Enabling Segment Routing Flexible Algorithm

Table 1: Feature History Table

Feature Name	**Release Information**	**Feature Description**
Segment Routing Flexible Algorithm	Release 24.4.1	Introduced in this release on: Fixed Systems(8700)(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) * The Segment Routing Flexible Algorithm functionality is now extended to the Cisco 8712-MOD-M routers.
Segment Routing Flexible Algorithm	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The Segment Routing flexible algorithm functionality is now extended to: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
Segment Routing Flexible Algorithm	Release 7.3.1	The Segment Routing architecture associates prefix-SIDs to an algorithm that defines how the path is computed. This feature allows for user-defined algorithms where the IGP computes paths based on a combination of metric type and constraint. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, this feature provides a traffic-engineered path computed automatically by the IGP to any destination reachable by the IGP. This release supports the following functionality: • TI-LFA (IS-IS/OSPF) • Microloop Avoidance (IS-IS) • Inter-AS Support (IS-IS) • SID Redistribution (IS-IS) • Metric minimization—avoidance, multi-plane, delay (IS-IS/OSPF) • Affinity include (IS-IS/OSPF) • Affinity exclude (IS-IS/OSPF)

Segment Routing Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

The SR architecture associates prefix-SIDs to an algorithm which defines how the path is computed. Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

This document describes the IS-IS and OSPF extensions to support Segment Routing Flexible Algorithm on an MPLS data-plane.

Prerequisites for Flexible Algorithm

Segment routing must be enabled on the router before the Flexible Algorithm functionality is activated.

Building Blocks of Segment Routing Flexible Algorithm

This section describes the building blocks that are required to support the SR Flexible Algorithm functionality in IS-IS and OSPF.

Flexible Algorithm Definition

Many possible constrains may be used to compute a path over a network. Some networks are deployed with multiple planes. A simple form of constrain may be to use a particular plane. A more sophisticated form of constrain can include some extended metric, like delay, as described in [RFC7810]. Even more advanced case could be to restrict the path and avoid links with certain affinities. Combinations of these are also possible. To provide a maximum flexibility, the mapping between the algorithm value and its meaning can be defined by the user. When all the routers in the domain have the common understanding what the particular algorithm value represents, the computation for such algorithm is consistent and the traffic is not subject to looping. Here, since the meaning of the algorithm is not defined by any standard, but is defined by the user, it is called as Flexible Algorithm.

Flexible Algorithm Support Advertisement

An algorithm defines how the best path is computed by IGP. Routers advertise the support for the algorithm as a node capability. Prefix-SIDs are also advertised with an algorithm value and are tightly coupled with the algorithm itself.

An algorithm is a one octet value. Values from 128 to 255 are reserved for user defined values and are used for Flexible Algorithm representation.

Flexible Algorithm Definition Advertisement

To guarantee the loop free forwarding for paths computed for a particular Flexible Algorithm, all routers in the network must share the same definition of the Flexible Algorithm. This is achieved by dedicated router(s) advertising the definition of each Flexible Algorithm. Such advertisement is associated with the priority to make sure that all routers will agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints

To enable the router to advertise the definition for the particular Flexible Algorithm, **advertise-definition** command is used. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. Without the valid definition being advertised, the Flexible Algorithm will not be functional.

Flexible Algorithm Prefix-SID Advertisement

To be able to forward traffic on a Flexible Algorithm specific path, all routers participating in the Flexible Algorithm will install a MPLS labeled path for the Flexible Algorithm specific SID that is advertised for the prefix. Only prefixes for which the Flexible Algorithm specific Prefix-SID is advertised is subject to Flexible Algorithm specific forwarding.

Calculation of Flexible Algorithm Path

A router may compute path for multiple Flexible Algorithms. A router must be configured to support particular Flexible Algorithm before it can compute any path for such Flexible Algorithm. A router must have a valid definition of the Flexible Algorithm before such Flexible Algorithm is used.

When computing the shortest path tree for particular Flexible Algorithm:

- All nodes that do not advertise support for such Flexible Algorithm will be pruned from the topology.
- If the Flexible Algorithm definition includes affinities that are excluded, then all links for which any of such affinities are advertised will be pruned from the topology.
- Router uses the metric that is part of the Flexible Algorithm definition. If the metric is not advertised for the particular link, such link will be pruned from the topology.

IS-IS supports Loop Free Alternate (LFA) paths, TI-LFA backup paths, and Microloop Avoidance paths for particular Flexible Algorithm. OSPF supports Loop Free Alternate (LFA) and TI-LFA backup paths for particular Flexible Algorithm. These paths are computed using the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use Prefix-SIDs advertised specifically for such Flexible Algorithm in order to enforce a backup or microloop avoidance path.

Installation of Forwarding Entries for Flexible Algorithm Paths

Flexible Algorithm path to any prefix must be installed in the forwarding using the Prefix-SID that was advertised for such Flexible Algorithm. If the Prefix-SID for Flexible Algorithm is not known, such Flexible Algorithm path is not installed in forwarding for such prefix..

Only MPLS to MPLS entries are installed for a Flexible Algorithm path. No IP to IP or IP to MPLS entries are installed. These follow the native IGP paths computed based on the default algorithm and regular IGP metrics.

Flexible Algorithm Prefix-SID Redistribution

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Algorithm Prefix-SID Redistribution for External Route Propagation	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature support is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
Flexible Algorithm Prefix-SID Redistribution for External Route Propagation	Release 7.5.2	You can now propagate flexible algorithm prefix-SIDs and their algorithm-specific metric between different IGP domains, such as OSPF to IS-IS RIP to OSPF. With this functionality enabling interdomain traffic engineering, you can export flexible algorithm labels from the OSPF domain to other domains and import the labels from other domains into OSPF. The **show ospf route flex-algo** command has been modified to include additional attributes to indicate the external routes.

Previously, prefix redistribution from IS-IS to another IS-IS instance or protocol was limited to SR algorithm 0 (regular SPF) prefix SIDs; SR algorithm 1 (Strict SPF) and SR algorithms 128-255 (Flexible Algorithm) prefix SIDs were not redistributed along with the prefix. The Segment Routing IS-IS Flexible Algorithm Prefix SID Redistribution feature allows redistribution of strict and flexible algorithms prefix SIDs from IS-IS to another IS-IS instance or protocols. This feature is enabled automatically when you configure redistribution of IS-IS Routes with strict or Flexible Algorithm SIDs.

Prefix redistribution from OSPF to another AS was limited to SR algorithm 0 (regular SPF) prefix SIDs; SR algorithm 1 (Strict SPF) and SR algorithms 128-255 (Flexible Algorithm) prefix SIDs were not redistributed along with the prefix. Starting from Cisco IOS XR Release 7.5.2, the Flexible Algorithm Prefix-SID Redistribution for External Route Propagation feature allows redistribution of strict and flexible algorithm prefixes SIDs from OSPF to another AS and also from another AS into OSPF.

Verification

This following show output displays the route-type as 'Extern' for the external routes.

```
Router#show ospf routes flex-algo 240 route-type external detail
Route Table of ospf-1 with router ID 192.168.0.2 (VRF default)

  Algorithm 240

  Route entry for 192.168.4.3/32, Metric 220, SID 536, Label 16536
   Priority : Medium

     Route type : Extern Type 1
Last updated : Apr 25 14:30:12.718
     Flags: Inuse

     Prefix Contrib Algo 240 SID 536
      From 192.168.0.4 Route-type 5
      Total Metric : 220 Base metric 20 FAPM 20
      Contrib Flags : Inuse, Reachable
      SID Flags : PHP off, Index, Global, Valid

      Path: 10.1.1.3, from 192.168.0.4, via GigabitEthernet0/2/0/2
       Out Label  : 16536
```

```
        Weight     : 0
        Area       : 0

       Path: 10.1.2.3, from 192.168.0.4, via GigabitEthernet0/2/0/3
        Out Label  : 16536
        Weight     : 0
        Area       : 0

       Path: 10.2.1.5, from 192.168.0.4, via GigabitEthernet0/2/0/4
        Out Label  : 16536
        Weight     : 0
        Area       : 0

   Route entry for 192.168.4.5/32, Metric 120, SID 556, Label 16556
    Priority : Medium

      Route type : Extern Type 1
      Last updated : Apr 25 14:30:12.724
      Flags: Inuse

      Prefix Contrib Algo 240 SID 556
       From 192.168.0.3 Route-type 5
       Total Metric : 120 Base metric 1 FAPM 20
       Contrib Flags : Inuse, Reachable
       SID Flags : PHP off, Index, Global, Valid

       Path: 10.1.1.3, from 192.168.0.3, via GigabitEthernet0/2/0/2
        Out Label  : 16556
        Weight     : 0
        Area       : 0

       Path: 10.1.2.3, from 192.168.0.3, via GigabitEthernet0/2/0/3
        Out Label  : 16556
        Weight     : 0
        Area       : 0
```

The following show output displays label information for flexible algorithm and its corresponding metric as added in RIB:

```
RP/0/RP0/CPU0:ios# show route 192.168.0.2/32 detail
Wed Apr  6 16:24:46.021 IST

Routing entry for 192.168.0.2/32
  Known via "ospf 1", distance 110, metric 2, labeled SR, type intra area
  Installed Apr  6 15:51:57.973 for 00:32:48
  Routing Descriptor Blocks
    10.10.10.2, from 192.168.0.2, via GigabitEthernet0/2/0/0, Protected
      Route metric is 2
      Label: 0x3 (3)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x1(Ref:1)
      Backup path id:65
      OSPF area: 1
    10.11.11.2, from 192.168.0.2, via GigabitEthernet0/2/0/1, Backup (Local-LFA)
      Route metric is 6
      Label: 0x3 (3)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:65              Path ref count:1
      NHID:0x2(Ref:1)
      OSPF area:
```

```
Route version is 0x12 (18)
Local Label: 0x3ee6 (16102)
Local Label Algo Set (ID, Label, Metric): (1, 16202, 0),(128, 17282, 2)
IP Precedence: Not Set
QoS Group ID: Not Set
Flow-tag: Not Set
Fwd-class: Not Set
Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 1, Download Version 38
No advertising protos.
```

Configuring Flexible Algorithm

> **Note** For information about the commands usage, see the Segment Routing Command Reference for Cisco 8000 Series Routers.

The following ISIS and OSPF configuration sub-mode is used to configure Flexible Algorithm:

flex-algo *algorithm number*

algorithm number —value from 128 to 255

Commands under Flexible Algorithm Configuration Mode

The following commands are used to configure Flexible Algorithm definition under the flex-algo sub-mode:

metric-type delay

> **Note** By default the regular IGP metric is used. If delay metric is enabled, the advertised delay on the link is used as a metric for Flexible Algorithm computation.

affinity {**include-any** | **include-all** | **exclude-any**} *name1, name2, …*

name—name of the affinity map

priority *priority value*

priority value—priority used during the Flexible Algorithm definition election.

The following command is used to enable advertisement of the Flexible Algorithm definition in IS-IS:

advertise-definition

Commands for Affinity Configuration

The following command is used for defining the affinity-map. Affinity-map associates the name with the particular bit positions in the Extended Admin Group bitmask.

affinity-map *name* **bit-position** *bit number*

- *name*—name of the affinity-map.

- *bit number*—bit position in the Extended Admin Group bitmask.

The following command is used to associate the affinity with an interface:

affinity flex-algo *name 1, name 2, …*

name—name of the affinity-map

Command for Prefix-SID Configuration

The following command is used to advertise prefix-SID for default and strict-SPF algorithm:

prefix-sid [**strict-spf** | **algorithm** *algorithm-number*] [**index** | **absolute**] *sid value*

- *algorithm-number*—Flexible Algorithm number
- *sid value*—SID value

IS-IS Enhancements: max-metric and data plane updates

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Enhancements: max-metric and data plane updates	Release 7.8.1	The new **anomaly** optional keyword is introduced to **affinity flex-algo** command. This keyword helps to advertise the flex-algo affinity when the performance measurement signals a link anomaly, such as an excessive delay on a link. You could use the anomaly option to exclude the link from flex-algo path computations. affinity flex-algo

With the IOS XR Release 7.8.1, the new optional keyword **anomaly** is introduced to the **interface** submode of **affinity flex-algo**. This keyword option helps to advertise flex-algo affinity on PM anomaly. The following command is used to associate the affinity with an interface:

router isis *instance* **interface** *type interface-path-id* **affinity flex-algo anomaly** *name 1, name 2, …*

router ospf *process* **area** *area* **interface** *type interface-path-id* **affinity flex-algo anomaly** *name 1, name 2, …*

name—name of the affinity-map

You can configure both normal and anomaly values. For the following example, the **blue** affinity is advertised. However, if a metric is received with the anomaly flag set, it will change to **red**:

```
Router# configure
Router(config)# router isis 1
Router(config-isis)#flex-algo 128
```

```
Router(config-isis-flex-algo)# interface GigabitEthernet0/0/0/2
Router(config-isis-flex-algo)# affinity flex-algo blue
Router(config-isis-flex-algo)# affinity flex-algo anomaly red
```

Configuring Flexible Algorithm with Exclude SRLG Constraint

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Algorithm to Exclude SRLGs for OSPF	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature support is now extended to the Cisco 8712-MOD-M routers.
Flexible Algorithm to Exclude SRLGs for OSPF	Release 7.5.2	You can now configure the flexible algorithm to exclude any link belonging to the Shared Risk Link Groups (SRLGs) from the path computation for OSPF. The ability to exclude the at-risk links ensures that the rest of the links in the network remain unaffected.

This feature allows the Flexible Algorithm definition to specify Shared Risk Link Groups (SRLGs) that the operator wants to exclude during the Flex-Algorithm path computation. A set of links that share a resource whose failure can affect all links in the set constitute a SRLG. An SRLG provides an indication of which links in the network might be at risk from the same failure.

This allows the setup of disjoint paths between two or more Flex Algos by leveraging deployed SRLG configurations. For example, multiple Flex Algos could be defined by excluding all SRLGs except one. Each FA will prune the links belonging to the excluded SRLGs from its topology on which it computes its paths.

This provides a new alternative to creating disjoint paths with FA, in addition to leveraging FA with link admin group (affinity) constraints.

The Flexible Algorithm definition (FAD) can advertise SRLGs that you want to exclude during the Flexible Algorithm path computation. The IS-IS Flexible Algorithm Exclude SRLG Sub-TLV (FAESRLG) is used to advertise the exclude rule that is used during the Flexible Algorithm path calculation, as specified in IETF draft https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/

The Flexible Algorithm path computation checks if an “exclude SRLG” rule is part of the FAD. If an “exclude SRLG” rule exists, it then checks if the link is part of an SRLG that is also part of the “exclude SRLG” rule. If the link is part of an excluded SRLG, the link is pruned from the path computation.

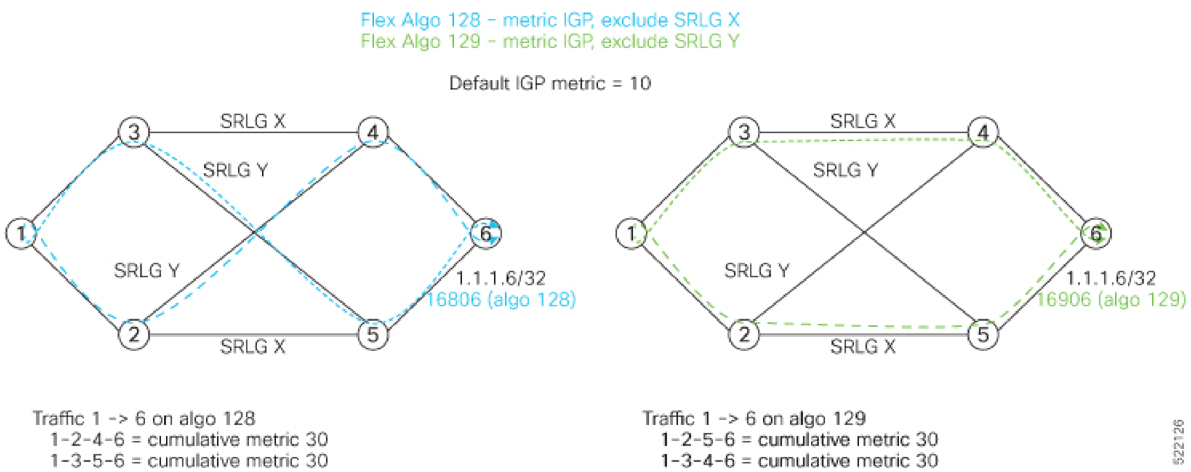
The figure below shows a topology configured with the following flex algos:

- Flex algo 128: metric IGP and exclude SRLG X constraint
- Flex algo 129: metric IGP and exclude SRLG Y constraint

The horizontal links between nodes 3 and 4 and between 2 and 5 are part of SRLG group X. The diagonal links between nodes 3 and 5 and between 2 and 4 are part of SRLG group Y. As a result, traffic from node 1 to node 6's FA 128 prefix SID (16806) avoids interfaces part of SRLG X. While traffic from node 1 to node 6's FA 129 prefix SID (16906) avoids interfaces part of SRLG Y.

Note See Constraints section in the *Configure SR-TE Policies* chapter for information about configuring SR policies with Flex-Algo constraints.

Figure 1: Flex Algo with Exclude SRLG Constraint



Configuration

Use the **router isis** *instance* **address-family ipv4 unicast advertise application flex-algo link-attributes srlg** command to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs.

Use the **router isis** *instance* **flex-algo** *algo* **srlg exclude-any** *srlg-name* . . . *srlg-name* command to configure the SRLG constraint which is advertised in the Flexible Algorithm definition (FAD) if the FAD advertisement is enabled under the flex-algo sub-mode. You can specify up to 32 SRLG names.

The SRLG configuration (value and port mapping) is performed under the global SRLG sub-mode. Refer to Configuring SRLG Node Protection for more information.

Example

The following example shows how to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs and to exclude SRLG groups from Flexible Algorithm path computation:

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/1/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/1/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
```

```
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# name groupX value 100
RP/0/RP0/CPU0:router(config-srlg)# name groupY value 200
RP/0/RP0/CPU0:router(config-srlg)# exit

RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-af)# advertise application flex-algo link-attributes srlg
RP/0/RP0/CPU0:router(config-isis-af)# exit
RP/0/RP0/CPU0:router(config-isis)# flex-algo 128
RP/0/RP0/CPU0:router(config-isis-flex-algo)# advertise-definition
RP/0/RP0/CPU0:router(config-isis-flex-algo)# srlg exclude-any groupX
RP/0/RP0/CPU0:router(config-isis-flex-algo)# exit
RP/0/RP0/CPU0:router(config-isis)# flex-algo 129
RP/0/RP0/CPU0:router(config-isis-flex-algo)# advertise-definition
RP/0/RP0/CPU0:router(config-isis-flex-algo)# srlg exclude-any groupY
RP/0/RP0/CPU0:router(config-isis-flex-algo)# commit
RP/0/RP0/CPU0:router(config-isis-flex-algo)# exit
RP/0/RP0/CPU0:router(config-isis)#
```

The following example shows how to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs and to exclude SRLG groups from Flexible Algorithm path computation for OSPF:

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/1/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/1/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# name groupX value 100
RP/0/RP0/CPU0:router(config-srlg)# name groupY value 200
RP/0/RP0/CPU0:router(config-srlg)# exit

RP/0/0/CPU0:r1(config)#router ospf 1
RP/0/0/CPU0:r1(config-ospf)#flex-algo 128
RP/0/0/CPU0:r1(config-ospf-flex-algo)#srlg exclude-any
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#groupX
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#groupY
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#commit
```

Verification

The following example shows how to verify the number of SRLGs excluded for OSPF:

```
RP/0/RP0/CPU0:router# show ospf topology summary
 Process ospf-1
 Instance default
  Router ID       : 192.168.0.1
  Number of Areas : 1
```

```
  Number of Algos : 1
  Max Path count  : 16
  Route count     : 10
  SR Global Block : 16000 - 23999

Area 0
    Number of Nodes : 6
     Algo 128
      FAD Advertising Router : 192.168.0.1
      FAD Area ID : 0
       Algo Type   : 0
       Metric Type : 0
          Number of Exlclude SRLGs  : (2)
          [1]:  100        [2]:  200
       FAPM supported : No
```

Flexible Algorithm with Exclude Minimum Bandwidth Constraint

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Flexible Algorithm with Exclude Minimum Bandwidth Constraint	Release 7.11.1	Traffic engineering in networks can be optimized by avoiding low-bandwidth links that may not be capable of handling high volumes of traffic. This feature allows you to use Flexible Algorithm to create topologies in your network that explicitly exclude high bandwidth traffic from utilizing links below a specified capacity. This constraint is achieved by introducing a new bandwidth-based metric type within the Flexible Algorithm framework. Links that do not satisfy the constraint are ignored when computing the associated Flexible Algorithm topology. This feature introduces these changes: **CLI**: • The **router isis** *instance* **flex-algo** *algo* command is modified with the new **minimum-bandwidth** *value* option. **YANG Data Model:** • This feature extends the native `Cisco-IOS-XR-clns-isis-cfg.yang` model (see GitHub, YANG Data Models Navigator)

This feature allows you to configure a minimum bandwidth value for computing a Flexible Algorithm path.

The IS-IS Flex-Algorithm Exclude Minimum Bandwidth sub-TLV (FAEMB) is a way to set a minimum bandwidth requirement for links in the Flex-Algorithm topology.

To determine if a link should be excluded based on this minimum bandwidth requirement, we compare the Minimum Bandwidth specified in the FAEMB sub-TLV with the Maximum Link Bandwidth advertised in the Area Supported by the Link Attribute (ASLA) sub-TLV.

If the Maximum Link Bandwidth is lower than the Minimum bandwidth specified, the link is excluded from the Flex-Algorithm topology. However, if the FAD includes the FAEMB sub-TLV but the Maximum Link Bandwidth is not advertised for the link, it should not be excluded based on the Minimum Bandwidth constraint.

Use the **router isis** *instance* **flex-algo** *algo* **minimum-bandwidth** *value* command to configure the minimum bandwidth value in kbps.

Example: Configuring IS-IS Flexible Algorithm with Minimum Bandwidth Constraint

```
router isis 1
 address-family ipv4 unicast
  segment-routing mpls
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator L1_A129
   !
  !
 !
 flex-algo 129
  advertise-definition
  minimum-bandwidth 10000000
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 100
   prefix-sid algorithm 129 index 300
  !
  address-family ipv6 unicast
 !
!
segment-routing
 srv6
  locators
   locator L1_A129
    micro-segment behavior unode psp-usd
    prefix cafe:0:2100::/48
    algorithm 129
   !
  !
 !
!
```

Flexible Algorithm with Exclude Maximum Delay Constraint

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Flexible Algorithm with Exclude Maximum Delay Constraint	Release 7.11.1	This feature enables you to configure topologies that exclude links that have delays over a specific threshold. This is especially critical for high-frequency trading applications, in satellite networks, or wherever there are fluctuations in link delays. This feature introduces these changes: **CLI**: • The **router isis** *instance* **flex-algo** *algo* command is modified with the new **maximum-delay** *value* option. **YANG Data Model:** • This feature extends the native Cisco-IOS-XR-clns-isis-cfg.yang model (see GitHub, YANG Data Models Navigator)

This feature allows you to configure a maximum delay value for computing a Flexible Algorithm path.

The Flexible Algorithm Exclude Minimum Delay (FAEMD) sub-TLV is used to specify the maximum delay requirement for links in a Flex-Algorithm topology. To ensure proper functioning, the FAEMD sub-TLV must appear only once in the FAD sub-TLV (Flexible Algorithm Definition). If it appears more than once, it should be ignored by the receiver. The maximum link delay advertised in the FAEMD sub-TLV is compared with the minimum unidirectional link delay advertised in the ASLA sub-TLV.

If the minimum unidirectional link delay is higher than the maximum link delay advertised in the FAEMD sub-TLV, the link must be excluded from the Flex-Algorithm topology.

However, if a link does not have the minimum unidirectional link delay advertised but the FAD contains the FAEMD sub-TLV, then based on the maximum delay constraint, that link should not be excluded from the topology.

Use the **router isis** *instance* **flex-algo** *algo* **maximum-delay** *delay* command to configure the maximum delay value in microseconds.

Example: Configuring IS-IS Flexible Algorithm with Maximum Delay Constraint

```
router isis 1
 address-family ipv4 unicast
  segment-routing mpls
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator L1_A128
    !
   !
 !
 flex-algo 128
  advertise-definition
  maximum-delay 300
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 100
   prefix-sid algorithm 128 index 200
  !
  address-family ipv6 unicast
 !
!
segment-routing
 srv6
  locators
   locator L1_A128
    micro-segment behavior unode psp-usd
    prefix cafe:0:1100::/48
    algorithm 128
   !
  !
 !
!
```

Maximum Paths Per IS-IS Flexible Algorithm Per Prefix

Table 7: Feature History Table

Feature Name	Release	Description
Maximum Paths Per IS-IS Flexible Algorithm Per Prefix	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100]) This feature support is now extended to the Cisco 8712-MOD-M routers.

Feature Name	Release	Description
Maximum Paths Per IS-IS Flexible Algorithm Per Prefix	Release 7.11.1	Previously, you could configure a maximum number of Equal-Cost Multi-path (ECMP) to be set for individual Flex Algorithms. This feature provides additional granularity to the IS-IS Maximum Paths Per-Algorithm feature by allowing you to specify a set of prefixes for Flexible Algorithm. Now you can achieve a balance between path diversity and computational and memory requirements by controlling the number of paths for each specific algorithm and destination prefix combination. This feature introduces these changes: **CLI** • **maximum-paths route-policy** *name* YANG Data Models: • This feature extends the native `Cisco-IOS-XR-clns-isis-cfg.yang` model See GitHub, Yang Data Models Navigator

Previously, you could set the maximum paths for a Flexible Algorithm per address-family.

With this feature, you can further refine the maximum paths configuration by associating it with specific prefixes for each Flexible Algorithm. The existing **maximum-paths** command is extended to include a **route-policy** qualifier to configure the maximum paths per algorithm per prefix-list.

When installing paths into the Routing Information Base (RIB) for Segment Routing with IPv6 (SRv6) or the Label Switched Database (LSD) for Segment Routing with MPLS (SR-MPLS), the system checks if a maximum paths value has been configured for the algorithm and the associated prefix. If such a configuration exists, it will be used instead of the existing address-family value to determine the number of paths to be installed.

Note Route policies that have the attribute **set maximum-paths** *number* are supported.

Note: For information on maximum paths per prefix for IS-IS algo 0 (SPF), refer to the "Implementing IS-IS" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Usage Guidelines and Limitations

- The **maximum-paths** *maximum-paths* and **maximum-paths route-policy** *name* configurations are mutually exclusive. You can configure either an unqualified number or a route-policy for any given IS-IS instance.
- The maximum paths per-algorithm per-prefix configuration takes precedence over maximum paths per-algorithm configuration. Likewise, the maximum paths per-algorithm configuration takes precedence over maximum paths per-address-family configuration. This hierarchy ensures that the most specific configuration is prioritized when determining the maximum paths for a given algorithm and prefix combination.

Example

The following example shows how to configure the maximum paths for Flex Algo 128:

- **Define a Prefix Set:**

```
prefix-set isis-ipv4-L1
  10.1.0.101/32
end-set
```

- **Create a Route Policy:**

```
route-policy isis-mp-if-L1
  if destination in isis-ipv4-L1 then
    set maximum-paths 2
  endif
end-policy
```

- **Apply Route Policy to Configure Maximum Paths Per-Algo Per-Prefix:**

```
router isis 10
 flex-algo 128
  address-family ipv4 unicast
   maximum-paths route-policy isis-mp-if-L1
```

Verification

```
Router# show isis route flex-algo 128

IS-IS 10 IPv4 Unicast routes Flex-Algo 128

Codes: L1 - level 1, L2 - level 2, ia - interarea (leaked into level 1)
       df - level 1 default (closest attached router), su - summary null
       C - connected, S - static, R - RIP, B - BGP, O - OSPF
       E - EIGRP, A - access/subscriber, M - mobile, a - application
       i - IS-IS (redistributed from another instance)

Maximum parallel path count: as defined in isis-mp-if-L1

L1 10.1.0.101/32 [121/115]
     via 15.15.15.2, GigabitEthernet0/0/0/5, hare, SRGB Base: 16000, Weight: 0
```

```
via 16.16.16.2, GigabitEthernet0/0/0/6, hare, SRGB Base: 16000, Weight: 0
```

Flexible algorithm with bandwidth optimization

Table 8: Feature History Table

Feature name	Release information	Feature description
Flexible algorithm with bandwidth optimization	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The enhanced IGP flexible algorithm path computation optimizes routing paths by automatically adjusting to changes in link bandwidth, which is especially beneficial for handling parallel L3 links and dynamic bandwidth variations, such as in L2 link bundles. The algorithm ensures optimal capacity paths by considering the cumulative bandwidth of parallel links and prefers paths with the highest available bandwidth, improving performance for high-bandwidth traffic flows. In addition to traditional metrics like link delay or monetary cost, the algorithm can also optimize paths based on the maximum available bandwidth of links, which can be locally configured or computed from advertised link bandwidth. The feature introduces these changes: **CLI:** • **bandwidth-metric flex-algo** • **metric-type bandwidth** • **reference-bandwidth** • **group-mode** **YANG Data Models:** • New Xpaths are introduced for `Cisco-IOS-XR-um-router-isis-cfg.yang` (see GitHub, YANG Data Models Navigator)

IGPs have traditionally computed paths by minimizing the sum of administratively defined link metrics, often inversely proportional to link bandwidth. In other cases, metrics are based on attributes like link delay or monetary cost. The enhanced Flexible Algorithm (FA) path computation provides an additional flex-algo topology that is optimized purely on bandwidth, ensuring more efficient use of network resources by selecting paths based on physical bandwidth.

Challenges with traditional metrics on parallel links

The handling of parallel L3 links with administratively assigned metrics, even if these metrics are based on link bandwidth, does not always result in the best capacity path. This is because traditional metrics do not account for the cumulative bandwidth available across multiple parallel links.

FA bandwidth optimization for parallel links

FA bandwidth optimization provides a more accurate path selection by considering the most available bandwidth across these parallel links, ensuring that the chosen path can handle the required traffic load more effectively.

If the bandwidth of the interface changes, such as in the case of an L2 link bundle where a bundle member link goes up or down, the advertised link bandwidth will change accordingly. With bandwidth optimization, FA paths can be automatically adjusted. This dynamic adjustment ensures that the network can adapt to changes in real-time, maintaining optimal performance and avoiding potential bottlenecks. For instance, if a link within a bundle fails, the Flexible Algorithm can recalculate the paths to route traffic through the remaining available bandwidth, thereby maintaining service continuity and efficiency.

FA forwarding is available in multiple transport mechanisms:

- SR-MPLS: Utilizes algorithm-specific Segment Identifiers (SIDs)
- SRv6: Uses algorithm-specific locators
- IP Algorithm: Employs IP algorithm prefixes

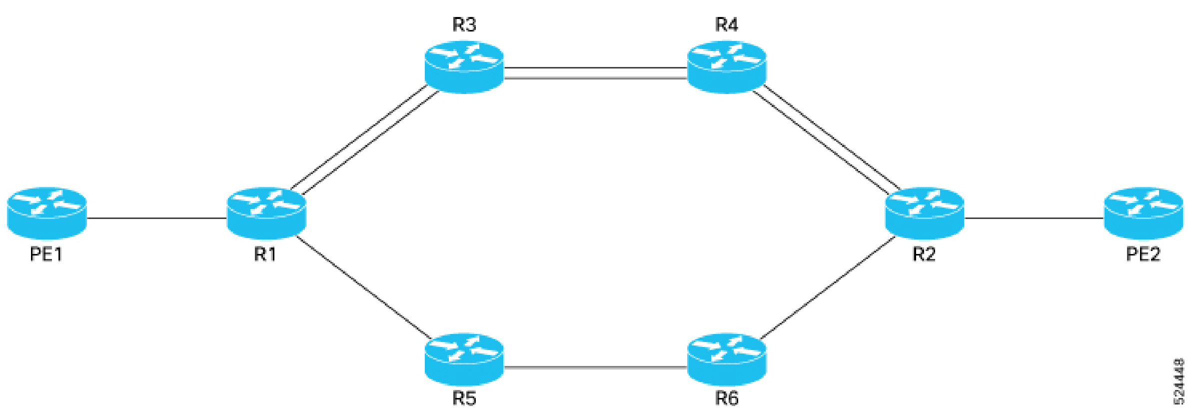
All these methods support bandwidth-based computation.

Cumulative bandwidth over parallel links

Consider this topology, where PE1 is connected to PE2 via two different paths,

- Path 1: PE1-R1-R3-R4-R2-PE2
- Path 2: PE1-R1-R5-R6-R2-PE2

Figure 2: Example topology



In traditional path computation, if all links are of same capacity and all have the same metric 10, traffic from PE1 to PE2 is load balanced equally between Path 1 and Path 2, without considering the parallel links available in Path 1. Bandwidth based optimization considers the presence of these parallel links and computes the path using the cumulative bandwidth. This is useful for traffic flows that involve a large volume of data being transferred over a long duration.

Cumulative bandwidth metric calculation can be enabled using the **group-mode** command.

Benefits of flexible algorithm with bandwidth optimization

- **Optimized path selection**: Ensures paths are selected based on physical bandwidth, leading to more efficient use of network resources.

- **Dynamic adjustment**: Automatically adjusts paths in real-time based on changes in link bandwidth, such as when a link in an L2 bundle goes up or down.
- **Improved handling of parallel links**: Considers cumulative bandwidth across parallel links, preferring paths with the best available bandwidth.

Usage guidelines for flexible algorithm with bandwidth optimization

Bandwidth metric

- Bandwidth metric can be manually computed and locally configured on a link using the **bandwidth-metric flex-algo** command. This is advertised similar to other metrics in IGP. Manual Bandwidth Metric is advertised in the ISIS Generic-Metric Sub-TLV, as type 3 metric.
- If no bandwidth metric is advertised, it is calculated for both local and remote links using the maximum physical link bandwidth from the interface attributes. This metric is derived from the maximum link bandwidth, using the Reference Bandwidth and Granularity Bandwidth provided by the Flexible Algorithm definition (FAD).

  If Granularity Bandwidth is less than or equal to Maximum Link Bandwidth, this is the bandwidth metric calculation:

  ```
  Bandwidth Metric = Reference Bandwidth / (Maximum Link Bandwidth - (MOD(Maximum Link
  Bandwidth , Granularity Bandwidth))
  ```

  If Granularity Bandwidth is more than Maximum Link Bandwidth, this is the bandwidth metric calculation:

  ```
  Bandwidth Metric = Reference Bandwidth / Maximum Link Bandwidth
  ```

- If there is only one link between nodes and bandwidth metric type is advertised for the link, the Flex-Algorithm calculation uses the advertised bandwidth metric.
- In interface group mode,
  - if all parallel links have a bandwidth metric advertised, the individual link bandwidth metric is used.
  - if none or only some parallel links have a bandwidth metric advertised, the advertised metrics are ignored, and the link metric is automatically calculated.
- If the calculated metric is zero, a metric of 1 is used.

Configure flexible algorithm with bandwidth optimization

This section includes configuration for bandwidth metrics for IS-IS Flexible Algorithms.

Procedure

Step 1 Define the bandwidth metric value for an interface using the **bandwidth-metric flex-algo** command in the IS-IS interface address-family submode. In the following example, the metric value is configured as 100.

Example:

```
Router(config)#router isis 1
Router(config-isis)#interface GigabitEthernet 0/2/0/7
Router(config-isis-if)#address-family ipv4 unicast
Router(config-isis-if-af)#bandwidth-metric flex-algo 100
```

Step 2 Define metric-type as bandwith using the **metric-type** command in the IS-IS flex-algo submode.

Example:

```
Router(config)#router isis 1
Router(config-isis)#flex-algo 131
Router(config-isis-flex-algo)#metric-type bandwidth
```

Step 3 Configure the parameters for automatic bandwidth metric calculation using the **reference-bandwidth** command in the IS-IS flex-algo auto-cost submode.

Example:

```
Router(config)#router isis 1
Router(config-isis)#flex-algo 131
Router(config-isis-flex-algo)#auto-cost reference
Router(config-isis-flex-algo-af-auto-cost)#reference-bandwidth 10000000 granularity 2000
```

Step 4 Configure interface group mode using the **group-mode** command in the IS-IS flex-algo auto-cost submode.

Example:

```
Router(config)#router isis 1
Router(config-isis)#flex-algo 131
Router(config-isis-flex-algo)#auto-cost reference
Router(config-isis-flex-algo-af-auto-cost)#group-mode
Router(config-isis-flex-algo-af-auto-cost)#commit
```

Step 5 Verify the running configuration using the **show running-config** command.

Example:

```
router isis 1
 flex-algo 131
  metric-type bandwidth
  auto-cost reference
   group-mode
   reference-bandwidth 10000000 granularity 2000
  !
 !
 interface GigabitEthernet0/2/0/7
  address-family ipv4 unicast
   bandwidth-metric flex-algo 100
  !
 !
!
```

Step 6 Verify the Flexible Algorithm Definition configuration using the **show isis flex-algo** command.

Example:

```
Router#show isis flex-algo 131
.......
Definition Source: plzen.00
Definition Equal to Local: No
Definition Metric Type: Bandwidth
Auto-cost Reference Bandwidth: 99999998 (kbits/sec), (12499999744 bytes/sec)
Auto-cost Granularity: 2000 (kbits/sec), (250000 bytes/sec)
Auto-cost Group Mode: No
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
.......
```

Step 7 Verify the bandwidth metric configured for an interface using the **show isis interface** command.

Example:

```
Router#show isis interface GigabitEthernet 0/2/0/7

GigabitEthernet0/2/0/7 Enabled
Adjacency Formation: Enabled
Prefix Advertisement: Enabled
Bandwidth: 1000000
 ...
 ...
IPv4 Unicast Topology: Enabled
Adjacency Formation: Running
Prefix Advertisement: Running
Policy (L1/L2): -/-
Metric (L1/L2): 10/10
Metric fallback:
...
BW Metric (L1/L2): 100/100
```

Example: Configuring IS-IS Flexible Algorithm

```
router isis 1
 affinity-map red bit-position 65
 affinity-map blue bit-position 8
 affinity-map green bit-position 201

 flex-algo 128
  advertise-definition
  affinity exclude-any red
  affinity include-any blue
 !
 flex-algo 129
  affinity exclude-any green
 !
!
address family ipv4 unicast
 segment-routing mpls
!
interface Loopback0
 address-family ipv4 unicast
  prefix-sid algorithm 128 index 100
  prefix-sid algorithm 129 index 101
!
!
interface GigabitEthernet0/0/0/0
 affinity flex-algo red
!
interface GigabitEthernet0/0/0/1
 affinity flex-algo blue red
!
interface GigabitEthernet0/0/0/2
 affinity flex-algo blue
!
```

Example: Configuring OSPF Flexible Algorithm

```
router ospf 1
 flex-algo 130
  priority 200
  affinity exclude-any
   red
   blue
  !
  metric-type delay
 !
 flex-algo 140
  affinity include-all
   green
  !
  affinity include-any
   red
  !
 !

  interface Loopback0
   prefix-sid index 10
   prefix-sid strict-spf index 40
   prefix-sid algorithm 128 absolute 16128
   prefix-sid algorithm 129 index 129
   prefix-sid algorithm 200 index 20
   prefix-sid algorithm 210 index 30
  !
 !

  interface GigabitEthernet0/0/0/0
   flex-algo affinity
    color red
    color blue
   !
  !

 affinity-map
  color red bit-position 10
  color blue bit-position 11
 !
```

User-Defined Generic Metric Support for IS-IS Flex Algo

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
User-Defined Generic Metric Support for IS-IS Flex Algo	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) *User-Defined Generic Metric Support for IS-IS Flex Algo is now supported on Cisco 8712-MOD-M routers.

Feature Name	Release Information	Feature Description
User-Defined Generic Metric Support for IS-IS Flex Algo	Release 24.2.11	This feature adds support for user-defined generic metric as a metric type for IS-IS Flexible Algorithm. You can now have more control over traffic flows using user-defined generic metrics. You can define a family of user-defined generic metrics that can advertise different types of administrative metrics such as jitter, reliability, and fiscal cost depending on the traffic class for Flexible Algorithms. You can selectively define and assign semantics of these metrics as per the network requirement. The feature introduces the following changes: **CLI**: • The feature introduces the **generic-metric flex-algo** and **metric-type generic** commands. **YANG Data Models:** • `Cisco-IOS-XR-um-router-isis-cfg.yang`

Control Traffic Flow with User-Defined Generic Metrics

With the addition of different traffic types, the need for alternate types of metrics has evolved. Flexible Algorithm already supports IGP, TE, and delay metrics. However a network operator might want to minimize their operational costs and might want a metric that reflects the actual fiscal costs of using a link. Other traffic may require low jitter, leading to an entirely different set of metrics. With Flexible Algorithm, all these different metrics could be used concurrently on the same network. These improvements are possible as you can now define a family of user-defined generic metrics that can advertise various types of administratively assigned metrics. These metrics are not predefined, which provides network administrators with the flexibility to assign their own meanings and semantics to the metrics. This means you can create metrics tailored to specific operational goals or traffic requirements.

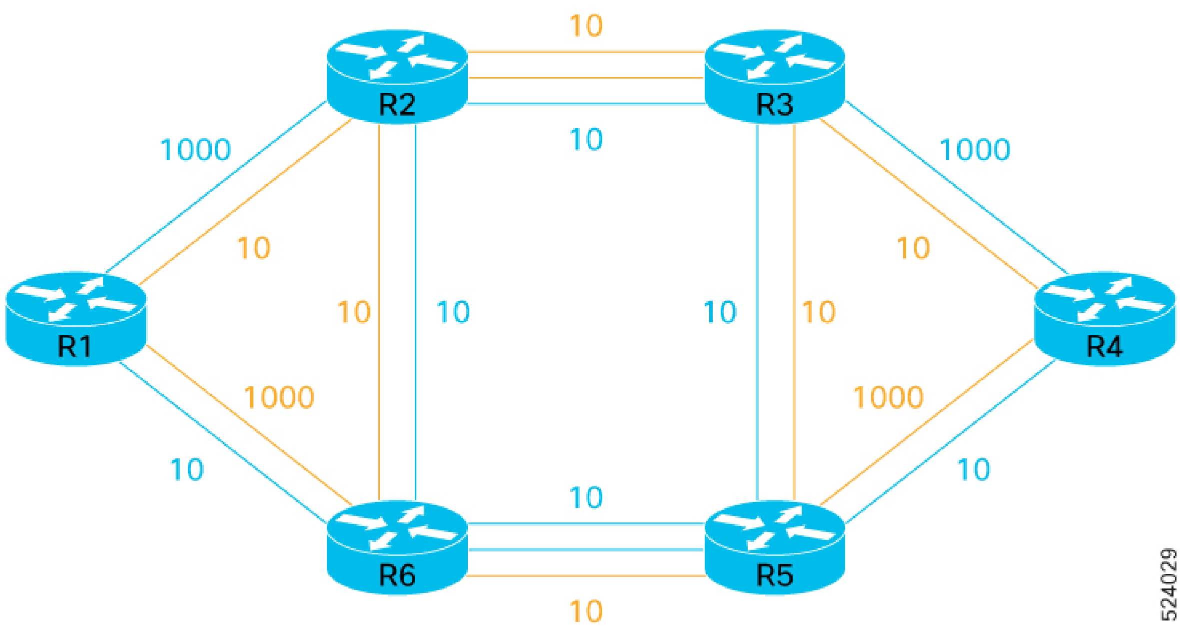
Apply User-Defined Generic Metrics in Flexible Algorithm Definition

You can apply the user-defined generic metrics within the Flexible Algorithm Definition (FAD) similar to how you would use TE or delay metrics. For example, you can assign specific user-defined generic metrics to individual Flex Algos, allowing for a customized path selection criteria. This capability enables different paths to use different metrics, ensuring that each path is optimized for the specific type of traffic it handles.

Split traffic with User-Defined Generic Metrics

Consider a scenario where for some networks, traffic engineering might require splitting east and west traffic for different Flexible Algorithms across multiple hardware data planes. In the following figure, the Flex Algo 128 (Orange) uses the upper primary path, while the Flex Algo 129 (Blue) uses the lower primary path. You can achieve this by defining specific user-defined generic metrics for the Flex Algo to determine the best path for each Flex Algo. The numbers 10 and 100 represent the user-defined generic metrics configured for the two different Flex Algos.

Figure 3: Splitting traffic using user-defined generic metrics for different Flex Algos



Advantages of splitting traffic using user-defined generic metrics

This approach offers several advantages. You can use Equal-Cost Multi-Path (ECMP) routing within the backbone, allowing multiple links of the same color between core nodes. It also ensures efficient backup routes and prevents U-loops with Flex Algo specific Loop-Free Alternates (LFA) and Topology-Independent Loop-Free Alternates (TI-LFA).

In the above example, the vertical core links are used as backup paths for all Flex Algos, and you can activate them only if a core link or node fails. These links are part of all Flex Algos, and you can use user-defined generic metrics to configure and advertise such metrics for each individual Flex Algo to ensure smooth operation.

Benefits of User-Defined Generic Metrics

The key benefits of user-defined generic metrics are:

- You can ensure precise control over routing decisions based on the assigned metrics, as the links that do not advertise the user-defined generic metric are excluded from the Flex Algo topology.
- You can use the metrics to reduce operational expenses by choosing cost-effective paths.
- You can customize metrics to fit unique operational goals and traffic requirements, providing tailored solutions for different scenarios.

- You can simultaneously apply multiple metrics across the same network, enhancing overall network performance and reliability.

Usage Guidelines and Limitations for User-Defined Generic Metrics

The user-defined generic metric is disabled by default. In other words, the metric is not advertised unless it is configured.

Configure User-Defined Generic Metrics

This section includes configuration for user-defined generic metrics for IS-IS Flex Algos.

Procedure

Step 1 Define the metric type and value for an interface using the **generic-metric flex-algo** command in the IS-IS interface address-family submode. In the following example, the user-defined generic metric type is 177 and the metric value as 100.

Example:

```
Router(config)#router isis 1
Router(config-isis)#interface GigabitEthernet 0/2/0/7
Router(config-isis-if)#address-family ipv4 unicast
Router(config-isis-if-af)#generic-metric flex-algo type 128 100
Router(config-isis-if-af)#generic-metric flex-algo type 177 100
Router(config-isis-if-af)#generic-metric flex-algo type 188 1000
```

Step 2 Associate or advertise the configured metric type to a Flexible Algorithm Definition using the **metric-type generic** command. In the following example, the metric type advertised is 177.

```
Router(config)#router isis 1
Router(config-isis)#flex-algo 128
Router(config-isis-flex-algo)#priority 254
Router(config-isis-flex-algo)#metric-type generic 177
Router(config-isis-flex-algo)#advertise-definition
```

Step 3 Verify the running configuration using the **show running-config** command.

Example:

```
router isis 1
 flex-algo 128
  priority 254
  metric-type generic 177
  advertise-definition
 !
 interface GigabitEthernet 0/2/0/7
  address-family ipv4 unicast
   generic-metric flex-algo type 128 100
   generic-metric flex-algo type 177 100
   generic-metric flex-algo type 188 1000
  !
 !
!
```

Step 4 Verify the Flexible Algorithm Definition configuration using the **show isis flex-algo** command.

Example:

```
Router#show isis flex-algo 128
Thu Dec 7 03:10:56.452 PST
IS-IS 1 Flex-Algo Database
Flex-Algo 128:
Level-2:
Definition Priority: 254
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: User-defined: 177.
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Minimum Link Bandwidth: 0 kbits/s
Maximum Link Delay: 0 us
Disabled: No
Topologies supported:
IPv4 Unicast
Local Priority: 254
FRR Disabled: No
Microloop Avoidance Disabled: No
UCMP Disabled: No
Data Plane Segment Routing: Yes
Data Plane IP: No
```

Step 5 Optionally, you can also verify the application-specific user-defined generic metric configured for an interface by using the **show isis interface** command.

Example:

```
Router#show isis interface GigabitEthernet 0/2/0/7
Thu Dec 7 03:13:34.140 PST
GigabitEthernet0/2/0/7 Enabled
Adjacency Formation: Enabled
Prefix Advertisement: Enabled
Bandwidth: 1000000
 ...
 ...
IPv4 Unicast Topology: Enabled
Adjacency Formation: Running
Prefix Advertisement: Running
Policy (L1/L2): -/-
Metric (L1/L2): 10/10
Metric fallback:
Bandwidth (L1/L2): Inactive/Inactive
Anomaly (L1/L2): Inactive/Inactive
Weight (L1/L2): 0/0
    L1 Flex-algo Generic-metrics:
       Type: 128           100
       Type: 177           100
       Type: 188           1000
    L2 Flex-algo Generic-metrics:
       Type: 128           100
       Type: 177           100
       Type: 188           1000
```