

Enabling Segment Routing Flexible Algorithm

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, on page 1
- Building Blocks of Segment Routing Flexible Algorithm, on page 1
- Configuring Flexible Algorithm, on page 10
- Example: Configuring IS-IS Flexible Algorithm, on page 27
- Example: Configuring OSPF Flexible Algorithm, on page 27
- User-Defined Generic Metric Support for IS-IS Flex Algo, on page 28
- Example: Traffic Steering to Flexible Algorithm Paths, on page 33
- Delay Normalization, on page 36

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 constraints may be used to compute a path over a network. Some networks are deployed with multiple planes. A simple form of constraint may be to use a particular plane. A more sophisticated form of constraint 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 a Flexible Algorithm.

Flexible Algorithm Membership

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
- Exclude SRLG constraint
- Minimum bandwidth constraint
- Maximum delay constraint

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 Link Attribute Advertisement

Various link attributes may be used during the Flexible Algorithm path calculation. For example, include or exclude rules based on link affinities can be part of the Flexible Algorithm definition, as defined in RFC9350 (IGP Flexible Algorithm).

Link attribute advertisements used during Flexible Algorithm calculation must use the Application-Specific Link Attribute (ASLA) advertisements, as defined in RFC8919 (IS-IS) and RFC8920 (OSPF). In the case of IS-IS, if the L-Flag is set in the ASLA advertisement, then legacy advertisements (IS-IS Extended Reachability TLV) are used instead.

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

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
OSPF: Microloop Avoidance for Flexible Algorithm	Release 7.4.1	This feature extends the current OSPF Flexible Algorithm functionality to support Microloop Avoidance.

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
OSPF: Microloop Avoidance for Flexible Algorithm	Release 7.3.2	This feature extends the current OSPF Flexible Algorithm functionality to support Microloop Avoidance.
OSPF: TI-LFA for Flexible Algorithm	Release 7.3.1	This feature extends the current OSPF Flexible Algorithm functionality to support TI-LFA.

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 Flexible Algorithm is used.

The router uses the following rules to prune links from the topology during the Flexible Algorithm computation:

- All nodes that don't advertise support for Flexible Algorithm are pruned from the topology.
- Affinities:
  - Check if any exclude affinity rule is part of the Flexible Algorithm Definition. If such exclude rule exists, check if any color that is part of the exclude rule is also set on the link. If such a color is set, the link must be pruned from the computation.
  - Check if any include-any affinity rule is part of the Flexible Algorithm Definition. If such include-any rule exists, check if any color that is part of the include-any rule is also set on the link. If no such color is set, the link must be pruned from the computation.
  - Check if any include-all affinity rule is part of the Flexible Algorithm Definition. If such include-all rule exists, check if all colors that are part of the include-all rule are also set on the link. If all such colors are not set on the link, the link must be pruned from the computation

Note See Flexible Algorithm Affinity Constraint.

- If the Flexible Algorithm definition includes an "exclude SRLG" rule, then all links that are part of such SRLG are pruned from the topology.

Note See Flexible Algorithm with Exclude SRLG Constraint, on page 16.

- If the minimum bandwidth constraint is configured, any link that does not meet the minimum link bandwidth threshold is pruned from the topology.
- If the maximum delay constraint is configured, any link that exceeds the maximum unidirectional link delay is pruned from the topology.
- Router uses the metric that is part of the Flexible Algorithm definition. If the metric isn't advertised for the particular link, the link is pruned from the topology.

Configuring Microloop Avoidance for Flexible Algorithm

By default, Microloop Avoidance per Flexible Algorithm instance follows Microloop Avoidance configuration for algo-0. For information about configuring Microloop Avoidance, see Configure Segment Routing Microloop Avoidance.

You can disable Microloop Avoidance for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo microloop avoidance disable

router ospf process flex-algo algo microloop avoidance disable
```

Configuring LFA / TI-LFA for Flexible Algorithm

By default, LFA/TI-LFA per Flexible Algorithm instance follows LFA/TI-LFA configuration for algo-0. For information about configuring TI-LFA, see Configure Topology-Independent Loop-Free Alternate (TI-LFA).

You can disable TI-LFA for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo fast-reroute disable

router ospf process flex-algo algo fast-reroute disable
```

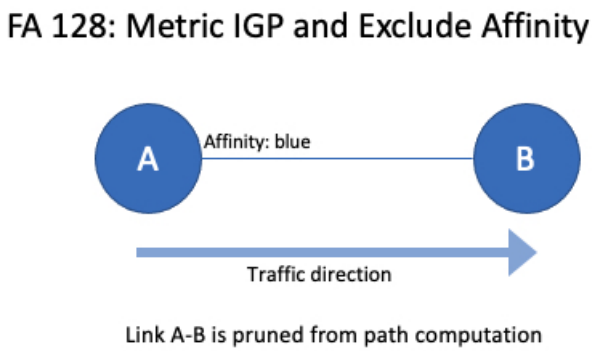
Flexible Algorithm Affinity Constraint

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS: Flexible Algorithm Reverse Affinity	Release 7.9.1	This feature enhances the IS-IS Flexible Algorithm link admin group (affinity) constraint to include link colors on links in the reverse direction toward the calculating router. The ability to apply affinity constraints in the reverse direction provides additional control for IS-IS Flexible Algorithm path computation. This feature intoduces the **reverse** keyword to the **router isis** *instance* **flex-algo** *algo* **affinity** command.

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 a Flexible Algorithm path and link colors in the forwarding direction. Flexible Algorithm computes a path that includes or excludes links that have specific colors, or combinations of colors.

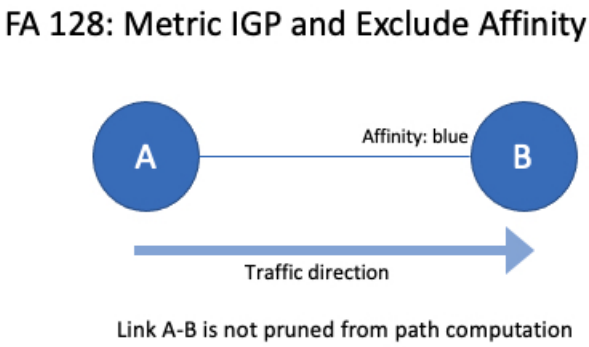
- Affinity “blue” is assigned to interface on node A; exclude affinity "blue": Link A-B is pruned from path calculation



In Cisco IOS XR release 7.9.1, for IS-IS Flexible Algorithm, you can also specify a reverse affinity between a Flexible Algorithm path and link colors (in the direction toward the computing router). Flexible Algorithm computes a path that includes or excludes links in the reverse direction that have specific colors, or combinations of colors.

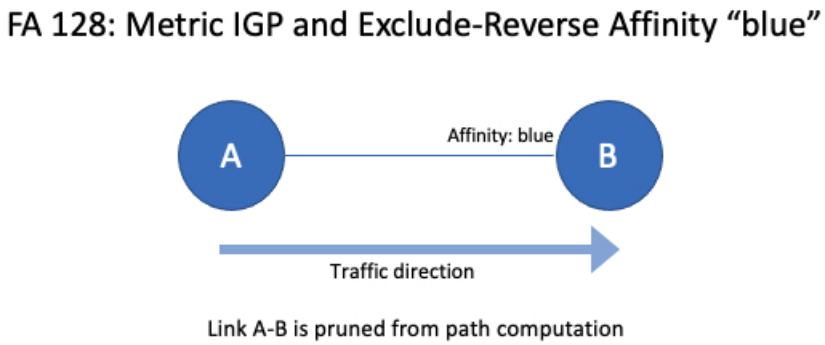
For example, on a point-to-point link between endpoints A and B and for the traffic flowing in the direction from A to B, the input errors can only be detected at node B. You may measure the rate of such input errors and set certain 'color' on a link locally on node B when the input error rate crosses a certain threshold.

- Affinity “blue” is assigned to interface on node B; exclude affinity "blue": Link A-B is not pruned from path calculation



With IS-IS Flexible Algorithm Reverse Affinity, when Flex-Algorithm calculation processes link A to B, it may look at the 'colors' of the link in the reverse direction (link B to A). This enables you to exclude this link from the Flex-Algorithm topology.

- Affinity “blue” is assigned to interface on node B; exclude reverse-affinity "blue": Link A-B is pruned from path calculation



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 IPG paths computed based on the default algorithm and regular IGP metrics.

Flexible Algorithm Prefix-SID Redistribution

Table 4: Feature History Table

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.

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.

Configuration Example

The following example shows how to configure redistribute and flexible algorithm to enable external routes.

```
RP/0/RP0/CPU0:ios(config)#router ospf 1
RP/0/RP0/CPU0:ios(config-ospf)#segment-routing mpls
RP/0/RP0/CPU0:ios(config-ospf)#segment-routing forwarding mpls
RP/0/RP0/CPU0:ios(config-ospf)#redistribute isis 2 route-policy loopback-type
RP/0/RP0/CPU0:ios(config-ospf)#flex-algo 240
RP/0/RP0/CPU0:ios(config-ospf-flex-algo)#metric-type delay
RP/0/RP0/CPU0:ios(config-ospf-flex-algo)#prefix-metric
RP/0/RP0/CPU0:ios(config-ospf-flex-algo)#advertise-definition
```

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.
```

Flexible Algorithm Prefix Metric

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Prefix Metric support for OSPF Flexible Algorithm	Release 7.5.1	This feature extends the current OSPF Flexible Algorithm functionality to introduce a Flexible Algorithm-specific prefix-metric in the OSPF prefix advertisement. The prefix-metric provides a way to compute the best end-to-end Flexible Algorithm optimized paths across multiple areas or domains.

A limitation of the existing Flexible Algorithm functionality in IS-IS and OSPF is the inability to compute the best path to a prefix in a remote area or remote IGP domain. Prefixes are advertised between IS-IS areas, OSPF processes, or between protocol domains, but the existing prefix metric does not reflect any of the constraints used for Flexible Algorithm path. Although the best Flexible Algorithm path can be computed to the inter-area or redistributed prefix inside the area, the path may not represent the overall best path through multiple areas or IGP domains.

The Flexible Algorithm Prefix Metric feature introduces a Flexible Algorithm-specific prefix-metric in the IS-IS and OSPF prefix advertisement. The prefix-metric provides a way to compute the best end-to-end Flexible Algorithm optimized paths across multiple areas or domains.

Note The Flexible Algorithm definition must be consistent between domains or areas. Refer to section 8 and section 9 in IETF draft https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/.

Configuring Flexible Algorithm

Table 6: 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

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
TE Metric Support for IS-IS Flex Algo	Release 7.4.1	Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type (path optimization objective) and constraint. This feature adds support for TE metric as a metric type for IS-IS Flexible Algorithm. This allows the TE metric, along with IGP and delay metrics, to be used when running shortest path computations.

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

```
router isis instance flex-algo algo
```

```
router ospf process flex-algo algo
```

algo—value from 128 to 255

Configuring Flexible Algorithm Definitions

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

- `router isis instance flex-algo algo metric-type {delay | te}`

  `router ospf process flex-algo algo metric-type {delay | te-metric}`

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

Note See Flexible Algorithm Link Attribute Advertisement Behavior, on page 12 for TE metric behaviors.

- **router isis** *instance* **flex-algo** *algo* **affinity** [**reverse**] { **include-any** | **include-all** | **exclude-any**} *name1, name2, …*

  router ospf *process* **flex-algo** *algo* **affinity** { **include-any** | **include-all** | **exclude-any**} *name1, name2, …*

  name—name of the affinity map

Note See Flexible Algorithm Affinity Constraint, on page 5 for information about affinity constraint behaviors.

- **router isis** *instance* **flex-algo** *algo* **priority** *priority value*

  router ospf *process* **flex-algo** *algo* **priority** *priority value*

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

The following command is used to to include the Flexible Algorithm prefix metric in the advertised Flexible Algorithm definition in IS-IS and OSPF :

```
router isis instance flex-algo algo prefix-metric

router ospf process flex-algo algo prefix-metric
```

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

```
router isis instance flex-algo algo advertise-definition
```

Configuring Affinity

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.

```
router isis instance flex-algo algo affinity-map name bit-position bit number

router ospf process flex-algo algo affinity-map name bit-position bit number
```

- *name*—name of the affinity-map.
- *bit number*—bit position in the Extended Admin Group bitmask.

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 Prefix-SID Advertisement

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

```
router isis instance interface type interface-path-id address-family {ipv4 | ipv6} [unicast]
 prefix-sid [strict-spf | algorithm algorithm-number]  [index | absolute] sid value

router ospf process area area interface Loopback interface-instance prefix-sid [strict-spf
 | algorithm algorithm-number]  [index | absolute] sid value
```

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

Flexible Algorithm Link Attribute Advertisement Behavior

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Advertisement of Link Attributes for IS-IS Flexible Algorithm	Release 7.4.1	Link attribute advertisements used during Flexible Algorithm path calculation must use the Application-Specific Link Attribute (ASLA) advertisements, as defined in IETF draft draft-ietf-lsr-flex-algo. This feature introduces support for ASLA advertisements during IS-IS Flexible Algorithm path calculation.

The following tables explain the behaviors for advertising (transmitting) and processing (receiving) Flexible Algorithm link attributes.

Table 9: OSPF

Link Attribute	Transmit	Receive
Link Delay Metric	IOS XR OSPF Flex Algo implementation advertises the link delay metric value using the OSPF ASLA sub-TLV with the F-bit set.	IOS XR OSPF only uses the link delay metric advertised in the ASLA sub-TLV for Flex Algo. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks.
Link TE Metric	IOS XR OSPF Flex Algo implementation advertises the link TE metric value using the OSPF ASLA sub-TLV with the F-bit set. The link TE metric values advertised are configured under SR-TE.	IOS XR OSPF only uses the TE metric advertised in the ASLA sub-TLV for Flex Algo. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks.
Link Admin Group/Extended Admin Group	IOS XR OSPF Flex Algo implementation advertises the link admin group value using both link admin group (AG) and link extended admin group (EAG) encoding using the OSPF ASLA sub-TLV with the F-bit set. The link admin group values advertised can be configured directly under the IGP and are therefore FA-specific. Otherwise, they will be derived from the link admin group values configured under SR-TE.	IOS XR OSPF only uses the AG/EAG (either one or both) advertised in the ASLA sub-TLV for Flex Algo. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks.

Table 10: IS-IS

Link Attribute	Transmit	Receive
Link Delay Metric	IOS XR IS-IS Flex Algo implementation advertises the link delay metric value using **both** the IS-IS Extended Reachability TLV and the IS-IS ASLA.	By default, IOS XR IS-IS Flex Algo implementation prefers the link delay metric value received in the IS-IS ASLA. Otherwise, it will use link delay metric value received in the IS-IS Extended Reachability TLV. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks. If the incoming ASLA includes the L-Flag, implementation derives the link delay metric value from the IS-IS Extended Reachability TLV. You can configure the IOS XR IS-IS Flex Algo implementation to strictly use the link delay metric value received in the IS-IS ASLA. See Strict IS-IS ASLA Link Attribute, on page 15.
Link TE Metric	IOS XR IS-IS Flex Algo implementation advertises the link TE metric value using the IS-IS ASLA. The link TE metric values advertised can be configured directly under the IGP and are therefore FA-specific. Otherwise, they will be derived from the link TE metric values configured under SR-TE. See Flexible Algorithm-Specific TE Metric, on page 15.	IOS XR IS-IS Flex Algo implementation processes the link TE metric value received in the IS-IS ASLA. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks. If incoming ASLA includes the L-Flag, implementation derives the link TE metric value from the IS-IS Extended Reachability TLV.
Link Admin Group/Extended Admin Group	IOS XR IS-IS Flex Algo implementation advertises the affinity value as both the link admin group (AG) TLV and the link extended admin group (EAG) TLV using the IS-IS ASLA when its value falls within the first 32 bits. Otherwise, the affinity value is advertised only as link EAG TLV using the IS-IS ASLA. The admin group values advertised are configured directly under the IGP and are therefore FA-specific.	IOS XR IS-IS Flex Algo implementation processes the affinity value received as either the link admin group TLV or link extended admin group TLV in the IS-IS ASLA. ASLA sub-TLV is supported with non-zero-length or with zero-length Application Identifier Bit Masks. If incoming ASLA includes the L-Flag, implementation derives the affinity value from the IS-IS Extended Reachability TLV.

Link Attribute	Transmit	Receive
Link SRLG	IOS XR IS-IS LFA implementation advertises the link SRLG value in the IS-IS ASLA.	IOS XR IS-IS LFA implementation processes the link SRLG value received in the IS-IS ASLA. If incoming ASLA includes the L-Flag, implementation derives the link SRLG value from the IS-IS Extended Reachability TLV.

Strict IS-IS ASLA Link Attribute

Use the following command to configure the IOS XR IS-IS Flex Algo implementation to strictly use the link delay metric value received in the IS-IS ASLA:

```
router isis instance-id receive application flex-algo delay app-only
```

Flexible Algorithm-Specific TE Metric

Use the following command to configure the Flexible Algorithm-specific TE metric value under IS-IS, where *metric_value* is from 1 to 16777214:

- **router isis** *instance* **interface** *type interface-path-id* **address-family** { **ipv4** | **ipv6**} [**unicast**] **te-metric flex-algo** *metric_value* [**level** {**1** | **2**}]

The following example shows how to configure the IS-IS Flexible Algorithm-specific TE metric value to 50:

```
Router(config)# router isis 1
Router(config-isis)# interface HundredGigE 0/0/0/2
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# te-metric flex-algo 50
```

Use the following command to configure the Flexible Algorithm-specific TE metric value under OSPF, where *metric_value* is from 1 to 2147483647:

- **router ospf** *process-name* **area** *area* **interface** *type interface-path-id* **te-metric flex-algo** *metric_value*

The following example shows how to configure the OSPF Flexible Algorithm-specific TE metric value to 50:

```
Router(config)# router ospf 1
Router(config-ospf)# area 0
Router(config-ospf-ar)# interface HundredGigE 0/0/0/2
Router(config-ospf-ar-if# te-metric flex-algo 50
```

Flexible Algorithm with Exclude SRLG Constraint

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
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.
IS-IS Flexible Algorithm: Exclude-SRLG Constraint	Release 7.5.1	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. The ability to exclude the at-risk links ensures that the rest of the links in the network remain unaffected. This allows the setup of disjoint paths between two or more Flex Algos by leveraging deployed SRLG configurations.

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:

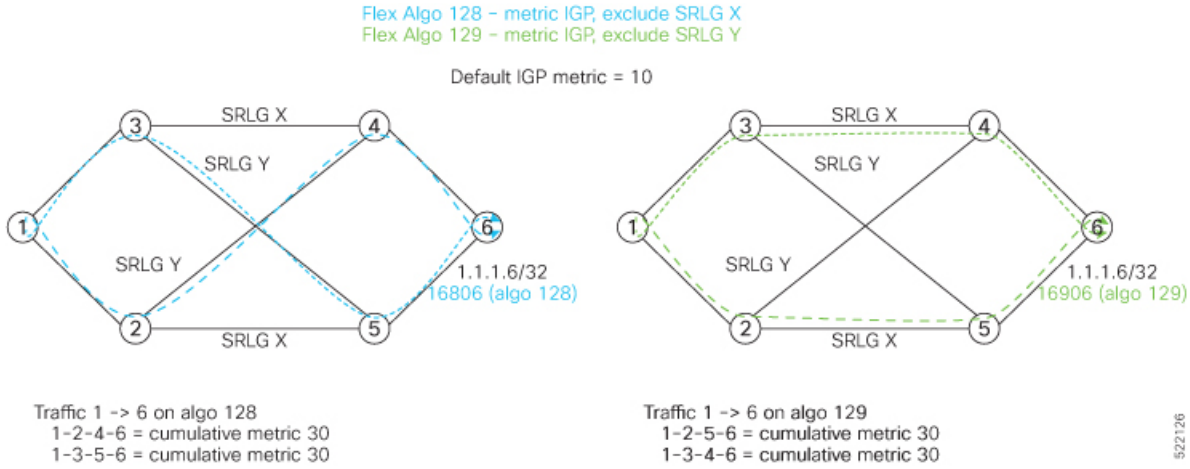
- Flex algo 128: metric IGP and exclude SRLG X constraint

- 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 MPLS Traffic Engineering Shared Risk Link Groups 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 12: 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 13: 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

Table 14: Feature History Table

Feature Name	Release	Description
IS-IS: Maximum Paths Per Flexible Algorithm	Release 7.8.1	This feature introduces a new subcommand under flex-algo command. This feature allows for maximum number of Equal-Cost Multi-path (ECMP) to be set for individual Flex Algorithms

A new subcommand under **flex-algo** is introduced.

The **flex-algo** command now includes the **address-family** <*ipv4*/*ipv6*> **unicast** subcommand, and the **maximum-paths** <*maximum-paths*> subcommand.

Note For information on IS-IS Algo0 Maximum Paths, refer to the "Implementing IS-IS" chapter in the *Routing Configuration Guide for Cisco NCS 560 Series Routers*.

The new subcommands allow for maximum number of Equal-Cost Multi-path (ECMP) to be set for individual algorithms. The value that is configured on a per-flex-algo per address-family basis overrides any value that is configured under the IS-IS global address-family submode.

Usage Guidelines and Limitations

- The **maximum-paths** configuration is not part of the Flexible Algorithm Definitions (FAD). If the advertised definition is configured for the flexible algorithm, the **maximum-paths** will not be propagated by the IS-IS.
- The maximum-paths per algorithm takes precedence over maximum-paths per address-family.
- The maximum paths effective for each algorithm are as follows:
  - For Flex-Algo 128:
    - IPv4: 5
    - IPv6: 3
  - For Flex-Algos 129 through 255:
    - IPv4: 12
    - IPv6:8

Configuration Example - Max Path

This example shows how you can set the maximum paths per-Flex-Algo:

```
Router(config)# router isis 10
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# metric-type te
Router(config-isis-flex-algo)# address-family ipv4 unicast
Router(config-isis-flex-algo-af)# maximum-paths 5
Router(config-isis-flex-algo)# exit
Router(config-isis-flex-algo-af)# address-family ipv6 unicast
Router(config-isis-flex-algo-af)# maximum-paths 3
Router(config-isis-flex-algo-af)# exit
```

Maximum Paths Per IS-IS Flexible Algorithm Per Prefix

Table 15: Feature History Table

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 NCS 560 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
```

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 16: Feature History Table

Feature Name	Release Information	Feature Description
User-Defined Generic Metric Support for IS-IS Flex Algo	Release 24.2.11	This feature now extends support on the Cisco NCS 540 Series routers running on Cisco IOS XR7.

Feature Name	Release Information	Feature Description
User-Defined Generic Metric Support for IS-IS Flex Algo	Release 24.2.1	Introduced in this release on the following Cisco NCS 540 router variants running on Cisco IOS XR: • N540-ACC-SYS • N540X-ACC-SYS • N540-24Z8Q2C-SYS 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.

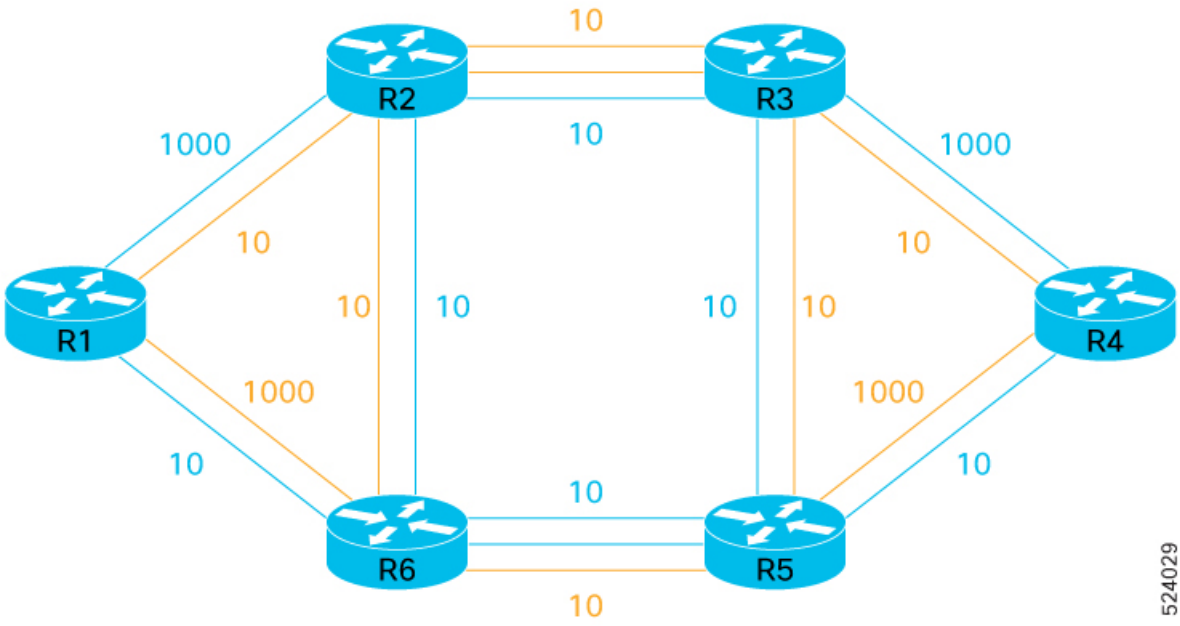
Apply User-Defined Generic Metrics in Flexible Algorithm Definition

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 2: 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
```

Example: Traffic Steering to Flexible Algorithm Paths

BGP Routes on PE – Color Based Steering

SR-TE On Demand Next-Hop (ODN) feature can be used to steer the BGP traffic towards the Flexible Algorithm paths.

The following example configuration shows how to setup BGP steering local policy, assuming two router: R1 (2.2.2.2) and R2 (4.4.4.4), in the topology.

Configuration on router R1:

```
vrf Test
address-family ipv4 unicast
  import route-target
   1:150
  !
  export route-policy SET_COLOR_RED_HI_BW
  export route-target
   1:150
  !
!
!
interface Loopback0
ipv4 address 2.2.2.2 255.255.255.255
!
interface Loopback150
vrf Test
ipv4 address 2.2.2.222 255.255.255.255
!
interface TenGigE0/1/0/3/0
description exr1 to cxr1
ipv4 address 10.0.20.2 255.255.255.0
!
extcommunity-set opaque color129-red-igp
  129
end-set
!
route-policy PASS
  pass
end-policy
!
route-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
end-policy
!
router isis 1
```

```
is-type level-2-only
net 49.0001.0000.0000.0002.00
log adjacency changes
affinity-map RED bit-position 28
flex-algo 128
  priority 228
!
address-family ipv4 unicast
  metric-style wide
  advertise link attributes
  router-id 2.2.2.2
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 2
   prefix-sid algorithm 128 index 282
  !
!
interface TenGigE0/1/0/3/0
  point-to-point
  address-family ipv4 unicast
  !
!
!
router bgp 65000
bgp router-id 2.2.2.2
address-family ipv4 unicast
!
address-family vpnv4 unicast
  retain route-target all
!
neighbor-group RR-services-group
  remote-as 65000
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
!
neighbor 4.4.4.4
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
segment-routing
traffic-eng
  logging
   policy status
  !
  segment-list sl-cxr1
   index 10 mpls label 16294
  !
  policy pol-foo
   color 129 end-point ipv4 4.4.4.4
   candidate-paths
    preference 100
     explicit segment-list sl-cxr1
     !
    !
   !
```

```
  !
!
!
```

Configuration on router R2:

```
vrf Test
address-family ipv4 unicast
  import route-target
   1:150
  !
  export route-policy SET_COLOR_RED_HI_BW
  export route-target
   1:150
  !
!
!
interface TenGigE0/1/0/1
description cxr1 to exr1
ipv4 address 10.0.20.1 255.255.255.0
!
extcommunity-set opaque color129-red-igp
  129
end-set
!
route-policy PASS
  pass
end-policy
!
route-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
end-policy
!
router isis 1
is-type level-2-only
net 49.0001.0000.0000.0004.00
log adjacency changes
affinity-map RED bit-position 28
affinity-map BLUE bit-position 29
affinity-map GREEN bit-position 30
flex-algo 128
  priority 228
!
flex-algo 129
  priority 229
!
flex-algo 130
  priority 230
!
address-family ipv4 unicast
  metric-style wide
  advertise link attributes
  router-id 4.4.4.4
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 4
   prefix-sid algorithm 128 index 284
   prefix-sid algorithm 129 index 294
   prefix-sid algorithm 130 index 304
  !
!
```

```
interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
!
interface TenGigE0/1/0/1
  point-to-point
  address-family ipv4 unicast
  !
!
router bgp 65000
bgp router-id 4.4.4.4
address-family ipv4 unicast
!
address-family vpnv4 unicast
!
neighbor-group RR-services-group
  remote-as 65000
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
!
neighbor 10.1.1.1
  use neighbor-group RR-services-group
!
neighbor 2.2.2.2
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 25.1.1.2
   remote-as 4
   address-family ipv4 unicast
    route-policy PASS in
    route-policy PASS out
   !
  !
!
!
segment-routing
!
end
```

Delay Normalization

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
SR-TE Delay Normalization for OSPF	Release 7.3.1	This feature extends the current Delay Normalization feature to support OSPF.

Performance measurement (PM) measures various link characteristics like packet loss and delay. Such characteristics can be used by IS-IS as a metric for Flexible Algorithm computation. Low latency routing using dynamic delay measurement is one of the primary use cases for Flexible Algorithm technology.

Delay is measured in microseconds. If delay values are taken as measured and used as link metrics during the IS-IS topology computation, some valid ECMP paths might be unused because of the negligible difference in the link delay.

The Delay Normalization feature computes a normalized delay value and uses the normalized value instead. This value is advertised and used as a metric during the Flexible Algorithm computation.

The normalization is performed when the delay is received from the delay measurement component. When the next value is received, it is normalized and compared to the previous saved normalized value. If the values are different, then the LSP generation is triggered.

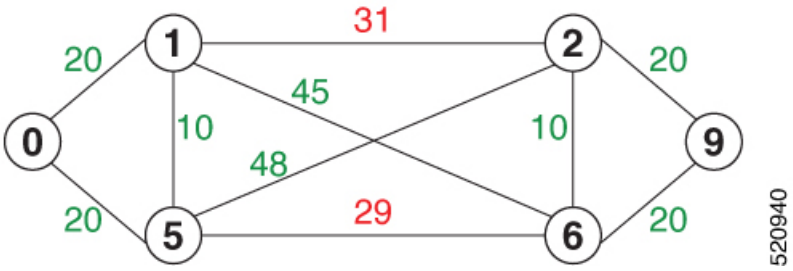
The following formula is used to calculate the normalized value:

- **Dm** – measured Delay
- **Int** – configured normalized Interval
- **Off** – configured normalized Offset (must be less than the normalized interval Int)
- **Dn** – normalized Delay
- **a** = Dm / Int (rounded down)
- **b** = a * Int + Off

If the measured delay (Dm) is less than or equal to **b**, then the normalized delay (Dn) is equal to **b**. Otherwise, Dn is **b + Int**.

Example

The following example shows a low-latency service. The intent is to avoid high-latency links (1-6, 5-2). Links 1-2 and 5-6 are both low-latency links. The measured latency is not equal, but the difference is insignificant.



We can normalize the measured latency before it is advertised and used by IS-IS. Consider a scenario with the following:

- Interval = 10
- Offset = 3

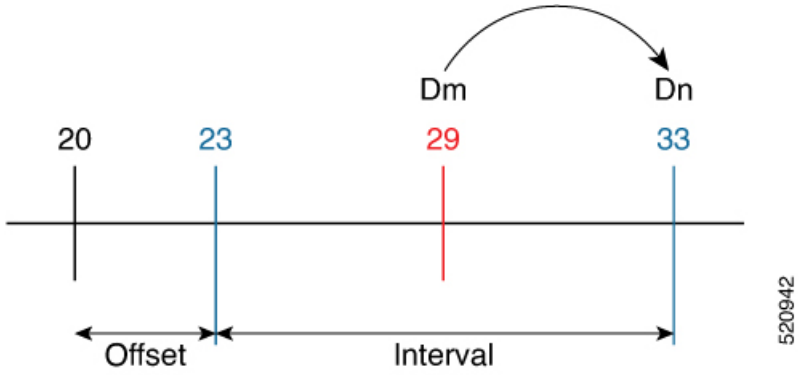
The measured delays will be normalized as follows:

- **Dm** = 29

  a = 29 / 10 = 2 (2.9, rounded down to 2)

b = 2 * 10 + 3 = 23

In this case, **Dm** (29) is greater than **b** (23); so **Dn** is equal to **b**+**I** (23 + 10) = **33**

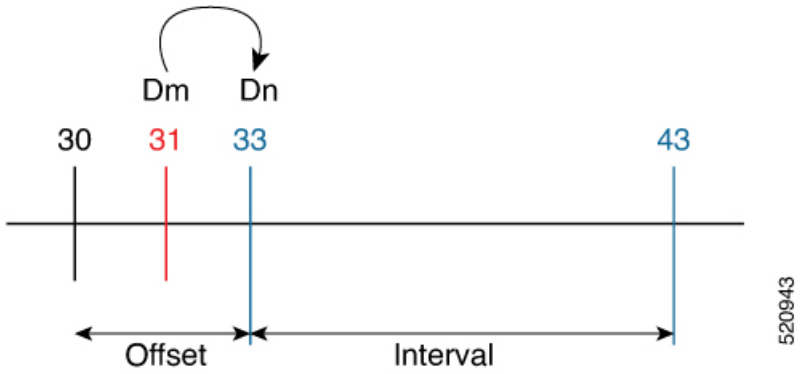


- **Dm** = 31

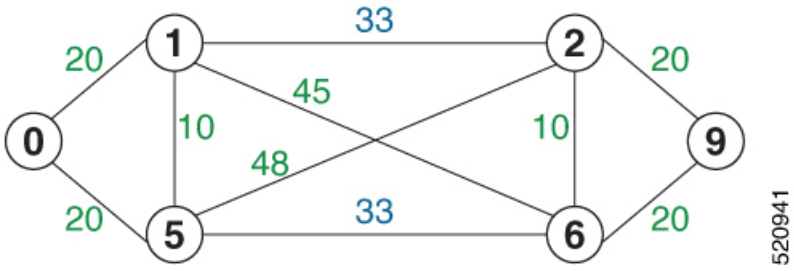
  a = 31 / 10 = 3 (3.1, rounded down to 3)

  b = 3 * 10 + 3 = 33

  In this case, **Dm** (31) is less than **b** (33); so **Dn** is **b** = **33**



The link delay between 1-2 and 5-6 is normalized to 33.



Configuration

Delay normalization is disabled by default. To enable and configure delay normalization, use the **delay normalize interval** *interval* [**offset** *offset*] command.

- *interval* – The value of the normalize interval in microseconds.

- *offset* – The value of the normalized offset in microseconds. This value must be smaller than the value of normalized interval.

IS-IS Configuration

```
router isis 1
 interface GigEth 0/0/0/0
  delay normalize interval 10 offset 3
  address-family ipv4 unicast
   metric 77
```

OSPF Configuration

```
router ospf 1
 area 0
  interface GigabitEthernet0/0/0/0
   delay normalize interval 10 offset 3
  !
 !
!
```