1-R2-R3-R4 where routers R1 and R4 are programmed with replication segments and routers R2 and R3 do not understand replication. Connectivity can be achieved from R1 to R4 using a stacking a unicast SID on top of the replication SID.

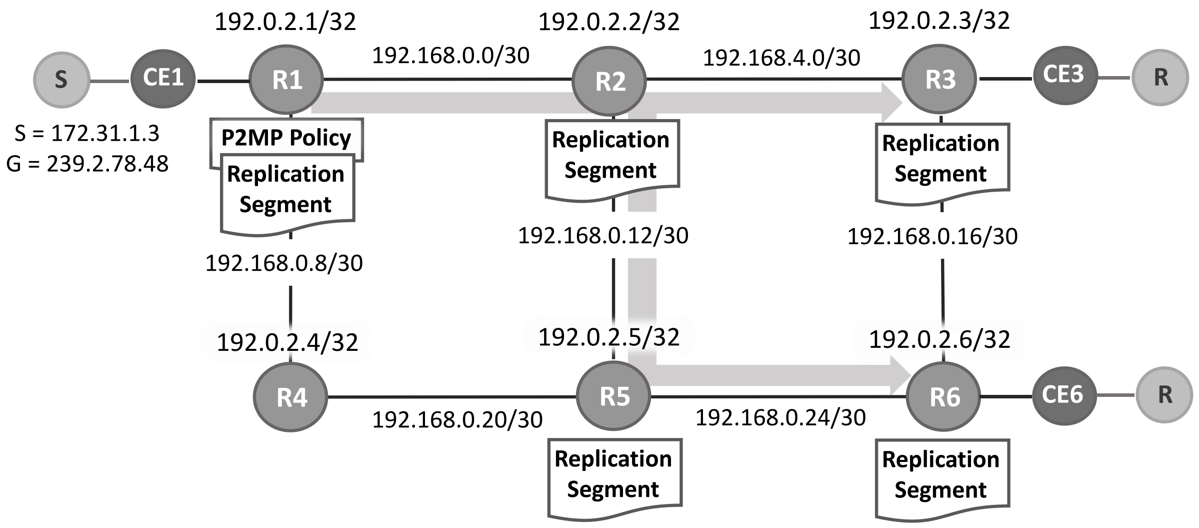
Once the P2MP tree is instantiated it is possible to create a multicast flow overlay that utilises that tree. Examples of multicast flow overlay include Multicast VPN, EVPN, and traditional PIM services. Currently SR-OS supports Multicast VPN overlay with BGP Auto-Discovery messages extended to allow the Provider Multicast Service Instance (PMSI) Tunnel Attribute (PTA) to signal creation of the P-tunnel using Tree-SID. Within each Multicast VPN context P2MP trees can be associated with either Inclusive PMSIs (I-PMSI) or Selective PMSIs (S-PMSI).

Configuration

Unlike unicast SR, P2MP SR trees introduces no additional control plane extensions or signalling protocols therefore a candidate path may be computed and instantiated either manually using CLI, or through use of a PCE. SR-OS currently only supports manual configuration of P2MP policies and replication policies at each node in the tree using CLI.

To illustrate the use of Tree-SID the network topology shown in *Figure 14-19* is used. Routers R1 to R6 are in a flat IS-IS Level 2 domain and SR is enabled with an SRGB of 12000-19999. For the purpose of clarity, unicast Node-SIDs are not shown in the figure. Reference is made as to how unicast Node-SIDs can be used later in this section, but specific Node-SID values will be called out as required. A P2MP tree will be built with R1 as root, and routers R3 and R6 as leaves. R2 and R5 will be transit routers creating the tree with the topology as shown in the diagram. To create a multicast overlay on the P2MP tree, a VPRN service is configured on R1, R3, and R6, that connects CE routers CE1, CE3, and CE6 respectively. These CE routers advertise IPv4 reachability and run PIM towards the core. A multicast source is connected to CE1, while receivers are connected to both CE3 and CE6.

Figure 14-19: Test Topology for Tree-SID



Replication SIDs are allocated local SID values, which means that the SID needs to be within the range of a locally-configured SRLB. As previously described, SRLBs are reserved label blocks used for specific local purposes with SR-MPLS, and a dedicated SRLB is required per-application which has local significance only. Since the SRLB has local significance, the same values could be used on all SR routers in the domain for operational ease. Ranges for each SRLB are taken from the dynamic label range. The SR Local Block is created using the **reserved-label-block** command within the **mpls-labels** context. Within the context of that local block, the **start-label** and **end-label** define the range of the block, and for Tree-SID the range {25000-25999} is allocated.

Once the SRLB is defined it is dedicated to the specific application, which in this case is **p2mp-sr-tree**. After the **reserved-label-block** has been configured, **p2mp-sr-tree** must be put into an **admin-state** of **enable**. The same configuration is applied to all routers in the test topology to implement a common SRLB range throughout.

```
    router "Base" {
        mpls-labels {
            reserved-label-block "Tree-SID" {
                start-label 25000
                end-label 25999
            }
        }
        p2mp-sr-tree {
            admin-state enable
            reserved-label-block "Tree-SID"
        }
    }
```

Configuring a P2MP tree can be decomposed into two main tasks:

- Configuration of a P2MP policy at the root of the P2MP tree. The P2MP policy consists of the root address, a Tree-ID, and one or more candidate paths.
- Configuration of replication segments. Replication segments are required at root nodes, intermediate nodes, and leaf nodes. They are associated with a P2MP policy of the root node and define the forwarding actions to be taken upon receipt of a given replication SID.

Root Node

The P2MP policy shown in *Output 14-80* is configured at router R1. The policy is configured within the **p2mp-sr-tree** context, and starts with a **root-address**, which is the system IP address of the root node, and a **tree-id** in the range 8193-16286. One or more **candidate-paths** are then configured. These candidate paths function the same as unicast SR policies discussed in chapter 7 and allow for end-to-end protection by configuring redundant and diverse trees from root to leaf. Each path has a **preference** value (default 100) and the path with the highest preference value is selected as the active path.

Note: Like unicast SR policies, the candidate path selection criteria will also consider protocol origin and lowest originator value if preference values are the same for multiple candidate paths. However, as the only possible protocol origin is currently CLI with a fixed value of 30, and the CLI originator value is fixed at <0,0.0.0.0>, preference is the key selection criteria. In the event of equal preferences across multiple candidate paths the system will select the candidate path that was created last.

In this example there is only one candidate path so end-to-end protection is not used. Other redundancy mechanisms exist and will be illustrated later in this section. Within each candidate path, up to two **path-instances** may exist. The purpose of **path-instances** is to allow for the co-existence of two separate paths belonging to the same **candidate-path**, of which only one can ever be active at a given time. It is not intended to be a redundancy mechanism. It is intended to allow for the provision of an optimised path, and for a make-before-break (MBB) migration to be carried out from the active instance to the optimised instance by modifying the **active-instance** command. It's worth noting of course that in order to execute such an MBB operation all the relevant replication segments need to be in place at other downstream nodes in the optimised path. In this example, only **path-instance 1** exists, and it is allocated an **instance-id** of 10, which must be unique in a **p2mp-policy** across **candidate-paths**. As only a single path-instance exists, it is the instance referenced in the **active-instance** command. Last but not least, the **p2mp-policy** is placed into an **admin-state** of **enable**.

Output 14-80: P2MP Policy at R1

```
    router "Base" {
        p2mp-sr-tree {
            p2mp-policy "P2MP-POL-R1-8193" {
                admin-state enable
                root-address 192.0.2.1
                tree-id 8193
                candidate-path "8193-PATH-1" {
                    active-instance 1
                    preference 100
                    path-instances 1 {
                        instance-id 10
                    }
                }
            }
        }
    }
```

A replication segment is also required at the root R1 and this is once again configured within the **p2mp-sr-tree** context as shown in *Output 14-81*. Before discussing the contents of the example **replication-segment**, notice that the first command in the **p2mp-sr-tree** context is **bfd-liveness [ipv4]**. Fast reroute is supported for **replication-segment** next-hops and like unicast SR requires a trigger to active a repair path. When BFD is enabled under the **p2mp-sr-tree**, all replication next-hops that use an IP interface with BFD enabled on that interface will associate themselves with the operational state of that BFD session. Note however that because Tree-SID does not introduce any additional control plane signalling, BFD has to be bootstrapped to another protocol such as OSPF or IS-IS for it to become active and for Tree-SID to link to the state of that BFD session. If the BFD session for a given interface should go down, any **replication-segment** next-hop associated with that interface will also go down, which will trigger a fast-reroute if it is programmed. Currently only the facility bypass backup method with link protection is supported. The configuration example in *Output 14-81* provides an example of how fast-reroute can be programmed at router R1 to protect link R1-R2, but for conciseness the necessary configuration is not repeated at subsequent downstream **replication-segments**.

Within the **replication-segment** context, the **instance-id** of 10 references the **instance-id** value configured in the associated **candidate-path**. A replication policy is identified with the tuple {Root ID, Tree-ID, path-instance ID}, hence the **root-address** and **tree-id** are also configured and reference the previously configured **p2mp-policy**. The **sid-action** command defines the MPLS action for the outgoing interface(s), and can be one of **push**, **pop**, **swap**, or **none**. As R1 is the root, the **sid-action** is **push**. A **downstream-nodes** context exists for every outgoing interface of the P2MP tree. In *Figure 14-19* router R1's only downstream node for the P2MP tree is router R2, but two **downstream-node** contexts exist in this configuration example as one will be used to illustrate how fast-reroute facility bypass can be employed. The **downstream-nodes 1** context is used to define the primary path. Within that context the **next-hop-address** command is used to specify the adjacent downstream Next-Hop which may use either IPv4 or IPv6 address families. In this example, the **next-hop-address** is 192.168.0.2 (R2). The **label** context allows for up to 11 **sid-lists** to be configured with each **sid-list** programming a single SID/label that together can form a label stack. In this example a single **sid-list** is configured that uses a **replication-sid** value of 25001. This is the SID (label) that R1 will push onto packets being forwarded into the P2MP tree when forwarding packets to router R2, and therefore must be in the range of R2's SRLB assigned to P2MP SR trees. In this test topology all the routers share a common value for the SRLB, but this may not always be the case. As this is a replication SID of an adjacent neighbor that has no forwarding semantics and cannot be associated with an outgoing interface, the **next-hop-address** is used to provide that forwarding information. The **protect-nexthop-id** allows for referencing a second **downstream-nodes** identifier where a fast reroute backup path is programmed. The construction of the repair path in the **downstream-nodes 2** context requires some detailed explanation so it is covered below the output.

Output 14-81: Replication Segment at R1

```
    router "Base" {
        p2mp-sr-tree {
            bfd-liveness [ipv4]
            replication-segment "R1-8193-Root-Downstream" {
                admin-state enable
                instance-id 10
                root-address 192.0.2.1
                tree-id 8193
                sid-action push
                downstream-nodes 1 {
                    admin-state enable
                    next-hop-address "192.168.0.2"
                    protect-nexthop-id 2
                    label {
                        sid-list 1 {
                            replication-sid 25001
                        }
                    }
                }
                downstream-nodes 2 {
                    admin-state enable
                    label {
                        sid-list 1 {
                            replication-sid 524268
                        }
                        sid-list 2 {
                            replication-sid 524273
                        }
                        sid-list 3 {
                            replication-sid 524277
                        }
                        sid-list 4 {
                            replication-sid 25001
                        }
                    }
                }
            }
        }
    }
```

The **downstream-nodes 2** context contains the instructions for the repair path expressed through the **sid-list**. There are two approaches for constructing a backup or repair path:

- Use of a single Tree-SID from the local block allocated to P2MP SR trees.
- Use of a unicast SR label or label stack.

When a Tree-SID is used for the **protect-next-hop-id** it will tunnel one or more primary paths inside the repair tunnel. The outer repair tunnel label uses a locally unique Tree-SID value because it is considered a *shared* replication segment that can be used by multiple P2MP trees. As a result, it can be considered a facility backup tunnel. The outer label will steer traffic towards the merge point (MP), at which point the inner label representing the primary path replication-SID will be exposed and used as a de-multiplexer to merge P2MP trees back onto their primary paths. Tree-SID labels are not inherently associated with forwarding instructions at transit nodes therefore any transit nodes require **replication-segments** to be configured to instruct them how to process and forward the repair tunnel packets towards the MP accordingly. In addition, if PHP is not used the MP will also need a **replication-segment** to pop the outer **replication-sid**.

When unicast SR labels are used for the **protect-next-hop-id** the **label** context may contain either a single **sid-list** index or multiple **sid-list** indices, where the latter can be used to construct a unicast SR label stack. The behaviour of the backup differs between a path constructed of single SR label and a path constructed of an SR label stack:

- If a single unicast SR label is used, the **protect-next-hop-id** will function largely the same as a Tree-SID from the perspective of the head-end node. It will tunnel one or more primary paths inside the repair path and can again be considered a facility backup tunnel. A single **sid-list** entry is also associated with a **next-hop-address** or **next-hop-interface** in the same manner as Tree-SID (despite the fact that the unicast SR label does have forwarding semantics). One advantage of this approach over the use of Tree-SID is that it doesn't require explicit configuration of replication-segments at transit nodes because it is a unicast SR label and therefore inherently associated with forwarding information.
- If multiple unicast SR labels are stacked through definition of two or more **sid-list** indices, the primary path **replication-sid** is not tunnelled inside the repair path. For that reason, this can be termed a repair path as opposed to a repair tunnel used with a single unicast SR label or a Tree-SID. The consequence of not tunnelling primary path replication-SIDs is that the list of SIDs defined in the **sid-list** needs to include any **replication-sid** that is being used on the primary path at the bottom of the stack. Moreover, by including the **replication-sid** of the primary path in the configured **sid-list**, the repair path cannot be a shared replication segment and can be considered one-to-one protection. When two or more **sid-list** entries are used to form a label stack, it is mutually exclusive with the use of either the **next-hop-address** or the **next-hop-interface** commands and the active label is used to derive an outgoing interface.

The use of a Tree-SIDs or unicast SR labels to construct a repair path is a decision for the operator based on the network environment and requirements. Both have advantages and disadvantages. Tree-SIDs have the advantage of being shared replication segments that can tunnel multiple P2MP trees, but equally they need replication-segments to be configured on any transit nodes (and the MP if PHP is not in use). Unicast SR labels have the advantage of not requiring configuration on transit nodes but in the case of a label stack can only provide one-to-one protection. It is however worth mentioning that the use of unicast SR labels in this manner extends beyond fast reroute repair paths. They could just as equally be used to create a label stack for a primary path to transit a non-multicast-capable SR router for example.

In the example of *Output 14-81*, the backup path consists of four segments or **sid-list** indices. The first three are unicast SR labels, so to some extent using the term **replication-sid** to configure them is somewhat misleading. The first label contained in **sid-list 1** is top of stack and is **replication-sid** 524268, which represents R1's Adj-SID to R4 and as such provides a next-hop forwarding interface. The second label contained in **sid-list 2** is the next label in the stack from top to bottom and is **replication-sid** 524273, representing R4's Ad-SID for the link R4-R5. The third label contained in **sid-list 3** is 524277, which is R5's Adj-SID for the link R5-R2, and as such these first three labels represent an explicit-routed unicast SR tunnel routed via R1-R4-R5-R2. The final label in the stack contained in **sid-list 4** is the **replication-sid** used on the primary path, 25001. Hence, packets pushed into the backup path will be forwarded by R1 towards R4 and will consist of the label stack {524273, 524277, 25001} noting that R1 does not impose its own Adj-SID in the data-path. R4 will swap label 524273 to implicit-null and forward to R5. Equally, R5 will swap label 524277 to implicit-null and forward to R2. What arrives at router R2 is packets containing an active segment of Tree-SID 25001, exactly the same as the primary path.

Finally, both **downstream-nodes** and the **replication-segment** are placed into an **admin-state** of **enable**.

Before moving on to look at the replication-segments configured at downstream routers, it's worth spending some time to validate the operational status of the P2MP tree and replication segment at R1. *Output 14-82* shows the operational status of the p2mp-policy, which shows that the policy is administratively and operationally up. The `activeCPName` field shows the configured candidate path '8193-PATH-1' as active and has an active instance ID of 1 (`ActInstId`).

Output 14-82: Operational State of P2MP Policy

```
A:admin@R1# show router p2mp-sr-tree p2mp-policy "P2MP-POL-R1-8193"

===============================================================================
P2MP Policies
===============================================================================
Name                                    TreeId NumPaths       Adm/Opr
RootAddr                                                      SvcId
activeCPName                                                  ActInstId
-------------------------------------------------------------------------------
P2MP-POL-R1-8193                        8193   1              Up/Up
192.0.2.1                                                     1
8193-PATH-1                                                   1

-------------------------------------------------------------------------------
Total P2MP Policies : 1
```

Output 14-83 shows the operational state of the **candidate-path** within the **p2mp-policy**. There are two instances visible. Instance 0 is the default and is not configured with an instance ID. Instance 1 is the non-default configured instance and is assigned ID 10, which is shown as the active instance. It is both administratively and operationally up.

Output 14-83: Operational State of Candidate Path

```
A:admin@R1# show router p2mp-sr-tree p2mp-policy "P2MP-POL-R1-8193" p2mp-candidate-path
"8193-PATH-1"

===============================================================================
P2MP-Policy "P2MP-POL-R1-8193" Candidate Paths
===============================================================================
Name                                   Pref           Origin PlspId   In- Adm/Opr
Instances                              ActInst               Discmnat Use
                                                             or
-------------------------------------------------------------------------------
8193-PATH-1                            100            Static 0        Yes Up/Up
0 1(10)                                1(10)                 1

-------------------------------------------------------------------------------
Total P2MP Candidate Paths : 1
```

To complete validation of the configured objects, *Output 14-84* shows the operational state of the **replication-segment**. The `InstId` and `TreeId` fields are hopefully clear by now. The `NumNhIds` field shows that there are two configured downstream-nodes (or next-hops). The `InSID` field is blank simply because router R1 is root of the P2MP tree and therefore does not have an incoming SID value. On downstream nodes, this incoming SID value will be populated by configuration of a **replication-sid** in the **replication-segment** that serves the same function that the Binding SID does with unicast SR. On the root node, this **replication-sid** is not necessary, and indeed is mutually exclusive with a **sid-action** of push. Finally, the **replication-segment** is both administratively and operationally up.

Output 14-84: Operational State of Replication-Segment

```
A:admin@R1# show router p2mp-sr-tree replication-segment "R1-8193-Root-Downstream"

===============================================================================
Replication Segments
===============================================================================
Name                                   Origin InstId        NumNhIds  Adm/Opr
RootAddr                                      TreeId        InSID     Action
-------------------------------------------------------------------------------
R1-8193-Root-Downstream                Static 10            2         Up/Up
192.0.2.1                                     8193          --        Push

-------------------------------------------------------------------------------
Total Replication Segments : 1
```

Recall that **bfd-liveness [ipv4]** was enabled in the **p2mp-sr-tree**. Router R1 runs IPv4 single-hop BFD with R2 on link R1-R2 and with R4 on link R1-R4, both of which are bootstrapped to IS-IS. When **bfd-liveness** is enabled in the **p2mp-sr-tree**, **replication-segment** next-hops that use **next-hop-address** or **next-hop-interface <name>** will register with BFD and their operational state will be determined by those BFD sessions. This can be seen in *Output 14-85* which shows that for interface link-to-R2 the registered protocols are IS-IS and Tree-SID. The interface link-to-R4 does not include Tree-SID because the **replication-segment** next-hop using that interface does not include a **next-hop-address** or **next-hop-interface**, but rather a unicast SR label stack. Note that only single-hop BFD is currently supported for P2MP trees and therefore does not provide end-to-end fault detection for a switchover of candidate paths in the same way that seamless BFD does for unicast SR.

Output 14-85: BFD Liveness for P2MP Trees

```
A:admin@R1# show router bfd session

===============================================================================
Legend:
  Session Id = Interface Name | LSP Name | Prefix | RSVP Sess Name | Service Id
  wp = Working path   pp = Protecting path
===============================================================================
BFD Session
===============================================================================
Session Id                                        State      Tx Pkts    Rx Pkts
  Rem Addr/Info/SdpId:VcId                      Multipl     Tx Intvl   Rx Intvl
  Protocols                                        Type     LAG Port     LAG ID
  Loc Addr                                                             LAG name
-------------------------------------------------------------------------------
link-to-R2                                           Up       100105     100111
  192.168.0.2                                         3         1000       1000
  isis treeSid                                      iom          N/A        N/A
  192.168.0.1
link-to-R4                                           Up        99886      99791
  192.168.0.10                                        3         1000       1000
  isis                                              iom          N/A        N/A
  192.168.0.9
-------------------------------------------------------------------------------
No. of BFD sessions: 2
```

The final output of the validation process at router R1 is shown in *Output 14-86* that shows the status of the **replication-segment** within the **p2mp-sr-tree** database. The output is busy which makes it a little uneasy on the eye, however, the information contained within it is useful. Each of the configured next-hops are shown together with the outgoing interface and imposed label stack. The `State` field also provides an indication as to whether the next-hop is in or out of service, with both of the next-hops in use showing as in service.

Output 14-86: P2MP-SR Tree Database for Replication-Segment

```
A:admin@R1# show router p2mp-sr-tree database replication-segment root-address 192.0.2.1
tree-id 8193

===============================================================================
Replication Segments
===============================================================================
Policy-name
RootAddr                               TreeId     NumNHLFEs   Instance
Down-Reason                            Tunnel     Origin      State
SID-action                             InLabel    LTN/Local   Update-ID
  Nexthop
  Interface
  Down-Reason                          REP-SID    NhId        State
    Nexthop
    Interface
    Down-Reason                        REP-SID    ProtectNhId State
-------------------------------------------------------------------------------
R1-8193-Root-Downstream
192.0.2.1                              8193       1           10
none                                   73732      Static      inService
push                                   N/A        True/False  0
  192.168.0.2
  link-to-R2
  none                                 25001      1           inService
    --
    --
    none                               524268, 52* 2           inService
```

```
-------------------------------------------------------------------------------
Total replication segments : 1
```

Intermediate Nodes

Moving down the P2MP tree to router R2, *Output 14-87* shows the necessary **replication-segment** configuration to replicate the incoming tree with outgoing interfaces towards R3 and R5. Most of the fields within the **replication-segment** have already been described, but as R2 is an intermediate replication node it's worth reiterating their purpose. The **replication-sid** has a value of 25001, which is the value that R1 imposed when forwarding packets to R2 and serves the same function as a BSID. This value must be in the range of the SRLB assigned to **p2mp-sr-trees** at R2. The tuple {root ID, Tree-ID, path-instance ID} identifies a given **replication-segment** hence they are all configured here. The **sid-action** is a **swap** action. There are two **downstream-nodes**; **downstream-node 1** has a **next-hop-address** of 192.168.0.6 (R3) while **downstream-node 2** has a **next-hop-address** of 192.168.0.14 (R5) and both have a **replication-sid** of 25001. In summary, the incoming SID is 25001, and the outgoing SID for both next-hops is 25001. Hence, the **sid-action** is a **swap** (or continue) action. The commands used to validate the **replication-segment** at R1 (*Output 14-84, Output 14-86*) can be repeated at all intermediate and leaf nodes to verify their operational status.

Output 14-87: Replication-Segment at an Intermediate Node

```
    router "Base" {
        p2mp-sr-tree {
            replication-segment "R1-8193-R2-Downstream" {
                admin-state enable
                replication-sid 25001
                instance-id 10
                root-address 192.0.2.1
                tree-id 8193
                sid-action swap
                downstream-nodes 1 {
                    admin-state enable
                    next-hop-address "192.168.0.6"
                    label {
                        sid-list 1 {
                            replication-sid 25001
                        }
                    }
                }
                downstream-nodes 2 {
                    admin-state enable
                    next-hop-address "192.168.0.14"
                    label {
                        sid-list 1 {
                            replication-sid 25001
                        }
                    }
                }
            }
        }
    }
```

Looking at the intended P2MP tree structure in *Figure 14-19*, router R5 has a very similar configuration to that of R2 with the exception being that R5 only has a single **downstream-node** (R6). For conciseness it is therefore not shown here.

Note: A bud node that contains both interested receivers and downstream nodes uses the same **replication-segment** configuration as an intermediate node, notably with a **sid-action** of **swap**. The MVPN service configuration will exist on the bud, and that is sufficient for SR-OS to determine that incoming packets need to be forwarded both in the service context and to one or more downstream nodes on the P2MP tree.

Leaf Nodes

The leaf nodes R3 and R6 also share identical configuration, and this is shown in *Output 14-88*. The purpose of the contents of the **replication-segment** have been previously described so are not repeated here. The main difference is that as a leaf of the P2MP tree the **sid-action** is **pop**. Routers R3 and R6 will pop the **replication-sid** label and process the packet that is below it in the label stack.

Output 14-88: Replication-Segment at a Leaf Node

```
    router "Base" {
        p2mp-sr-tree {
            replication-segment "R1-8193-Leaf" {
                admin-state enable
                replication-sid 25001
                instance-id 10
                root-address 192.0.2.1
                tree-id 8193
                sid-action pop
            }
        }
    }
```

Multicast Flow Overlay

With the P2MP tree in place, the multicast flow overlay is enabled in the form of a multicast VPN (MVPN) configured at routers R1, R3, and R6. For simplicity and to represent the topology of the example P2MP tree, router R1 is configured as an MVPN **mdt-type sender-only**, while routers R3 and R6 are configured as **mdt-type receiver-only**. The difference is that R1 will advertise an MVPN Intra-AS I-PMSI Auto-Discovery route (type 1) that includes a PMSI Tunnel Attribute (PTA) notifying the leaves of how to join to the tree, whereas R3 and R6 will not include the PTA attribute in their Intra-AS I-PMSI Auto-Discovery routes (since both do not have a tree to join to). This avoids creating a full-mesh of trees which is appropriate for some networks but in this case is simply convenient for this example to overlay the configured P2MP tree.

SR-OS supports multiple forms of PE-CE multicast protocols including but not limited to Protocol Independent Multicast (PIM) in Any-Source Multicast (ASM) and Source-Specific

Multicast (SSM) modes, Internet Group Management Protocol (IGMP), and Multicast Listener Discovery (MLD). In this test topology, all CE routers peer with their adjacent core routers using PIM. It is not the intention of this document to fully describe service creation attributes, but for completeness the configuration required at router R1 to use a P2MP SR tree for an inclusive PMSI is shown in *Output 14-89*. A similar configuration would be used for a selective PMSI tunnel. Specific to the use of a P2MP SR tree is the creation of the **p2mp-sr** context within the **provider-tunnel inclusive** context. Within the **p2mp-sr** context the root of the tree references the previously configured **p2mp-policy** using the **static-policy** command. The context is also put into an **admin-state** of **enable**.

Output 14-89: Root (Sender-Only) MVPN Configuration with P2MP SR Tree

```
    service {
        vprn "one" {
            mvpn {
                c-mcast-signaling bgp
                mdt-type sender-only
                auto-discovery {
                    type bgp
                }
                vrf-target {
                    community "target:64496:1"
                }
                provider-tunnel {
                    inclusive {
                        p2mp-sr {
                            admin-state enable
                            static-policy "P2MP-POL-R1-8193"
                        }
                    }
                }
            }
        }
    }
```

The pertinent changes in the MVPN configuration at the receivers R3 and R6 are shown in *Output 14-90*. Since the P2MP policy is only relevant (and configured) at the root of the tree it cannot be referenced by the leaves. Therefore, the generic command **p2mp-policy true** is used within the **p2mp-sr** context. As previously described, the **mdt-type** is set to **receiver-only** to overlay the topology of the example P2MP tree.

Output 14-90: Leaf MVPN Configuration with P2MP SR Tree

```
    service {
        vprn "one" {
            mvpn {
                mdt-type receiver-only
                provider-tunnel {
                    inclusive {
                        p2mp-sr {
                            admin-state enable
                            p2mp-policy true
                        }
                    }
                }
            }

        }
    }
```

With the MVPN configuration in place, *Output 14-91* shows the MVPN Intra-AS I-PMSI Auto-Discovery route advertised by router R1, truncated to show only the information relevant to the P2MP SR tree. In that regard, the notable part is the PMSI Tunnel Attribute (PTA) that shows a tunnel type of Tree-SID (IANA allocated tunnel type 0x0C for SR-MPLS P2MP tree). The root address is the system address of R1, and the Tree-ID is the configured value of 8193. There are no flags set, and no MPLS label is carried since this is a source-driven multicast tree that will be identified by downstream nodes using the replication SID. This replication SID will thereafter be associated with relevant replication segment identified by the {Root ID, Tree-ID, Instance-ID} tuple, the former two of which are advertised here.

Output 14-91: MVPN I-PMSI Intra-AS Auto-Discovery Route from R1

```
A:admin@R1# show router bgp routes mvpn-ipv4 type intra-ad hunt
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP MVPN-IPv4 Routes
===============================================================================
--- [snip] ---
-------------------------------------------------------------------------------
RIB Out Entries
-------------------------------------------------------------------------------
Route Type     : Intra-Ad
Route Dist.    : 64496:1
Originator IP  : 192.0.2.1
Nexthop        : 192.0.2.1
--- [snip] ---
-------------------------------------------------------------------------------
PMSI Tunnel Attributes :
Tunnel-type    : TREE-SID
Flags          : Type: RNVE(0) BM: 0 U: 0 Leaf: not required
MPLS Label     : 0
Root-addr      : 192.0.2.1              Tree-Idx       : 8193
-------------------------------------------------------------------------------
```

The leaf nodes R3 and R6 also advertise Intra-AS I-PMSI Auto-Discovery routes to complete the process of discovering the participating members in the MVPN, however, they do not signal a PTA as they are configured to be purely receivers in the MDT. When the auto-discovery process is complete the relevant PMSI tunnel interfaces can be seen in the MVPN context, and this is shown in *Output 14-92* from the perspective of R1. Router R1 (192.0.2.1) is the root and is therefore shown as a transport type of Tx-IPMSI, while routers R3 (192.0.2.3) and R6 (192.0.2.6) are leaves and are shown with a transport type of Rx-IPMSI.

Output 14-92: MVPN PMSI Tunnel Interfaces

```
A:admin@R1# show router 1 pim tunnel-interface

===============================================================================
PIM Interfaces ipv4
===============================================================================
Interface                        Originator Address   Adm  Opr  Transport Type
-------------------------------------------------------------------------------
```

```
mpls-if-73732                   192.0.2.1           Up   Up   Tx-IPMSI
mpls-virt-if-1005865            192.0.2.6           Up   Up   Rx-IPMSI
mpls-virt-if-1005866            192.0.2.3           Up   Up   Rx-IPMSI
-------------------------------------------------------------------------------
Interfaces : 3
```

PIM adjacencies are also formed as part of the auto-discovery process. The adjacency to CE1 through interface 'link-to-CE1' is native PIM and has an expiration time relevant to PIM's Hello message exchange mechanism. The adjacencies to R3 (192.0.2.3) and R6 (192.0.2.6) have expiration times of never simply because neighbour discovery was facilitated by BGP (MVPN) Intra-AS I-PMSI Auto-Discovery routes, and if a given neighbour is no longer available it will withdraw that route. There is no requirement for PIM to maintain the adjacency.

Output 14-93: MVPN PIM Adjacencies

```
A:admin@R1# show router 1 pim neighbor

===============================================================================
PIM Neighbor ipv4
===============================================================================
Interface               Nbr DR Prty     Up Time       Expiry Time     Hold Time
   Nbr Address
-------------------------------------------------------------------------------
link-to-CE1             1               0d 14:34:46   0d 00:01:41     105
   192.168.100.2
mpls-virt-if-1005865    1               0d 14:34:46   never           65535
   192.0.2.6
mpls-virt-if-1005866    1               0d 14:34:46   never           65535
   192.0.2.3
-------------------------------------------------------------------------------
Neighbors : 3
```

Multicast source S1 (connected to CE1) at IP address 172.31.1.3 sends traffic to group address 232.2.78.48 at a rate of ~392Kb/s. Both CE3 and CE6 send PIM joins for the group to their adjacent core routers, which triggers both R3 and R6 to advertise C-Multicast Source Tree Join routes. This subsequently triggers the creation of PIM state within the MVPN for the associated group address and source. The incoming interface is the interface towards CE1, and the outgoing interface (not shown without the detail argument) is the I-PMSI and the P2MP SR tree.

Output 14-94: PIM State for Group 232.2.78.48 at R1

```
A:admin@R1# show router 1 pim group 232.2.78.48

===============================================================================
Legend:  A = Active   S = Standby
===============================================================================
PIM Groups ipv4
===============================================================================
Group Address             Type                Spt Bit  Inc Intf       No.Oifs
   Source Address            RP                  State    Inc Intf(S)
-------------------------------------------------------------------------------
232.2.78.48               (S,G)               spt      link-to-CE1    1
   172.31.1.3                172.31.100.254
-------------------------------------------------------------------------------
Groups : 1
```

With the PIM and C-Multicast BGP state established, all that remains is to verify that the multicast traffic is being forwarded correctly through the P2MP SR tree. *Output 14-95* shows the MVPN PIM state at the leaves of the tree, R3 and R6. The incoming interface is the I-PMSI, and the outgoing interface is the relevant interface towards the adjacent CE router. The current forwarding rate is 392Kb/s, which verifies the function of the P2MP SR tree with the created topology.

Output 14-95: Multicast Forwarding Rates for Group 232.2.78.48 at R3 and R6

```
A:R3# show router 1 pim group 232.2.78.48 detail | match expression "Group Address|Incoming
Intf|Outgoing Intf List|Curr Fwding Rate"
Group Address      : 232.2.78.48
Incoming Intf      : mpls-if-73732
Outgoing Intf List : link-to-CE3
Curr Fwding Rate   : 392.000 kbps


A:R6# show router 1 pim group 232.2.78.48 detail | match expression "Group Address|Incoming
Intf|Outgoing Intf List|Curr Fwding Rate"
Group Address      : 232.2.78.48
Incoming Intf      : mpls-if-73732
Outgoing Intf List : link-to-CE6
Curr Fwding Rate   : 392.000 kbps
```

Appendix A: Installing X.509 Certificates

There are numerous ways to obtain, install, and manage X.509 digital certificates in SR-OS including the Certificate Management Protocol version 2 (CMPv2), Enrolment over Secure Transport (EST), and through the gNOI CERT service using the gRPC interface. Indeed, certificates can simply be transferred onto the node using FTP or SCP. The example in this appendix uses the CMPv2 protocol to register and obtain an X.509 certificate. While there are steps in the procedure that are specific to the use of CMPv2, the majority of the procedure is generic and applicable to whatever mechanism is used to obtain a signed certificate.

Transfer and install a CA Certificate and optional Certificate Revocation List (CRL) onto the device. In the following example an open source public key infrastructure (PKI) software is used known as EJBCA. The EJBCA software performs the role of a Certificate Authority and generates its own CA certificate and CRL, which are transferred onto CF3:/ of the target device.

Output A-1: CA Certificate and CRL on CF3:/

```
A:admin@R1# file list cf3:/EJBCA*

Volume in drive cf3 on slot A is SROS VM.

Volume in drive cf3 on slot A is formatted as FAT32

Directory of cf3:

12/03/2021  12:57p                1847 EJBCAcert.pem
12/03/2021  12:57p                 607 EJBCAcrl.crl
               2 File(s)                2454 bytes.
               0 Dir(s)            673476608 bytes free.
```

Generate a public/private keypair on the SR-OS device.

Output A-2: Generate an RSA Public/Private Keypair

```
*A:R1# admin certificate gen-keypair cf3:/tls-key size 2048 type rsa
```

Import the keypair such that it can be used for PKI purposes. When the ‘admin certificate import’ is used to import certificates, keys, or CRLs it loads that entity into system memory so that it can be used for PKI purposes. When the first certificate, key, or CRL is imported the system also automatically creates a cf3:/system-pki/directory where the output file is contained.

Output A-3: Import the Public/Private Keypair

```
*A:R1# admin certificate import type key input cf3:/tls-key output tls-key format der



*A:R1# file dir cf3:/system-pki/

Volume in drive cf3 on slot A is SROS VM.

Volume in drive cf3 on slot A is formatted as FAT32
```

```
Directory of cf3:\system-pki\

12/03/2021  12:43p      <DIR>          ./
12/03/2021  12:43p      <DIR>          ../
12/03/2021  12:57p                1256 tls-key
               1 File(s)                  1256 bytes.
               2 Dir(s)              673476608 bytes free.
```

Import the CA certificate and CRL file into the system-pki directory so that they can be also used for PKI purposes.

Output A-4: Import CA Certificate and CRL

```
A:R1# admin certificate import type cert input cf3:/EJBCAcert.pem output EJBCAcert format
pem

A:R1# admin certificate import type crl input cf3:/EJBCAcrl.crl output EJBCAcrl format
der
```

Display the CA certificate.

Output A-5: Displaying the Imported CA Certificate

```
*A:R1# admin certificate display type cert format der cf3:/system-pki/EJBCAcert
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            30:9c:8e:db:d7:41:1a:ef:5f:c8:23:0d:d3:02:b4:ef:d2:3e:cf:c5
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: UID=c-0r5hus85emrwnnd2o, CN=ManagementCA, O=EJBCA Container Quickstart
        Validity
            Not Before: Mar 24 10:20:06 2021 GMT
            Not After : Mar 24 10:20:06 2031 GMT
        Subject: UID=c-0r5hus85emrwnnd2o, CN=ManagementCA, O=EJBCA Container Quickstart
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (3072 bit)
                Modulus:
                    00:a5:f7:61:5b:a1:09:2d:51:56:7b:07:7f:25:6c:
                    68:90:7e:b6:7d:16:86:d3:36:1e:34:a9:7d:0d:82:
                    54:35:e5:4f:0c:2f:bc:38:1f:52:07:dd:29:70:46:
                    dc:22:97:d9:b1:32:ac:d6:f8:6a:ac:8d:8c:e3:1e:
                    0a:f4:74:26:eb:a1:39:d8:a2:11:8c:09:73:bf:bc:
                    12:a5:81:a7:f5:fe:f7:08:04:ca:92:a5:74:75:8b:
                    2f:d8:9d:4d:93:a4:8d:8d:9c:24:e6:fe:5b:4e:fb:
                    4e:86:fc:fe:00:2b:8d:a5:f1:6c:3d:23:e3:be:12:
                    31:67:14:52:b2:4c:fd:a2:f2:df:29:32:cf:cc:4a:
                    ee:1e:af:5b:36:fb:c7:bf:0e:78:2d:3a:c6:e3:b5:
                    3f:4c:10:d7:8a:56:3b:f2:01:68:00:ed:b6:90:a7:
                    92:2b:21:fc:fe:2d:a4:3a:a9:39:53:87:12:d3:3f:
                    cf:a6:9c:2c:d1:cd:cc:5a:53:66:ec:da:12:d3:00:
                    af:50:33:68:8f:38:37:85:b1:d3:85:c9:cd:f5:10:
                    3b:53:7e:6b:19:e9:5a:b6:6f:5f:63:63:1e:73:83:
                    98:43:bb:b6:5b:13:91:71:93:ad:a5:1a:2a:f1:0a:
                    cb:9f:3c:77:5d:ec:2c:42:aa:d2:fd:12:96:ac:02:
                    58:27:89:dc:b8:8b:8b:74:3d:2e:80:d4:24:85:11:
                    e6:31:ee:d3:34:fd:38:fd:27:fb:c8:ca:3d:12:36:
                    97:81:ae:88:e9:3b:94:1c:89:85:61:25:90:4a:98:
                    3e:35:f4:06:8a:4f:06:e4:e4:a1:d5:17:75:cd:a5:
                    7c:9d:6e:03:a6:62:30:de:52:42:c9:cb:02:b4:4a:
                    92:b6:99:06:be:53:e5:e4:b5:43:45:37:cc:7f:48:
                    d5:af:a9:fa:f6:c6:93:9b:b1:11:a7:fb:46:20:b6:
                    36:68:15:c9:69:0b:1d:3d:20:cc:c6:b9:c8:3c:e4:
```

```
                    40:8f:2e:54:c4:dd:ee:30:df:07
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Authority Key Identifier:
                keyid:45:9A:F0:24:23:5A:F5:6A:FE:F9:29:0F:0A:C5:47:9F:F9:55:99:E3

            X509v3 Subject Key Identifier:
                45:9A:F0:24:23:5A:F5:6A:FE:F9:29:0F:0A:C5:47:9F:F9:55:99:E3
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         63:26:33:4b:c5:25:14:5d:77:8b:db:8f:66:77:64:b4:ae:69:
         b9:52:5d:fb:d5:ad:c2:32:32:da:67:9a:b0:c8:83:7c:a4:59:
         45:ae:b8:b1:4b:43:6a:59:83:2d:52:3c:1c:9f:00:9f:16:10:
         9b:5b:44:69:c8:eb:5f:bf:7e:63:dd:18:da:30:b5:ce:19:e7:
         7c:58:ae:94:a1:32:5c:4e:cc:20:6d:fb:f5:69:59:1d:04:5a:
         02:e2:09:71:9c:ad:9b:c5:7d:8c:f6:f6:21:5b:e7:0c:68:7e:
         8c:69:2d:87:eb:90:fa:ff:a4:86:b0:5b:3c:39:4b:ca:9d:2e:
         be:50:b3:c6:60:b4:03:62:44:a2:91:ca:0e:e5:a9:94:2c:a8:
         39:de:d8:91:c1:fc:22:9f:db:79:8f:11:1e:ae:89:cd:6b:7e:
         65:c1:78:5a:a4:e6:83:0b:23:74:67:70:9a:d9:1b:19:ab:67:
         fb:cc:2c:c7:0d:7e:32:3b:f0:8c:a9:1a:50:e8:2e:29:d9:90:
         ce:19:18:f1:82:cd:d6:59:2d:54:a2:b8:13:86:e6:5c:f5:5c:
         c9:23:2a:90:67:5c:97:e4:26:4b:a3:a7:9b:e5:dd:fb:a9:c1:
         46:0a:ab:21:db:6c:dc:ec:a4:d1:dd:a8:18:97:47:ac:92:1d:
         0c:fd:54:64:52:ea:00:d3:96:b9:c0:32:28:d5:32:19:79:8a:
         1d:ca:ee:39:9c:6c:fb:6a:57:f4:19:8a:c4:29:2f:3d:00:d8:
         3d:ba:eb:60:cc:05:a5:25:8e:17:59:ad:21:e2:b6:d7:48:bd:
         f8:1c:f6:78:ba:6a:20:46:05:6e:69:76:c3:1b:a1:64:77:b0:
         da:dd:78:83:ec:fd:c9:e9:22:c0:ed:2e:c5:fd:4e:f8:37:c6:
         6e:78:c4:97:00:e3:30:cb:7d:ce:b3:77:78:53:28:9b:8a:7c:
         7d:48:6f:d9:94:ad:7b:c2:9a:e0:df:f8:55:c9:ef:e5:0e:53:
         b1:99:87:ae:d1:11
```

To allow for the use of CMPv2 for certificate registration, a **ca-profile** is configured within the **pki** context. The **cert-file** and **crl-file** reference the previously imported CA certificate and CRL respectively (the CA-profile should not be put into an **admin-state** of **enable** until these files are imported). The **url-string** contains the IP address of the EJBCA server followed the relevant directory structure.

In the below configuration the CA-profile refers to the previously generated **ca-crl**. However, PKI is being used purely for CMPv2 and the only certificate the system will receive for the purpose of authentication is the CA’s certificate. In this case checking the CA’s CRL for the validity of the CA’s certificate is largely superfluous. The **ca-crl** can therefore arguably be left unconfigured.

Output A-6: PKI CA-Profile Configuration for CMPv2

```
    system {
        security {
            pki {
                ca-profile "EJBCA-CA" {
                    admin-state enable
                    cert-file "EJBCAcert"
                    crl-file "EJBCAcrl"
                    revocation-check crl-optional
                    cmpv2 {
                        always-set-sender-for-ir true
                        response-signing-cert "EJBCAcert"
                        accept-unprotected-message {
                            error-message true
                            pkiconf-message true
                        }
                        url {
                            url-string
"http://192.168.254.41:8080/ejbca/publicweb/cmp/CMP-Server"
                        }
                    }
                }
            }
        }
    }
```

The following output shows the operational state of the **ca-profile** after it is placed into an **admin-state** of **enable** and the CA certificate and CRL files are referenced.

Output A-7: Operational State of CA-Profile

```
A:admin@R1# show certificate ca-profile "EJBCA-CA"

===============================================================================
PKI CA-Profile Information
===============================================================================
CA Profile     : EJBCA-CA                    Admin State    : up
Description    : (Not Specified)
CRL File       : EJBCAcrl
Cert File      : EJBCAcert
Oper State     : up
Oper Flags     : <none>
Revoke Chk     : crl-optional

CMPv2
-------------------------------------------------------------------------------
HTTP Timeout   : 30 secs                     Router         : Base
CA URL         : http://192.168.254.41:8080/ejbca/publicweb/cmp/CMP-Server
Sign Cert URL  : EJBCAcert
Unprot Err Msg : enabled                     Unprot Pki Conf: enabled
Same RecipNonce: disabled
for Poll-reqs
Set Sndr for IR: True
HTTP version   : 1.1

OCSP
-------------------------------------------------------------------------------
Responder URL  : (Not Specified)
Router         : Base
Trans Profile  : (Not Specified)
```

The next step is the CMPv2 Initial Registration, which will generate a certificate request and request the CA to authorise it. If an automated certificate enrolment protocol such as CMPv2 or EST were not in use, it is possible to generate the request from SR-OS using the "admin certificate gen-local-cert-req" command and save it locally, after which it can be manually transferred to the CA for authorisation.

The **admin certificate** command shown in *Output A-8* executes the CMPv2 **initial-registration** to generate a certificate request and receive a signed certificate from the CA. The **ca** argument instructs the system to use the previously configured **ca-profile** "EJBCA-CA" to execute the initial registration and uses the **key-to-certify** argument to reference the previously generated and imported public/private key pair "tls-key". The **subject-dn** refers to the subject Distinguished Name (DN) and consists of a number of attributes such as country (C), state (ST), organisation (O), and common name (CN) to uniquely identify an end entity. Finally, the **save-as** argument indicates where to the certificate will be saved if the exchange is successful.

Output A-8: CMPv2 Initial-Registration

```
*A:R1# admin certificate cmpv2 initial-registration ca "EJBCA-CA" key-to-certify tls-key
protection-alg password r1 reference 1 subject-dn C=UK,ST=SU,O=ION,CN=r1 save-as cf3:/tls-
cert.der
Processing request...
Received 'accepted'.
```

Once the certificate has been transferred by the CA, it needs to be imported into the cf3:/system-pki directory so that it too can be used for PKI purposes. Note that because the input file is in DER format, the output file is also in DER format. Outputting to a different format will result in a failed import.

Output A-9: Server Certificate Import

```
*A:R1# admin certificate import type cert input cf3:/tls-cert.der output tls-cert format
der
```

Once imported the server certificate can be displayed.

Output A-10: Display Server Certificate

```
*A:R1# admin certificate display type cert format der cf3:/system-pki/tls-cert
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4d:49:25:a0:bb:43:4e:df:cc:da:bd:8d:59:8f:a8:ef:26:14:aa:c8
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: UID=c-0r5hus85emrwnnd2o, CN=ManagementCA, O=EJBCA Container Quickstart
        Validity
            Not Before: Dec  3 12:50:06 2021 GMT
            Not After : Dec  3 12:50:06 2023 GMT
        Subject: CN=r1, O=ION, ST=SU, C=UK
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:de:3d:10:61:26:f5:56:37:1e:06:5f:2b:8b:1a:
                    50:4c:46:89:37:87:02:d8:39:1f:ce:26:cf:33:f4:
                    c5:26:e2:56:dc:7c:bb:9e:3a:13:45:7e:b4:de:5c:
                    e6:ec:32:d9:7b:51:4f:13:87:b4:fd:8b:68:d5:b2:
                    9e:04:c8:28:87:89:dd:ac:8a:08:de:ca:50:58:0e:
                    9e:f9:4d:60:da:f4:ed:1d:96:7b:e2:81:55:ce:0e:
                    c3:b1:6e:25:db:18:85:8b:28:cc:ad:2c:0b:dd:64:
                    94:b5:f3:70:bc:41:c4:d8:f6:93:bd:ca:20:4e:a0:
                    ca:54:0f:eb:8d:3e:3b:af:99:15:00:89:5d:de:a8:
                    07:74:ee:e7:ee:36:33:2f:f8:fb:0e:34:05:7e:ee:
                    81:78:c8:40:70:db:d9:fd:27:3c:1e:47:cf:af:da:
                    84:0c:d2:89:9b:53:f2:7c:d2:27:f7:b2:78:ba:7e:
                    2b:43:3f:e3:3b:01:67:52:d0:00:50:4e:d6:b5:ec:
                    ca:45:b4:f0:28:30:9f:64:d1:22:29:3f:58:d8:d9:
                    2e:f7:39:d0:a5:be:0d:ec:5a:b9:cd:c3:58:3d:b2:
                    91:f3:d8:c1:b1:4c:97:23:01:ae:73:93:a8:af:c5:
                    63:2b:2a:eb:c2:45:b1:97:2a:4d:f9:bc:3d:21:a4:
                    39:cf
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                IP Address:192.0.2.1
    Signature Algorithm: sha256WithRSAEncryption
         15:aa:51:8e:98:ea:c0:8d:0b:ee:8c:64:ec:f8:f9:9a:9e:84:
         32:2e:97:00:24:99:aa:c4:b3:65:56:4d:ff:81:39:90:7a:e2:
         be:7b:67:8f:21:42:03:c6:31:aa:50:8a:f3:78:0c:8c:4a:44:
         1d:23:06:71:4d:5e:07:73:dd:9b:b6:fd:e9:31:5b:cd:c5:42:
         1b:7b:98:7a:24:5e:92:7a:73:56:78:6f:15:8e:3e:43:07:ad:
         4f:db:71:53:7b:14:09:d6:5e:59:89:9f:a8:64:fb:f8:73:55:
         36:b9:43:3a:1f:11:38:ff:73:07:88:c5:93:c9:4a:7f:b0:14:
         f8:20:d3:32:eb:64:ff:23:f7:43:09:af:68:44:3d:2d:77:5d:
         42:01:14:7a:92:fb:fe:61:aa:2e:49:62:7e:d8:db:b5:b6:bf:
         55:b7:c6:bb:a8:53:48:c9:ec:a9:df:80:46:b0:66:b4:18:72:
         7c:63:7b:aa:a2:7d:6c:43:5c:53:ea:c0:3c:06:97:fb:8c:e7:
         bb:34:eb:61:db:9f:49:5d:46:cf:da:21:1a:bf:81:c5:3b:ca:
         95:6f:57:13:3e:5f:0a:96:80:5a:47:c9:2a:ea:aa:93:77:06:
         a1:94:44:25:38:a9:09:03:03:42:84:ae:73:1a:24:7f:38:97:
         0d:1e:25:5d:d9:1b:df:98:83:55:7a:dc:28:9e:c7:b2:ce:83:
         cc:fe:bc:c6:90:58:3a:66:73:a3:0c:6a:e5:3f:7c:f6:10:9e:
         83:a6:b5:82:ee:13:0b:87:20:4a:10:f1:e6:19:f1:fa:1d:0b:
         b2:ee:dd:ea:f0:34:96:f0:47:fc:49:cf:5c:5c:58:ce:f0:d9:
         99:f5:06:00:e8:79:d2:9a:ae:60:81:5e:d7:e4:e4:b1:07:5e:
         a8:70:dd:fe:e3:70:9a:c6:85:87:f2:f4:2c:80:00:52:ac:ee:
         43:a5:6e:7f:5a:95:c7:bb:57:a8:85:ea:f7:4e:bd:e5:a2:88:
         5f:90:ab:ae:4f:a4
```

The CA certificate, CA CRL, end-entity keys, and end-entity certificate are all now installed in SR-OS and available to any application that requires their use, whether that is IPSec, TLS, or some other application.

Appendix B: SRv6 with 16-bit Function Length

The use of a 16-bit function length for SRv6 has some notable differences in SR-OS than the use of a 20-bit function length, hence an example is provided here for completeness. The same topology and VPRN service used in Chapter 12 is used within this appendix to allow direct behavioural comparisons if needed. For conciseness, the configuration outputs provided in this appendix should be considered deltas to that VPRN service configuration example.

As previously indicated, when the function length is not 16-bits the label block for service functions is drawn from the dynamic label range. When 16-bit function lengths are used however, a reserved label-block must be assigned on FP-based platforms, and labels for both static and dynamic service functions are drawn from this block. *Output B-0-1* shows a reserved label-block for SRv6 16-bit function length with a range of 26000-2999.

Output B-0-1: Reserved Label-Block for SRv6 16-bit Function Length

```
        mpls-labels {
            reserved-label-block "SRv6-16bit-FL" {
                start-label 26000
                end-label 29999
            }
        }
```

Note: Configuration of a **reserved-label-block** for 16-bit Function length is only required on FP-based platforms. It is not required on 7250 IXR Gen 2/2c platforms supporting SRv6.

The locator is then configured with a **function-length** of 16-bits and the label-block assigned.

Output B-0-2: Locator Configuration with 16-bit FL and Label-Block

```
        segment-routing {
            segment-routing-v6 {
                locator "Alg128-locator" {
                    function-length 16
                    label-block "SRv6-16bit-FL"
                }
            }
        }
```

A VPRN service is configured for SRv6 working as described in Chapter 12. When using a locator with a 16-bit function length, a different mechanism is used for the allocation and advertisement of service functions/labels:

a) The system dynamically assigns a free function that is lower than (2^16 -1) or (end-label – start-label +1) and greater than **static-function max-entries** (which defaults to 1).
b) The label derived from the above is advertised as the label value in the service route. These are mapped into the most significant 16-bits bits of the label field.

c) Internally, the system assigns a label value equal to (start-label + function-value -1) as the label value that is programmed as an ILM entry.

CE3 advertises IPv4 prefix 172.31.3.0/24 to R3, which in turn advertises that prefix as a VPN-IPv4 route towards RR1. *Output B-0-3* shows the relevant parts of the VPN-IPv4 prefix as received at R1. The SID Structure sub-sub TLV shows the locator block length of 48-bits and locator node length of an additional 16-bits, totalling 64-bits. The function length is 16-bits as configured. The transposition offset is therefore 64-bits, with a transposition length of 16-bits.

Output B-0-3: VPN-IPv4 Prefix as Received at R1

```
A:admin@R1# show router bgp routes 172.31.3.0/24 vpn-ipv4 detail
===============================================================================
 BGP Router ID:192.0.2.1        AS:64496       Local AS:64496
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
                 l - leaked, x - stale, > - best, b - backup, p - purge
 Origin codes  : i - IGP, e - EGP, ? - incomplete

===============================================================================
BGP VPN-IPv4 Routes
===============================================================================
Original Attributes

Network         : 172.31.3.0/24
Nexthop         : 192.0.2.3
Route Dist.     : 64496:1                VPN Label      : 2
Path Id         : None
From            : 192.0.2.10
 --- [snip] ---
SRv6 TLV Type   : SRv6 L3 Service TLV (5)
SRv6 SubTLV     : SRv6 SID Information (1)
Sid             : 2001:db8:128:3::
Full Sid        : 2001:db8:128:3:2::
Behavior        : End.DT4 (19)
SRv6 SubSubTLV : SRv6 SID Structure (1)
Loc-Block-Len  : 48                     Loc-Node-Len   : 16
Func-Len        : 16                     Arg-Len        : 0
Tpose-Len       : 16                     Tpose-offset   : 64
VPRN Imported  :  1
```

The allocated VPN label value of 2 in *Output B-0-3* may be a little confusing. Having configured a reserved local-block with a range of 26000-29999 one might reasonably expect the allocated label to be within this range, but the value 2 is derived as a function of step a) above. This can also be confirmed by looking at R3's locally allocated SIDs for the relevant VPRN (context). In this case, the function (label) value is 2.

Output B-0-4: Local SIDs Instantiated at R3 for VPRN Context "1"

```
A:admin@R3# show router segment-routing-v6 local-sid end-dt4 context "1"

===============================================================================
Segment Routing v6 Local SIDs
===============================================================================
SID                                                 Type           Function
  Locator
  Context
-------------------------------------------------------------------------------
```

```
2001:db8:128:3:2::                               End.DT4        2
  Alg128-Locator
  SvcId: 1 Name: one
-------------------------------------------------------------------------------
SIDs : 1
```

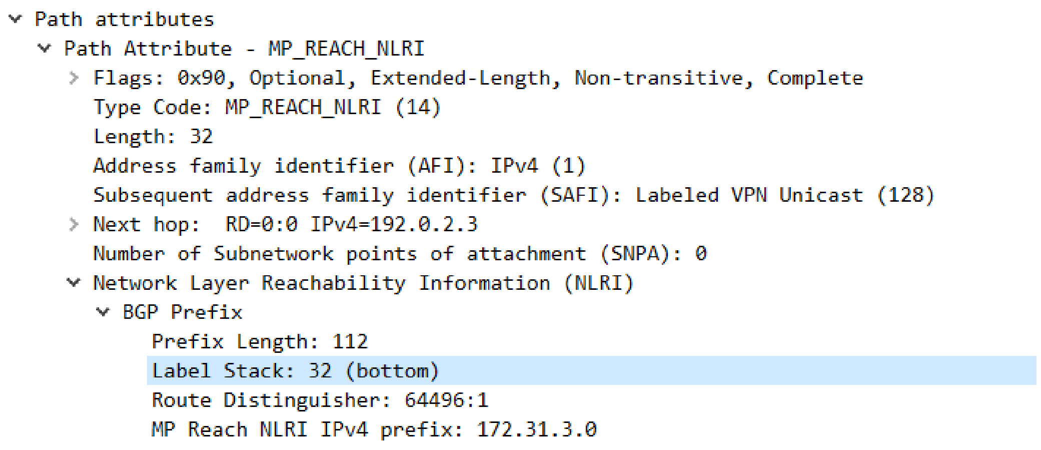
Recall that when the 16-bit function length is transposed into the VPN-IPv4 label field, the 16 most significant bits of the 20-bit MPLS label field are used. When sent on the wire therefore, the actual label value is not seen as 2, but rather label 32. This can be seen in *Output B-0-5* which shows a debug capture of the VPN-IPv4 prefix as received at R1. Within this debug capture, the label value is 32 and shows a raw label value of 0x201. This raw value represents 0x20 for label 32, together with 0x01 to indicate bottom-of-stack bit is set.

Output B-0-5: Debug of VPN-IPv4 Prefix Received at R1

```
15 2022/06/07 15:15:57.399 BST minor: DEBUG #2001 Base Peer 1: 192.0.2.10
Peer 1: 192.0.2.10: UPDATE
Peer 1: 192.0.2.10 - Received BGP UPDATE:
    Withdrawn Length = 0
    Total Path Attr Length = 129
    Flag: 0x90 Type: 14 Len: 32 Multiprotocol Reachable NLRI:
        Address Family VPN_IPV4
        NextHop len 12 NextHop 192.0.2.3
        172.31.3.0/24 RD 64496:1 Label 32 (Raw label 0x201)
    Flag: 0x40 Type: 1 Len: 1 Origin: 0
    Flag: 0x40 Type: 2 Len: 6 AS Path:
        Type: 2 Len: 1 < 64512 >
    Flag: 0x40 Type: 5 Len: 4 Local Preference: 100
    Flag: 0x80 Type: 9 Len: 4 Originator ID: 192.0.2.3
    Flag: 0x80 Type: 10 Len: 4 Cluster ID:
        192.0.2.10
    Flag: 0xc0 Type: 16 Len: 16 Extended Community:
        target:64496:1
        color:01:100
    Flag: 0xc0 Type: 40 Len: 37 Prefix-SID-attr:
       SRv6 Services TLV (37 bytes):-
           Type: SRV6 L3 Service TLV (5)
           Length: 34 bytes, Reserved: 0x0
         SRv6 Service Information Sub-TLV (33 bytes)
             Type: 1 Len: 30 Rsvd1: 0x0
             SRv6 SID: 2001:db8:128:3::
             SID Flags: 0x0 Endpoint Behavior: 0x13 Rsvd2: 0x0
             SRv6 SID Sub-Sub-TLV
                Type: 1 Len: 6
                BL:48 NL:16 FL:16 AL:0 TL:16 TO:64
```

This transposition behaviour can be further verified with a packet capture of the VPN-IPv4 prefix advertised by R3 shown in *Figure B-0-1*. Again it can be seen that the label stack value is 32 and that the bottom-of-stack indicator is set to 1.

Figure B-0-1: Packet Capture of VPN-IPv4 NLRI with Label Value



```
v Path attributes
  v Path Attribute - MP_REACH_NLRI
    > Flags: 0x90, Optional, Extended-Length, Non-transitive, Complete
      Type Code: MP_REACH_NLRI (14)
      Length: 32
      Address family identifier (AFI): IPv4 (1)
      Subsequent address family identifier (SAFI): Labeled VPN Unicast (128)
    > Next hop:  RD=0:0 IPv4=192.0.2.3
      Number of Subnetwork points of attachment (SNPA): 0
    v Network Layer Reachability Information (NLRI)
      v BGP Prefix
          Prefix Length: 112
          Label Stack: 32 (bottom)
          Route Distinguisher: 64496:1
          MP Reach NLRI IPv4 prefix: 172.31.3.0
```

As the final step of the allocation of service function/label with 16-bit function lengths, in step c) the system assigns a label value equal to (start-label + function-value -1) as the label value that is programmed as an ILM entry. In this case, that value is 26001.

References

1	draft-ietf-lsr-flex-algo	IGP Flexible Algorithm
2	RFC8667	IS-IS Extensions for Segment Routing
3	RFC7981	IS-IS Extensions for Advertising Router Information.
4	RFC7770	OSPF Capability Extensions
5	RFC8665	OSPF Extensions for Segment Routing
6	RFC8491	Signalling Maximum SID Depth (MSD) Using IS-IS
7	RFC8476	Signalling Maximum SID Depth (MSD) using OSPF
8	RFC9088	Signalling Entropy Label Capability and Entropy Readable Label Depth Using IS-IS
9	RFC9089	Signalling Entropy Label Capability and Entropy Readable Label Depth Using OSPF
10	RFC7794	IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability
11	RFC8661	SR MPLS Interworking with LDP
12	RFC8402	Segment Routing Architecture
13	RFC5714	IP Fast Reroute Framework
14	RFC5286	IP Fast Reroute: Loop-Free Alternates
15	RFC7490	Remote Loop-Free Alternate (LFA) Fast Reroute
16	RFC7916	Operational Management of Loop-Free Alternates
17	draft-ietf-rtgwg-segment-routing-ti-lfa	Topology Independent Fast Reroute using Segment Routing
18	draft-bashandy-rtgwg-segment-routing-uloop	Loop Avoidance using Segment Routing
19	RFC5440	Path Computation Element (PCEP) Communication Protocol (PCEP)
20	RFC8231	PCEP Extensions for Stateful PCE
21	RFC8664	Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing
22	RFC8408	Conveying Path Setup Type in PCE Communication Protocol (PCEP) Messages
23	RFC8281	PCEP extensions for PCE-initiated LSP Setup in a Stateful PCE Model
24	RFC7752	North-Bound Distribution of Link-State and Traffic Engineering (TE) Information using BGP.

25	RFC9085	BGP-Link State (BGP-LS) Extensions for Segment Routing
26	draft-ietf-idr-rfc7752bis	North-Bound Distribution of Link-State and Traffic Engineering (TE) Information using BGP.
27	draft-alvarez-pce-path-profiles	PCE Path Profiles
28	RFC8697	PCEP Extensions for Establishing Relationships between Sets of Label Switched Paths (LSPs)
29	RFC8745	PCEP Extensions for Associating Working and Protection LSPs with Stateful PCE
30	RFC8800	PCEP Extension for LSP Diversity Constraint Signalling
31	RFC9005	PCEP Extension for Associating Policies and Label Switched Paths (LSPs)
32	RFC7880	Seamless Bidirectional Forwarding Base
33	RFC7881	Seamless Bidirectional Forwarding (S-BFD) for IPv4, IPv6, and MPLS.
34	RFC9256	Segment Routing Policy Architecture
35	draft-ietf-idr-segment-routing-te-policy	Advertising Segment Routing Policies in BGP
36	RFC9012	The BGP Tunnel Encapsulation Attribute
37	RFC6119	IPv6 Traffic Engineering in IS-IS
38	RFC5549	Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop
39	draft-ietf-lsr-flex-algo	IGP Flexible Algorithm
40	RFC8919	IS-IS Application-Specific Link Attributes
41	RFC8920	OSPF Application-Specific Link Attributes
42	RFC5305	IS-IS Extensions for Traffic Engineering
43	RFC3630	OSPF Extensions for Traffic Engineering
44	RFC7308	Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)
45	RFC8570	IS-IS Traffic Engineering (TE) Metric Extensions
46	RFC7471	OSPF Traffic Engineering (TE) Metric Extensions
47	RFC7883	Advertising Seamless Bidirectional Forwarding Detection (S-BFD) Discriminators in IS-IS
48	RFC7884	Advertising Seamless Bidirectional Forwarding Detection (S-BFD) Discriminators in OSPF

49	RFC8972	Simple Two Way Active Measurement Protocol (STAMP)
50	RFC8029	Detecting MPLS Data-Plane Failures
51	RFC8287	LSP Ping/Trace for SR Prefix-SID and Adjacency-SID with MPLS Data Planes
52	RFC6790	The Use of Entropy Labels in MPLS Forwarding
53	RFC8200	IPv6 Specification
54	RFC8754	IPv6 Segment Routing Header
55	RFC8986	Segment Routing over IPv6 (SRv6) Network Programming
56	draft-filfils-spring-srv6-net-pgm-insertion	SRv6 NET-PGM extension: Insertion
57	RFC9252	SRv6 BGP Based Overlay Services
58	RFC8669	Segment Routing Prefix Segment Identifier Extensions for BGP
59	draft-ietf-spring-compression-requirement	Compressed SRv6 SID List Requirements
60	draft-ietf-spring-compression-analysis	Compressed SRv6 SID List Analysis
61	draft-ietf-spring-srv6-srh-compression	Compressed SRv6 Segment List Encoding in SRH
62	draft-filsfils-spring-net-pgm-extensions-srv6-usid	Network Programming Extensions: SRv6 uSID Instruction
63	draft-cl-spring-generalized-srv6-for-cmpr	Generalized SRv6 Network Programming for SRv6 Compression
64	RFC4291	IPv6 Addressing Architecture
65	RFC9087	Segment Routing Centralised Egress Peer Engineering
66	RFC9086	BGP Link State Extensions for Segment Routing Egress Peer Engineering
67	RFC8277	Using BGP to Bind MPLS Labels to Address Prefixes
68	RFC8253	PCEPS: Usage of TLS to Provide a Secure Transport for the Path Computation Element Communication Protocol (PCEP)
69	RFC8202	Multi-Instance IS-IS
70	draft-gredler-idr-bgplu-epe	Egress Peer Engineering using BGP-LU
71	draft-ietf-6man-sids	Segment Identifiers in SRv6
72	RFC8200	IPv6 Architecture

Glossary

ABR	Area Border Router
AFI	Address Family Indicator
ASBR	Autonomous System Border Router
ASLA	Application Specific Link Attribute
ASM	Any-Source Multicast
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
BGP-LS	BGP Link State
BMI	Base MPLS Imposition
BSID	Binding SID
CA	Certificate Authority
CE	Customer Edge (Router)
CLI	Command Line Interface
CMP	Certificate Management Protocol
CO	Color Only (Bits)
COC	Continue of Compression
CPM	Control and Processor Module
CRL	Certificate Revocation List
CSID	Compressed Segment Identifier
CSPF	Constrained Shortest Path First
DAG	Disjoint Association Group
DDMAP	Downstream Detailed Mapping
DLFA	Directed Loop-Free Alternate
EAG	Extended Admin Group
EBGP	External Border Gateway Protocol
ECMP	Equal Cost Multipath
EL	Entropy Label
ELC	Entropy Label Capability
ELI	Entropy Label Indicator
EPE	Egress Peer Engineering
ERLD	Entropy Readable Label Depth
ERO	Explicit Route Object
ESI	Ethernet Segment Identifier
EST	Enrolment over Secure Transport
EVPN	Ethernet Virtual Private Network
FAD	Flexible Algorithm Definition
FAPM	Flexible Algorithm Prefix Metric
FEC	Forwarding Equivalence Class
FIB	Forwarding Information Base
FPE	Forwarding Path Extension

FTP	File Transfer Protocol
gNMI	Google Network Management Interface
gNOI	Google Network Operations Interfafce
gRPC	Google Remote Procedure Call
GTP	General Packet Radio Service Tunnelling Protocol
G-SID	Generalised Segment Identifier
IANA	Internet Assigned Numbers Authority
IBGP	Interior Border Gateway Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Membership Protocol
IGP	Interior Gateway Protocol
ILM	Incoming Label Map
IS-IS	Intermediate System to Intermediate System
I-PMSI	Inclusive Provider Multicast Service Instance
LAG	Link Aggregation Group
LDP	Label Distribution Protocol
LER	Label Edge Router
LFA	Loop-Free Alternate
LSA	Link State Advertisement
LSP	Label Switched Path (MPLS)
LSP	Link State Protocol Data Unit
LSPA	LSP Attribute (Object)
LSR	Label Switching Router
MBB	Make Before Break
MC	Mapping Client
MDT	Multicast Distribution Tree
MLD	Multicast Listener Discovery
MPLS	Multi-Protocol Label Switching
MSD	Maximum Segment Depth
MVPN	Multicast Virtual Private Network
NHLFE	Next-Hop Label Forwarding Entry
NLRI	Network Layer Reachability Information
OAM	Operations and Maintenance
OAM-PM	Operations and Maintenance Performance Management
OSPF	Open Shortest Path First
PAG	Policy Association Group
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PCEPS	PCEP Secure
PCInit	Path Computation Initialisation
PCRep	Path Computation Reply
PCReq	Path Computation Request
PCRpt	Path Computation LSP State Report

PCUpd	Path Computation Update
PE	Provider Edge (Router)
PHP	Penultimate Hop Popping
PIM	Protocol Independent Multicast
PKI	Public Key Infrastructure
PLR	Point of Local Repair
PMSI	Provider Multicast Service Instance
PSP	Penultimate Segment Popping
PST	Path Setup Type
PTA	PMSI Tunnel Attribute
PTP	Precision Time Protocol
PXC	Port Cross Connect
P2MP	Point to Multipoint
RI	Router Information
RLFA	Remote Loop-Free Alternate
RP	Request Parameters (Object)
RPC	Remote Procedure Call
RRO	Record Route Object
RSC	Return Sub-Code
RSVP	Resource Reservation Protocol
RTM	Route-Table Manager
SABM	Standard Application Bit Mask
SAFI	Subsequent Address Family Indicator
SBFD	Seamless Bidirectional Forwarding Detection
SCP	Secure Copy
SDN	Software Defined Networking
SDP	Service Distribution Point
SID	Segment Identifier
SL	Segments Left
SPF	Shortest Path First
SPRING	Source Packet Routing in Networking
SPT	Shortest Path Tree
SR	Segment Routing
SRGB	Segment Routing Global Block
SRH	Segment Routing Header
SRLB	Segment Routing Local Block
SRLG	Shared Risk Link Group
SRMS	Segment Routing Mapping Server
SRP	Stateful Request Parameters (Object)
SR-TE	Segment Routing Traffic Engineering
SRv6	Segment Routing Version 6
SR-OS	Service Router Operating System
SSM	Source-Specific Multicast
STAMP	Simple Two-Way Active Measurement Protocol

SVEC	Synchronisation Vector (Object)
S-PMSI	Selective Provider Multicast Service Instance
TCP	Transmission Control Protocol
TE	Traffic Engineering
TI-LFA	Topology Independent Loop-Free Alternate
TLS	Transport Layer Security
TLV	Type Length Value
TTM	Tunnel Table Manager
TWAMP	Two-Way Active Measurement Protocol
UDABM	User Defined Application Bit Mask
UDP	User Datagram Protocol
USD	Ultimate Segment Decapsulation
USID	Micro Segment Identifier
USP	Ultimate Segment Popping
VPRN	Virtual Private Routed Network
VPLS	Virtual Private LAN Service
VPWS	Virtual Private Wire Service

Change Control

Date	Version	Remarks
7th February 2022	1.0	Initial document creation
13th March 2022	1.1	Incorporate SR-OS 22.2.R1 features (minor) General updates and housekeeping
31st May 2022	1.2	Incorporate SR-OS 22.5.R1 features (minor) General updates and housekeeping
29th August 2022	1.3	Incorporate SR-OS 22.7.R1 features (more SRv6 services, dual Protected/Unprotected Adj-SIDs, SRv6 Policies). General clean-up of nits.
28th October 2022	1.4	Incorporate SR-OS 22.10.R1 features (SRv6 uSID, generic SRv6 support on 7250 IXR gen 2/2c. Backward compatibility of the PCEP SR-PCE-Capability TLV. Echo mode S-BFD for MPLS-based SR Policy

NOKIA