Configure SR-TE Policies

This module provides information about segment routing for traffic engineering (SR-TE) policies, how to configure SR-TE policies, and how to steer traffic into an SR-TE policy.

Note
Configuring SR-TE policies with 3 or more labels and an L2 Transport Interface on the same network processing unit (NPU) can cause traffic loss.

- Configure SR-TE Policies, on page 1
- BGP SR-TE, on page 6
- Using Binding Segments, on page 9

Configure SR-TE Policies

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.

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 path computation engine (PCE). For information on configuring the XTC as a PCE, see the Configure IOS XR Traffic Controller (XTC) 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.

Configuration Example

To configure a local SR-TE policy, you must complete the following configurations:

1. Create the segment lists.
2. Create the policy.

Configure Local SR-TE Policy

/* Enter the global configuration mode and create the SR-TE segment lists */
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name Plist-1
Router(config-sr-te-sl)# index 1 mpls label 400102
Router(config-sr-te-sl)# index 2 mpls label 400106
Router(config-sr-te-sl)# exit

Router(config-sr-te)# segment-list name Plist-2
Router(config-sr-te-sl)# index 1 mpls label 400222
Router(config-sr-te-sl)# index 2 mpls label 400106
Router(config-sr-te-sl)# exit

/* Create the SR-TE policy */
Router(config-sr-te)# policy P1
Router(config-sr-te-policy)# binding-sid mpls 15001
Router(config-sr-te-policy)# color 1 end-point ipv4 6.6.6.6
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 10
Router(config-sr-te-pp-index)# explicit segment-list Plist-1
Router(config-sr-te-pp-info)# weight 2
Router(config-sr-te-pp-info)# exit
Configure SR-TE Policies

Running Configuration

Router# show running-configuration
segment-routing
  traffic-eng
    segment-list name Plist-1
      index 1 mpls label 400102
      index 2 mpls label 400106
    !
    segment-list name Plist-2
      index 1 mpls label 400222
      index 2 mpls label 400106
    !
  policy P1
    binding-sid mpls 15001
    color 1 end-point ipv4 6.6.6.6
    candidate-paths
      preference 10
        explicit segment-list Plist-1
          weight 2
        ! explicit segment-list Plist-2
          weight 2
        !
  !

Verification

Router# show segment-routing traffic-eng policy name srte_c_1_ep_6.6.6.6
Sat Jul 8 12:25:34.114 UTC
SR-TE policy database
---------------------
Name: P1 (Color: 1, End-point: 6.6.6.6)
Status:
  Admin: up Operational: up for 00:06:21 (since Jul 8 12:19:13.198)
Candidate-paths:
  Preference 10:
    Explicit: segment-list Plist-1 (active)
      Weight: 2
    400102 [Prefix-SID, 2.1.1.1]
    400106
    Explicit: segment-list Plist-2 (active)
      Weight: 2
    400222 [Prefix-SID, 22.11.1.1]
    400106
Attributes:
  Binding SID: 15001
  Allocation mode: explicit
  State: programmed
**Autoroute Include**

You can configure SR-TE policies with Autoroute Include to steer specific IGP (IS-IS, OSPF) prefixes over non-shortest paths and to divert the traffic for those prefixes on to the SR-TE policy. Autoroute Include applies Autoroute Announce functionality to the specified destinations or prefixes.

The Autoroute SR-TE policy adds the prefixes into the IGP, which determines if the prefixes on the endpoint or downstream of the endpoint are eligible to use the SR-TE policy. If a prefix is eligible, then the IGP checks if the prefix is listed in the Autoroute Include configuration. If the prefix is included, then the IGP downloads the prefix route with the SR-TE policy as the outgoing path.

Autoroute Include supports three metric types:

- **Default (no metric):** The path over the SR-TE policy inherits the shortest path metric.
- **Absolute metric:** The shortest path metric to the policy endpoint is replaced with the configured absolute metric. The metric to any prefix that is Autoroute Included is modified to the absolute metric.
- **Relative metric:** The shortest path metric to the policy endpoint is modified with the relative value configured (plus or minus).

---

**Note**

To prevent load-balancing over IGP paths, you can specify a metric that is lower than the value that IGP takes into account for autorouted destinations (for example, `autoroute metric relative -1`).

---

**Configuration Example**

```plaintext
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P1
Router(config-sr-te-policy)# color 20 end ipv4 1.1.1.2
Router(config-sr-te-policy)# autoroute include ipv4 1.1.1.21/32
Router(config-sr-te-policy)# autoroute include ipv4 1.1.1.23/32
Router(config-sr-te-policy)# autoroute metric constant 1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-pp-index)# explicit segment-list Plist-1
```

---

**Color-Only Steering**

Color-only steering is a traffic steering mechanism where a policy is created with given color, regardless of the endpoint.

You can create an SR-TE policy for a specific color that uses a NULL end-point (0.0.0.0 for IPv4 NULL, and ::0 for IPv6 NULL end-point). This means that you can have a single policy that can steer traffic that is based on that color and a NULL endpoint for routes with a particular color extended community, but different destinations (next-hop).
Every SR-TE policy with a NULL end-point must have an explicit path-option. The policy cannot have a dynamic path-option (where the path is computed by the head-end or PCE) since there is no destination for the policy.

You can also specify a color-only (CO) flag in the color extended community for overlay routes. The CO flag allows the selection of an SR-policy with a matching color, regardless of endpoint Sub-address Family Identifier (SAFI) (IPv4 or IPv6). See Setting CO Flag, on page 9.

**Configure Color-Only Steering**

```plaintext
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P1
Router(config-sr-te-policy)# color 1 end-point ipv4 0.0.0.0

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P2
Router(config-sr-te-policy)# color 2 end-point ipv6 ::
```

**Address-Family Agnostic Steering**

Address-family agnostic steering uses an SR-TE policy to steer both labeled and unlabeled IPv4 and IPv6 traffic. This feature requires support of IPv6 encapsulation (IPv6 caps) over IPv4 endpoint policy.

IPv6 caps for IPv4 NULL end-point is enabled automatically when the policy is created in XR Traffic Controller (XTC). The binding SID (BSID) state notification for each policy contains an "ipv6_caps" flag that notifies XTC clients of the status of IPv6 caps (enabled or disabled).

An SR-TE policy with a given color and IPv4 NULL end-point could have more than one candidate path. If any of the candidate paths has IPv6 caps enabled, then all of the remaining candidate paths need IPv6 caps enabled. If IPv6 caps is not enabled on all candidate paths of same color and end-point, traffic drops can occur.

You can disable IPv6 caps for a particular color and IPv4 NULL end-point using the `ipv6 disable` command on the local policy. This command disables IPv6 caps on all candidate paths that share the same color and IPv4 NULL end-point.
### 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 TE tunnel with specific characteristics and explicit paths. On the receiver side, a TE tunnel 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 tunnel 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 5500 Series Routers*.

---

### Configure Explicit BGP SR-TE

Perform this task to configure explicit BGP SR-TE:

#### SUMMARY STEPS

1. configure
2. extcommunity-set opaque name
3. name
4. end-set
5. route-policy route-policy-name
6. end-policy
7. router bgp as-number
8. bgp router-id ip-address
9. address-family {ipv4 | ipv6} sr-policy
10. exit
11. neighbor ip-address
12. remote-as as-number
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>extcommunity-name opaque name</td>
<td>Defines the color extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# extcommunity-name opaque color1</td>
</tr>
<tr>
<td>3.</td>
<td>name</td>
<td>Defines the color extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ext)# 1</td>
</tr>
<tr>
<td>4.</td>
<td>end-set</td>
<td>Ends the definition of the extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ext)# end-set</td>
</tr>
<tr>
<td>5.</td>
<td>route-policy route-policy-name</td>
<td>Creates a route policy and enters route policy configuration mode, where you can define the route policy to mark the prefixes with the color extended community value.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# route-policy color RP/0/RSP0/CPU0:router(config-rpl)# if destination in (5.5.5.1/32) then RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity color color1 RP/0/RSP0/CPU0:router(config-rpl-if)# endif RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
</tr>
<tr>
<td>6.</td>
<td>end-policy</td>
<td>Ends the definition of a route policy and exits route policy configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
</tr>
<tr>
<td>7.</td>
<td>router bgp as-number</td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 1</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>8</td>
<td>bgp router-id <em>ip-address</em></td>
<td>Configures the local router with a specified router ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.10.0.2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>address-family {ipv4</td>
<td>ipv6} sr-policy</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 sr-policy</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>exit</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>neighbor <em>ip-address</em></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.0.1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>remote-as <em>as-number</em></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>address-family {ipv4</td>
<td>ipv6} unicast</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>route-policy <em>route-policy-name</em> {in</td>
<td>out}</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr-asf)# route-policy color out</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>send-extended-community-ebgp</td>
<td>Sends extended community attributes to external Border Gateway Protocol (eBGP) neighbors.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-bgp-nbr-asf)# send-extended-community-ebgp</td>
<td></td>
</tr>
</tbody>
</table>
Setting CO Flag

The BGP-based steering mechanism matches BGP color and next-hop with that of an SR-TE policy. If the policy does not exist, BGP requests XTC to create an SR-TE policy with the associated color, end-point, and explicit paths. For color-only steering (NULL end-point), you can configure a color-only (CO) flag as part of the color extended community in BGP.

Note

See Color-Only Steering, on page 4 for information about color-only steering (NULL end-point).

The behavior of the steering mechanism is based on the following values of the CO flags:

| co-flag 00 | 1. The BGP next-hop and color <N, C> is matched with an SR-TE policy of same <N, C>.  
| | 2. If a policy does not exist, then IGP path for the next-hop N is chosen. |
| co-flag 01 | 1. The BGP next-hop and color <N, C> is matched with an SR-TE policy of same <N, C>.  
| | 2. If a policy does not exist, then an SR-TE policy with NULL end-point with the same address-family as N and color C is chosen.  
| | 3. If a policy with NULL end-point with same address-family as N does not exist, then an SR-TE policy with any NULL end-point and color C is chosen.  
| | 4. If no match is found, then IGP path for the next-hop N is chosen. |

Configuration Example

Router(config)# extcommunity-set opaque overlay-color  
Router(config-ext)# 1 co-flag 01  
Router(config-ext)# end-set  
Router(config)# router-policy color  
Router(config-rpl)# if destination in (5.5.5.1/32) then  
Router(config-rpl-if)# set extcommunity color overlay-color  
Router(config-rpl-if)# endif  
Router(config-rpl)# pass  
Router(config-rpl)# end-policy  
Router(config)#

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.
In Cisco IOS XR 6.3.2 and later releases, you can specify an explicit BSID for an SR-TE policy. See the following Explicit Binding SID section.

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.

**Explicit Binding SID**

Use the `binding-sid explicit {fallback-dynamic | enforce-srlb}` command to request that the SR-TE policy uses a BSID value that you provide. 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 this 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.

You can 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
```
**Stitching SR-TE Polices Using Binding SID: Example**

In this intra-domain example, three SR-TE policies are stitched together to form a seamless end-to-end path from node 1 to node 10.

*Figure 1: Intra-Domain Topology*

### Step 1
Configure an SR-TE policy on node 5 to node 10 via node 9. Node 5 automatically allocates a binding-SID (24012) for the SR-TE policy.

**Example:**

```conf
RP/0/0/CPU0:xrvr-5(config)# explicit-path name PATH5-9_10
RP/0/0/CPU0:xrvr-5(config-expl-path)# index 10 next-address strict ipv4 unicast 192.168.59.9
RP/0/0/CPU0:xrvr-5(config-expl-path)# index 20 next-address strict ipv4 unicast 10.1.1.10

RP/0/0/CPU0:xrvr-5(config)# interface tunnel-te1
RP/0/0/CPU0:xrvr-5(config-if)# ipv4 unnumbered Loopback0
RP/0/0/CPU0:xrvr-5(config-if)# destination 10.1.1.10
RP/0/0/CPU0:xrvr-5(config-if)# path-option 1 explicit name PATH5-9_10 segment-routing
RP/0/0/CPU0:xrvr-5(config-if)# commit
```

```
Name: tunnel-te1 Destination: 10.1.1.10 Ifhandle:0x680
Signalled-Name: xrvr-5_t1
Status: Admin: up Oper: up Path: valid Signalling: connected
   path option 1, (Segment-Routing) type dynamic (Basis for Setup, path weight 10)
<...>
   Binding SID: 24012
<...>
   Segment-Routing Path Info (IS-IS 1 level-2)
   Segment0[Link]: 192.168.59.5 - 192.168.59.9, Label: 24007
   Segment1[Node]: 10.1.1.10, Label: 16010
```

### Step 2
Configure an SR-TE policy on node 3 to node 5 via node 4 and Link4-6, and push the binding-SID of the SR-TE policy at node 5 (24012) to stitch to the SR-TE policy on node 5. Node 3 automatically allocates a binding-SID (24008) for this SR-TE policy.

**Example:**

```conf
RP/0/0/CPU0:xrvr-3(config)# explicit-path name PATH4_4-6_5_BSID
RP/0/0/CPU0:xrvr-3(config-expl-path)# index 10 next-address strict ipv4 unicast 10.1.1.4
```
Stitching SR-TE Policies Using Binding SID: Example

Step 3

Configure an SR-TE policy on node 1 to node 3 and push the binding-SID of the SR-TE policy at node 3 (24008) to stitch to the SR-TE policy on node 3.

Example:

```
RP/0/0/CPU0:xrvr-1(config)# explicit-path name PATH3_BSID
RP/0/0/CPU0:xrvr-1(config-expl-path)# index 10 next-address strict ipv4 unicast 10.1.1.3
RP/0/0/CPU0:xrvr-1(config-expl-path)# index 20 next-label 24008
RP/0/0/CPU0:xrvr-1(config-expl-path)# exit

RP/0/0/CPU0:xrvr-1(config)# interface tunnel-te1
RP/0/0/CPU0:xrvr-1(config-if)# ipv4 unnumbered Loopback0
RP/0/0/CPU0:xrvr-1(config-if)# destination 10.1.1.10
RP/0/0/CPU0:xrvr-1(config-if)# path-option 1 explicit name PATH3_BSID segment-routing
RP/0/0/CPU0:xrvr-1(config-if)# commit

RP/0/0/CPU0:xrvr-1# show mpls traffic-eng tunnels 1 detail
Name: tunnel-te1 Destination: 10.1.1.10 Ifhandle:0x2f80
Signalled-Name: xrvr-1_t1
Status:
  Admin:  up  Oper:  up  Path:  valid  Signalling: connected
  path option 1, (Segment-Routing) type explicit PATH3_BSID (Basis for Setup)
<...>
Binding SID: 24008
<...>
Segment-Routing Path Info (IS-IS 1 level-2)
  Segment0[Node]: 10.1.1.3, Label: 16003
  Segment1[- ]: Label: 24008
```

The path is a chain of SR-TE policies stitched together using the binding-SIDs, providing a seamless end-to-end path.
RP/0/0/CPU0:xrvr-1# traceroute 10.1.1.10
Type escape sequence to abort.
Tracing the route to 10.1.1.10
1 99.1.2.2 [MPLS: Labels 16003/24008 Exp 0] 29 msec 19 msec 19 msec
2 99.2.3.3 [MPLS: Label 24008 Exp 0] 29 msec 19 msec 19 msec
3 99.3.4.4 [MPLS: Labels 24003/16005/24012 Exp 0] 29 msec 19 msec 19 msec
4 99.4.6.6 [MPLS: Labels 16005/24012 Exp 0] 29 msec 29 msec 19 msec
5 99.5.6.5 [MPLS: Label 24012 Exp 0] 29 msec 29 msec 19 msec
6 99.5.9.9 [MPLS: Label 16010 Exp 0] 19 msec 19 msec 19 msec
7 99.9.10.10 29 msec 19 msec 19 msec