



## SR-TE Policy Overview

Segment routing for traffic engineering (SR-TE) uses a “policy” to steer traffic through the network. An SR-TE policy path is expressed as a list of segments that specifies the path, called a segment ID (SID) list. Each segment is an end-to-end path from the source to the destination, and instructs the routers in the network to follow the specified path instead of following the shortest path calculated by the IGP. If a packet is steered into an SR-TE policy, the SID list is pushed on the packet by the head-end. The rest of the network executes the instructions embedded in the SID list.

An SR-TE policy is identified as an ordered list (head-end, color, end-point):

- Head-end – Where the SR-TE policy is instantiated
- Color – A numerical value that distinguishes between two or more policies to the same node pairs (Head-end – End point)
- End-point – The destination of the SR-TE policy

Every SR-TE policy has a color value. Every policy between the same node pairs requires a unique color value.

An SR-TE policy uses one or more candidate paths. A candidate path is a single segment list (SID-list) or a set of weighted SID-lists (for weighted equal cost multi-path [WECMP]). A candidate path is either dynamic or explicit. See *SR-TE Policy Path Types* section for more information.

- [Auto-Route Announce for SR-TE, on page 1](#)
- [Instantiation of an SR Policy, on page 2](#)
- [SR-TE Policy Path Types, on page 2](#)
- [Protocols, on page 11](#)
- [Traffic Steering, on page 14](#)

## Auto-Route Announce for SR-TE

Auto-route announce for SR-TE cannot handle LDP-over-SR-TE if the SR-TE terminates at an LDP mid-node.

Let us consider the following topology:

R1---R2---R3---R4---R5---R6

If there is an SR-TE route from R1 to R4, and an LDP prefix is learnt from R6, then auto-route announce will fail.

# Instantiation of an SR Policy

An SR policy is instantiated, or implemented, at the head-end router.

The following sections provide details on the SR policy instantiation methods:

- [Manually Provisioned SR Policy, on page 2](#)

## Manually Provisioned SR Policy

Manually provisioned SR policies are configured on the head-end router. These policies can use dynamic paths or explicit paths. See the [SR-TE Policy Path Types, on page 2](#) section for information on manually provisioning an SR policy using dynamic or explicit paths.

## SR-TE Policy Path Types

A **dynamic** path is based on an optimization objective and a set of constraints. The head-end computes a solution, resulting in a SID-list or a set of SID-lists. When the topology changes, a new path is computed. If the head-end does not have enough information about the topology, the head-end might delegate the computation to a Segment Routing Path Computation Element (SR-PCE). For information on configuring SR-PCE, see *Configure Segment Routing Path Computation Element* chapter.

An **explicit** path is a specified SID-list or set of SID-lists.

An SR-TE policy initiates a single (selected) path in RIB/FIB. This is the preferred valid candidate path.

A candidate path has the following characteristics:

- It has a preference – If two policies have same {color, endpoint} but different preferences, the policy with the highest preference is selected.
- It is associated with a single binding SID (BSID) – A BSID conflict occurs when there are different SR policies with the same BSID. In this case, the policy that is installed first gets the BSID and is selected.
- It is valid if it is usable.

A path is selected when the path is valid and its preference is the best among all candidate paths for that policy.



---

**Note** The protocol of the source is not relevant in the path selection logic.

---

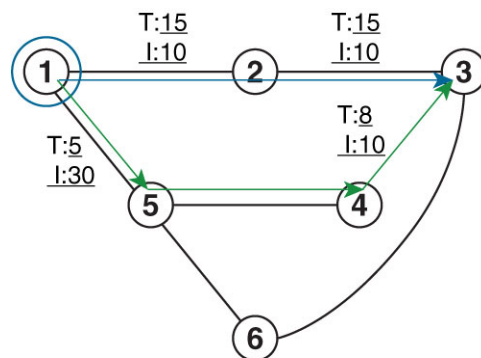
## Dynamic Paths

### Optimization Objectives

Optimization objectives allow the head-end router to compute a SID-list that expresses the shortest dynamic path according to the selected metric type:

- IGP metric — Refer to the "Implementing IS-IS" and "Implementing OSPF" chapters in the *Routing Configuration Guide for Series Routers*.
- TE metric — See the [Configure Interface TE Metrics, on page 3](#) section for information about configuring TE metrics.

This example shows a dynamic path from head-end router 1 to end-point router 3 that minimizes IGP or TE metric:



Default IGP link metric: I:10  
Default TE link metric T:10

520018

- The blue path uses the minimum IGP metric: Min-Metric (1 → 3, IGP) = SID-list <16003>; cumulative IGP metric: 20
- The green path uses the minimum TE metric: Min-Metric (1 → 3, TE) = SID-list <16005, 16004, 16003>; cumulative TE metric: 23

## Configure Interface TE Metrics

Use the **metric value** command in SR-TE interface submode to configure the TE metric for interfaces. The **value** range is from 0 to 2147483647.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface type interface-path-id
Router(config-sr-te-if)# metric value
```

### Configuring TE Metric: Example

The following configuration example shows how to set the TE metric for various interfaces:

```
segment-routing
traffic-eng
interface TenGigE0/0/0/0
metric 100
!
interface TenGigE0/0/0/1
metric 1000
!
interface TenGigE0/0/2/0
metric 50
!
!
end
```

## Constraints

Constraints allow the head-end router to compute a dynamic path according to the selected metric type:

- **TE affinity** — You can apply a color or name to links or interfaces by assigning affinity bit-maps to them. You can then specify an affinity (or relationship) between an SR policy path and link colors. SR-TE computes a path that includes or excludes links that have specific colors, or combinations of colors. See the [Named Interface Link Admin Groups and SR-TE Affinity Maps, on page 4](#) section for information on named interface link admin groups and SR-TE Affinity Maps.
- **Disjoint** — SR-TE computes a path that is disjoint from another path in the same disjoint-group. Disjoint paths do not share network resources. Path disjointness may be required for paths between the same pair of nodes, between different pairs of nodes, or a combination (only same head-end or only same end-point).
- **Flexible Algorithm** — Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

### Named Interface Link Admin Groups and SR-TE Affinity Maps

Named Interface Link Admin Groups and SR-TE Affinity Maps provide a simplified and more flexible means of configuring link attributes and path affinities to compute paths for SR-TE policies.

In the traditional TE scheme, links are configured with attribute-flags that are flooded with TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

Named Interface Link Admin Groups and SR-TE Affinity Maps let you assign, or map, up to 32 color names for affinity and attribute-flag attributes instead of 32-bit hexadecimal numbers. After mappings are defined, the attributes can be referred to by the corresponding color name in the CLI. Furthermore, you can define constraints using *include-any*, *include-all*, and *exclude-any* arguments, where each statement can contain up to 10 colors.




---

**Note** You can configure affinity constraints using attribute flags or the Flexible Name Based Policy Constraints scheme; however, when configurations for both schemes exist, only the configuration pertaining to the new scheme is applied.

---

### Configure Named Interface Link Admin Groups and SR-TE Affinity Maps

Use the **affinity name NAME** command in SR-TE interface submode to assign affinity to interfaces. Configure this on routers with interfaces that have an associated admin group attribute.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface TenGigE0/0/1/2
Router(config-sr-if)# affinity
Router(config-sr-if-affinity)# name RED
```

Use the **affinity-map name NAME bit-position bit-position** command in SR-TE sub-mode to define affinity maps. The *bit-position* range is from 0 to 255.

Configure affinity maps on the following routers:

- Routers with interfaces that have an associated admin group attribute.
- Routers that act as SR-TE head-ends for SR policies that include affinity constraints.

```

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# name RED bit-position 23

```

### Configuring Link Admin Group: Example

The following example shows how to assign affinity to interfaces and to define affinity maps. This configuration is applicable to any router (SR-TE head-end or transit node) with colored interfaces.

```

segment-routing
traffic-eng
interface TenGigE0/0/1/1
  affinity
  name CROSS
  name RED
  !
!
interface TenGigE0/0/1/2
  affinity
  name RED
  !
!
interface TenGigE0/0/2/0
  affinity
  name BLUE
  !
!
affinity-map
  name RED bit-position 23
  name BLUE bit-position 24
  name CROSS bit-position 25
  !
end

```

## Configure SR Policy with Dynamic Path

To configure a SR-TE policy with a dynamic path, optimization objectives, and affinity constraints, complete the following configurations:

1. Define the optimization objectives. See the [Optimization Objectives, on page 2](#) section.
2. Define the constraints. See the [Constraints, on page 4](#) section.
3. Create the policy.

### Behaviors and Limitations

The following example shows a configuration of an SR policy at an SR-TE head-end router. The policy has a dynamic path with optimization objectives and affinity constraints computed by the head-end router.

```

segment-routing
traffic-eng
policy foo
  color 100 end-point ipv4 1.1.1.2
  candidate-paths
  preference 100
  dynamic

```



When configuring an explicit path using IP addresses of intermediate links, the SR-TE process selects either the protected or the unprotected Adj-SID of the link, depending on the order in which the Adj-SIDs were received.

### Configure Local SR-TE Policy Using Explicit Paths

Create a segment list with IP addresses:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name SIDLIST1
Router(config-sr-te-sl)# index 10 address ipv4 1.1.1.2
Router(config-sr-te-sl)# index 20 address ipv4 1.1.1.3
Router(config-sr-te-sl)# index 30 address ipv4 1.1.1.4
Router(config-sr-te-sl)# exit
```

Create a segment list with MPLS labels:

```
Router(config-sr-te)# segment-list name SIDLIST2
Router(config-sr-te-sl)# index 10 mpls label 16002
Router(config-sr-te-sl)# index 20 mpls label 16003
Router(config-sr-te-sl)# index 30 mpls label 16004
Router(config-sr-te-sl)# exit
```

Create a segment list with IP addresses and MPLS labels:

```
Router(config-sr-te)# segment-list name SIDLIST3
Router(config-sr-te-sl)# index 10 address ipv4 1.1.1.2
Router(config-sr-te-sl)# index 20 mpls label 16003
Router(config-sr-te-sl)# index 30 mpls label 16004
Router(config-sr-te-sl)# exit
```

Create the SR-TE policy:

```
Router(config-sr-te)# policy POLICY1
Router(config-sr-te-policy)# color 10 end-point ipv4 1.1.1.4
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST1
Router(config-sr-te-policy-path-pref)# exit
Router(config-sr-te-pp-info)# exit
```

```
Router(config-sr-te)# policy POLICY2
Router(config-sr-te-policy)# color 20 end-point ipv4 1.1.1.4
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST2
Router(config-sr-te-policy-path-pref)# exit
Router(config-sr-te-pp-info)# exit
```

```
Router(config-sr-te)# policy POLICY3
Router(config-sr-te-policy)# color 30 end-point ipv4 1.1.1.4
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST3
Router(config-sr-te-policy-path-pref)# commit
```

## Running Configuration

```

Router# show running-configuration
segment-routing
traffic-eng
  segment-list SIDLIST1
    index 10 address ipv4 1.1.1.2
    index 20 address ipv4 1.1.1.3
    index 30 address ipv4 1.1.1.4
  !
  segment-list SIDLIST2
    index 10 mpls label 16002
    index 20 mpls label 16003
    index 30 mpls label 16004
  !
  segment-list SIDLIST3
    index 10 address ipv4 1.1.1.2
    index 20 mpls label 16003
    index 30 mpls label 16004
  !
  policy POLICY1
    color 10 end-point ipv4 1.1.1.4
    candidate-paths
      preference 100
      explicit segment-list SIDLIST1
    !
  !
  !
  policy POLICY2
    color 20 end-point ipv4 1.1.1.4
    candidate-paths
      preference 100
      explicit segment-list SIDLIST2
    !
  !
  !
  policy POLICY3
    color 30 end-point ipv4 1.1.1.4
    candidate-paths
      preference 100
      explicit segment-list SIDLIST3
    !
  !
  !

```

## Verification

```

Router# show segment-routing traffic-eng policy name srte_c_20_ep_1.1.1.4
Sat Jul  8 12:25:34.114 UTC
SR-TE policy database
-----
Name: P1 (Color: 20, End-point: 1.1.1.4)
Status:
  Admin: up Operational: up for 00:06:21 (since Jul  8 12:19:13.198)
Candidate-paths:
  Preference 10:
    Explicit: segment-list SIDLIST1 (active)
      Weight: 2
      400102 [Prefix-SID, 2.1.1.1]
      400106
    Explicit: segment-list SIDLIST2 (active)
      Weight: 2

```



```

400222 [Prefix-SID, 22.11.1.1]
400106
Attributes:
  Binding SID: 15001
  Allocation mode: explicit
  State: programmed
  Policy selected: yes
  Forward Class: 0

```

## Configuring Explicit Path with Affinity Constraint Validation

To fully configure SR-TE flexible name-based policy constraints, you must complete these high-level tasks in order:

1. Assign Color Names to Numeric Values
2. Associate Affinity-Names with SR-TE Links
3. Associate Affinity Constraints for SR-TE Policies

```

/* Enter the global configuration mode and assign color names to numeric values
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# blue bit-position 0
Router(config-sr-te-affinity-map)# green bit-position 1
Router(config-sr-te-affinity-map)# red bit-position 2
Router(config-sr-te-affinity-map)# exit

```

```

/* Associate affinity-names with SR-TE links
Router(config-sr-te)# interface Gi0/0/0/0
Router(config-sr-te-if)# affinity
Router(config-sr-te-if-affinity)# blue
Router(config-sr-te-if-affinity)# exit
Router(config-sr-te-if)# exit
Router(config-sr-te)# interface Gi0/0/0/1
Router(config-sr-te-if)# affinity
Router(config-sr-te-if-affinity)# blue
Router(config-sr-te-if-affinity)# green
Router(config-sr-te-if-affinity)# exit
Router(config-sr-te-if)# exit
Router(config-sr-te)#

```

```

/* Associate affinity constraints for SR-TE policies
Router(config-sr-te)# segment-list name SIDLIST1
Router(config-sr-te-sl)# index 10 address ipv4 1.1.1.2
Router(config-sr-te-sl)# index 20 address ipv4 2.2.2.23
Router(config-sr-te-sl)# index 30 address ipv4 1.1.1.4
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list name SIDLIST2
Router(config-sr-te-sl)# index 10 address ipv4 1.1.1.2
Router(config-sr-te-sl)# index 30 address ipv4 1.1.1.4
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list name SIDLIST3
Router(config-sr-te-sl)# index 10 address ipv4 1.1.1.5
Router(config-sr-te-sl)# index 30 address ipv4 1.1.1.4
Router(config-sr-te-sl)# exit

```

```

Router(config-sr-te)# policy POLICY1
Router(config-sr-te-policy)# color 20 end-point ipv4 1.1.1.4
Router(config-sr-te-policy)# binding-sid mpls 1000
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 200
Router(config-sr-te-policy-path-pref)# constraints affinity exclude-any red
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST1
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# exit
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST3

```

## Running Configuration

```

Router# show running-configuration
segment-routing
 traffic-eng

 interface GigabitEthernet0/0/0/0
  affinity
   blue
  !
 !
 interface GigabitEthernet0/0/0/1
  affinity
   blue
   green
  !
 !

 segment-list name SIDLIST1
  index 10 address ipv4 1.1.1.2
  index 20 address ipv4 2.2.2.23
  index 30 address ipv4 1.1.1.4
 !
 segment-list name SIDLIST2
  index 10 address ipv4 1.1.1.2
  index 30 address ipv4 1.1.1.4
 !
 segment-list name SIDLIST3
  index 10 address ipv4 1.1.1.5
  index 30 address ipv4 1.1.1.4
 !
 policy POLICY1
  binding-sid mpls 1000
  color 20 end-point ipv4 1.1.1.4
  candidate-paths
  preference 100
   explicit segment-list SIDLIST3
  !
 !
  preference 200
   explicit segment-list SIDLIST1
  !
   explicit segment-list SIDLIST2
  !
  constraints
   affinity

```

```

        exclude-any
        red
        !
        !
        !
        !
        !
        !
        affinity-map
        blue bit-position 0
        green bit-position 1
        red bit-position 2
        !
        !
        !

```

## Protocols

### Path Computation Element Protocol

The path computation element protocol (PCEP) describes a set of procedures by which a path computation client (PCC) can report and delegate control of head-end label switched paths (LSPs) sourced from the PCC to a PCE peer. The PCE can request the PCC to update and modify parameters of LSPs it controls. The stateful model also enables a PCC to allow the PCE to initiate computations allowing the PCE to perform network-wide orchestration.

### BGP SR-TE

SR-TE can be used by data center (DC) operators to provide different levels of Service Level Assurance (SLA). Setting up SR-TE paths using BGP (BGP SR-TE) simplifies DC network operation without introducing a new protocol for this purpose.

#### Explicit BGP SR-TE

Explicit BGP SR-TE uses an SR-TE policy (identified by a unique color ID) that contains a list of explicit paths with SIDs that correspond to each explicit path. A BGP speaker signals an explicit SR-TE policy to a remote peer, which triggers the setup of an SR-TE policy with specific characteristics and explicit paths. On the receiver side, an SR-TE policy that corresponds to the explicit path is setup by BGP. The packets for the destination mentioned in the BGP update follow the explicit path described by the policy. Each policy can include multiple explicit paths, and TE will create a policy for each path.




---

**Note** For more information on routing policies and routing policy language (RPL), refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco NCS 540 Series Routers*.

---

IPv4 and IPv6 SR policies can be advertised over BGPv4 or BGPv6 sessions between the SR-TE controller and the SR-TE headend. The Cisco IOS-XR implementation supports the following combinations:

- IPv4 SR policy advertised over BGPv4 session

- IPv6 SR policy advertised over BGPv4 session
- IPv6 SR policy advertised over BGPv6 session

## Configure Explicit BGP SR-TE

Perform this task to configure explicit BGP SR-TE:

### Procedure

	Command or Action	Purpose
<b>Step 1</b>	<b>configure</b>	
<b>Step 2</b>	<b>router bgp</b> <i>as-number</i> <b>Example:</b> RP/0/RP0/CPU0:router(config)# <b>router bgp</b> <b>65000</b>	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
<b>Step 3</b>	<b>bgp router-id</b> <i>ip-address</i> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp)# <b>bgp</b> <b>router-id 1.1.1.1</b>	Configures the local router with a specified router ID.
<b>Step 4</b>	<b>address-family</b> { <b>ipv4</b>   <b>ipv6</b> } <b>sr-policy</b> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp)# <b>address-family ipv4 sr-policy</b>	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
<b>Step 5</b>	<b>exit</b>	
<b>Step 6</b>	<b>neighbor</b> <i>ip-address</i> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp)# <b>neighbor 10.10.0.1</b>	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
<b>Step 7</b>	<b>remote-as</b> <i>as-number</i> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp-nbr)# <b>remote-as 1</b>	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
<b>Step 8</b>	<b>address-family {ipv4   ipv6} sr-policy</b> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp-nbr)# <b>address-family ipv4 sr-policy</b>	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
<b>Step 9</b>	<b>route-policy route-policy-name {in   out}</b> <b>Example:</b> RP/0/RP0/CPU0:router(config-bgp-nbr-af)# <b>route-policy pass out</b>	Applies the specified policy to IPv4 or IPv6 unicast routes.

#### Example: BGP SR-TE with BGPv4 Neighbor to BGP SR-TE Controller

The following configuration shows the an SR-TE head-end with a BGPv4 session towards a BGP SR-TE controller. This BGP session is used to signal both IPv4 and IPv6 SR policies.

```
router bgp 65000
  bgp router-id 1.1.1.1
  !
  address-family ipv4 sr-policy
  !
  address-family ipv6 sr-policy
  !
  neighbor 10.1.3.1
    remote-as 10
    description *** eBGP session to BGP SRTE controller ***
    address-family ipv4 sr-policy
      route-policy pass in
      route-policy pass out
    !
    address-family ipv6 sr-policy
      route-policy pass in
      route-policy pass out
    !
  !
  !
  !
```

#### Example: BGP SR-TE with BGPv6 Neighbor to BGP SR-TE Controller

The following configuration shows an SR-TE head-end with a BGPv6 session towards a BGP SR-TE controller. This BGP session is used to signal IPv6 SR policies.

```
router bgp 65000
  bgp router-id 1.1.1.1
  address-family ipv6 sr-policy
  !
  neighbor 3001::10:1:3:1
    remote-as 10
    description *** eBGP session to BGP SRTE controller ***
    address-family ipv6 sr-policy
      route-policy pass in
      route-policy pass out
    !
  !
  !
  !
```

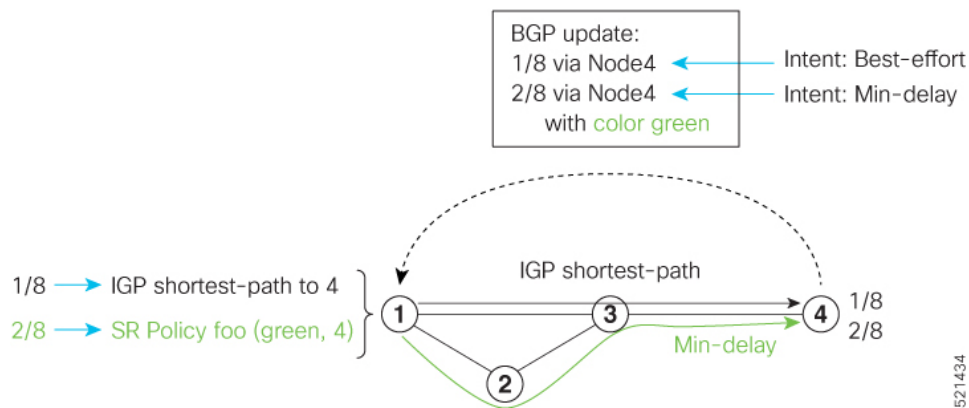
# Traffic Steering

## Automated Steering

Automated steering (AS) allows service traffic to be automatically steered onto the required transport SLA path programmed by an SR policy.

With AS, BGP automatically steers traffic onto an SR Policy based on the next-hop and color of a BGP service route. The color of a BGP service route is specified by a color extended community attribute. This color is used as a transport SLA indicator, such as min-delay or min-cost.

When the next-hop and color of a BGP service route matches the end-point and color of an SR Policy, BGP automatically installs the route resolving onto the BSID of the matching SR Policy. Recall that an SR Policy on a head-end is uniquely identified by an end-point and color.



When a BGP route has multiple extended-color communities, each with a valid SR Policy, the BGP process installs the route on the SR Policy giving preference to the color with the highest numerical value.

The granularity of AS behaviors can be applied at multiple levels, for example:

- At a service level—When traffic destined to all prefixes in a given service is associated to the same transport path type. All prefixes share the same color.
- At a destination/prefix level—When traffic destined to a prefix in a given service is associated to a specific transport path type. Each prefix could be assigned a different color.
- At a flow level—When flows destined to the same prefix are associated with different transport path types

AS behaviors apply regardless of the instantiation method of the SR policy, including:

- On-demand SR policy
- Manually provisioned SR policy
- PCE-initiated SR policy

## Using Binding Segments

The binding segment is a local segment identifying an SR-TE policy. Each SR-TE policy is associated with a binding segment ID (BSID). The BSID is a local label that is automatically allocated for each SR-TE policy when the SR-TE policy is instantiated.

BSID can be used to steer traffic into the SR-TE policy and across domain borders, creating seamless end-to-end inter-domain SR-TE policies. Each domain controls its local SR-TE policies; local SR-TE policies can be validated and rerouted if needed, independent from the remote domain's head-end. Using binding segments isolates the head-end from topology changes in the remote domain.

Packets received with a BSID as top label are steered into the SR-TE policy associated with the BSID. When the BSID label is popped, the SR-TE policy's SID list is pushed.

BSID can be used in the following cases:

- Multi-Domain (inter-domain, inter-autonomous system)—BSIDs can be used to steer traffic across domain borders, creating seamless end-to-end inter-domain SR-TE policies.
- Large-Scale within a single domain—The head-end can use hierarchical SR-TE policies by nesting the end-to-end (edge-to-edge) SR-TE policy within another layer of SR-TE policies (aggregation-to-aggregation). The SR-TE policies are nested within another layer of policies using the BSIDs, resulting in seamless end-to-end SR-TE policies.
- Label stack compression—If the label-stack size required for an SR-TE policy exceeds the platform capability, the SR-TE policy can be seamlessly stitched to, or nested within, other SR-TE policies using a binding segment.
- BGP SR-TE Dynamic—The head-end steers the packet into a BGP-based FIB entry whose next hop is a binding-SID.

### Explicit Binding SID

Use the **binding-sid mpls label** command in SR-TE policy configuration mode to specify the explicit BSID. Explicit BSIDs are allocated from the segment routing local block (SRLB) or the dynamic range of labels. A best-effort is made to request and obtain the BSID for the SR-TE policy. If requested BSID is not available (if it does not fall within the available SRLB or is already used by another application or SR-TE policy), the policy stays down.

Use the **binding-sid explicit {fallback-dynamic | enforce-srlb}** command to specify how the BSID allocation behaves if the BSID value is not available.

- Fallback to dynamic allocation – If the BSID is not available, the BSID is allocated dynamically and the policy comes up:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# binding-sid explicit fallback-dynamic
```

- Strict SRLB enforcement – If the BSID is not within the SRLB, the policy stays down:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
```

```
Router(config-sr-te)# binding-sid explicit enforce-srlb
```

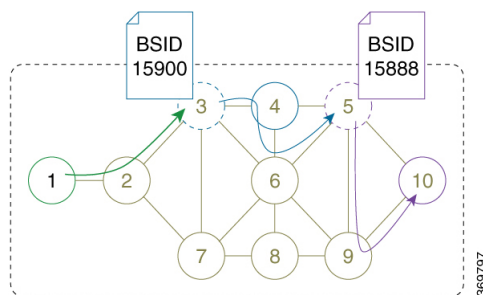
This example shows how to configure an SR policy to use an explicit BSID of 1000. If the BSID is not available, the BSID is allocated dynamically and the policy comes up.

```
segment-routing
traffic-eng
binding-sid explicit fallback-dynamic
policy goo
binding-sid mpls 1000
!
!
!
```

## Stitching SR-TE Polices Using Binding SID: Example

In this example, three SR-TE policies are stitched together to form a seamless end-to-end path from node 1 to node 10. The path is a chain of SR-TE policies stitched together using the binding-SIDs of intermediate policies, providing a seamless end-to-end path.

**Figure 1: Stitching SR-TE Polices Using Binding SID**



**Table 1: Router IP Address**

Router	Prefix Address	Prefix SID/Adj-SID
3	Loopback0 - 1.1.1.3	Prefix SID - 16003
4	Loopback0 - 1.1.1.4 Link node 4 to node 6 - 10.4.6.4	Prefix SID - 16004 Adjacency SID - dynamic
5	Loopback0 - 1.1.1.5	Prefix SID - 16005
6	Loopback0 - 1.1.1.6 Link node 4 to node 6 - 10.4.6.6	Prefix SID - 16006 Adjacency SID - dynamic
9	Loopback0 - 1.1.1.9	Prefix SID - 16009
10	Loopback0 - 1.1.1.10	Prefix SID - 16010



## Procedure

### Step 1

On node 5, do the following:

- a) Define an SR-TE policy with an explicit path configured using the loopback interface IP addresses of node 9 and node 10.
- b) Define an explicit binding-SID (**mpls label 15888**) allocated from SRLB for the SR-TE policy.

#### Example:

#### Node 5

```

segment-routing
traffic-eng
segment-list PATH-9_10
index 10 address ipv4 1.1.1.9
index 20 address ipv4 1.1.1.10
!
policy foo
binding-sid mpls 15888
color 777 end-point ipv4 1.1.1.10
candidate-paths
preference 100
explicit segment-list PATH5-9_10
!
!
!
!
!
!
RP/0/RSP0/CPU0:Node-5# show segment-routing traffic-eng policy color 777

SR-TE policy database
-----

Color: 777, End-point: 1.1.1.10
Name: srte_c_777_ep_1.1.1.10
Status:
Admin: up Operational: up for 00:00:52 (since Aug 19 07:40:12.662)
Candidate-paths:
Preference: 100 (configuration) (active)
Name: foo
Requested BSID: 15888
PCC info:
Symbolic name: cfg_foo_discr_100
PLSP-ID: 70
Explicit: segment-list PATH-9_10 (valid)
Weight: 1, Metric Type: TE
16009 [Prefix-SID, 1.1.1.9]
16010 [Prefix-SID, 1.1.1.10]
Attributes:
Binding SID: 15888 (SRLB)
Forward Class: 0
Steering BGP disabled: no
IPv6 caps enable: yes

```

### Step 2

On node 3, do the following:

- a) Define an SR-TE policy with an explicit path configured using the following:
  - Loopback interface IP address of node 4

- Interface IP address of link between node 4 and node 6
- Loopback interface IP address of node 5
- Binding-SID of the SR-TE policy defined in Step 1 (**mpls label 15888**)

**Note** This last segment allows the stitching of these policies.

- b) Define an explicit binding-SID (**mpls label 15900**) allocated from SRLB for the SR-TE policy.

### Example:

#### Node 3

```
segment-routing
traffic-eng
segment-list PATH-4_4-6_5_BSID
index 10 address ipv4 1.1.1.4
index 20 address ipv4 10.4.6.6
index 30 address ipv4 1.1.1.5
index 40 mpls label 15888
!
policy baa
binding-sid mpls 15900
color 777 end-point ipv4 1.1.1.5
candidate-paths
preference 100
explicit segment-list PATH-4_4-6_5_BSID
!
!
!
!
!
!
RP/0/RSP0/CPU0:Node-3# show segment-routing traffic-eng policy color 777

SR-TE policy database
-----

Color: 777, End-point: 1.1.1.5
Name: srte_c_777_ep_1.1.1.5
Status:
Admin: up Operational: up for 00:00:32 (since Aug 19 07:40:32.662)
Candidate-paths:
Preference: 100 (configuration) (active)
Name: baa
Requested BSID: 15900
PCC info:
Symbolic name: cfg_baa_discr_100
PLSP-ID: 70
Explicit: segment-list PATH-4_4-6_5_BSID (valid)
Weight: 1, Metric Type: TE
16004 [Prefix-SID, 1.1.1.4]
80005 [Adjacency-SID, 10.4.6.4 - 10.4.6.6]
16005 [Prefix-SID, 1.1.1.5]
15888
Attributes:
Binding SID: 15900 (SRLB)
Forward Class: 0
Steering BGP disabled: no
IPv6 caps enable: yes
```

**Step 3** On node 1, define an SR-TE policy with an explicit path configured using the loopback interface IP address of node 3 and the binding-SID of the SR-TE policy defined in step 2 (**mpls label 15900**). This last segment allows the stitching of these policies.

**Example:**

**Node 1**

```
segment-routing
traffic-eng
segment-list PATH-3_BSID
index 10 address ipv4 1.1.1.3
index 20 mpls label 15900
!
policy bar
color 777 end-point ipv4 1.1.1.3
candidate-paths
preference 100
explicit segment-list PATH-3_BSID
!
!
!
!
!
```

```
RP/0/RSP0/CPU0:Node-1# show segment-routing traffic-eng policy color 777
```

```
SR-TE policy database
```

```
-----
Color: 777, End-point: 1.1.1.3
Name: srte_c_777_ep_1.1.1.3
Status:
  Admin: up Operational: up for 00:00:12 (since Aug 19 07:40:52.662)
Candidate-paths:
  Preference: 100 (configuration) (active)
  Name: bar
  Requested BSID: dynamic
  PCC info:
    Symbolic name: cfg_bar_discr_100
    PLSP-ID: 70
  Explicit: segment-list PATH-3_BSID (valid)
    Weight: 1, Metric Type: TE
    16003 [Prefix-SID, 1.1.1.3]
    15900
Attributes:
  Binding SID: 80021
  Forward Class: 0
  Steering BGP disabled: no
  IPv6 caps enable: yes
```

## L2VPN Preferred Path

EVPN VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for EVPN VPWS pseudowire (PW) using SR-TE policy.

L2VPN VPLS or VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for L2VPN Virtual Private LAN Service (VPLS) or Virtual Private Wire Service (VPWS) using SR-TE policy.

Refer to the [EVPN VPWS Preferred Path over SR-TE Policy](#) and [L2VPN VPLS or VPWS Preferred Path over SR-TE Policy](#) sections in the "L2VPN Services over Segment Routing for Traffic Engineering Policy" chapter of the *L2VPN and Ethernet Services Configuration Guide*.