Description

IS-IS flexible algorithm (FlexAlgo) provides a lightweight, simplified mechanism for performing basic traffic engineering functions within a single IS-IS area. FlexAlgo requires the cooperation of all nodes within the IS-IS area but does not require an external controller. Paths are computed by each node within the area, resulting in an MPLS-switched forwarding path to nodes that are advertising a node Segment Identifier (SID) for the algorithm. The results of the path computation are placed in the colored tunnel RIB or system tunnel RIB, which simplifies route resolution.

Multiple algorithms may be used concurrently, allowing for different types of path computations to co-exist.

In addition to the normal IGP metric, algorithms may consider:

- The administrative group (color) assigned to a link
- A link's Shared Risk Link Group (SRLG)
- The statically configured delay on a link
- The Traffic Engineering Metric of a link

Platform compatibility

FlexAlgo is supported on all platforms that support SR-TE (Arista 7800R3, Arista 7500R/R2/R3, 7280R/R2/R3 and 7020R families).

Feature Support

As of 4.27.0F, FlexAlgo supports:

- Algorithm definition selection based on advertised priority.
- Path computation controlled by link administrative groups. Algorithms may:
    - Exclude administrative groups
    - Require that a link's administrative groups intersect with a set of administrative groups
    - Require that a link's administrative groups be a superset of a set of administrative groups.

- Path computation that excludes SRLGs
- Path computation based on the IGP metric or based on the statically configured minimum link delay
- Placing path computation results in the system tunnel RIB or the colored tunnel RIB
- Packet forwarding using the SR data plane
- Equal cost multi-path routes
- Operation in multi-instance networks
- Operation in multi-agent mode
- Management by OpenConfig

As of 4.30.1F, FlexAlgo also supports:

- Path computation based on the TE Metric of a link.

As of 4.30.2F, FlexAlgo also supports:

- Fallback to legacy TE SubTLVs in case of legacy devices.
- TI-LFA backup path computation for FlexAlgo node segments .

As of 4.31.2F, FlexAlgo also supports

- specifying extended administrative groups in the range of 0-127

As of 4.34.1F, FlexAlgo also supports

- Primary path computation for prefix/anycast segment
- TI-LFA backup path computation for prefix/anycast segment

As of 4.32.2F, FlexAlgo also supports :

- Handling of multiple Router Capability TLVs and multiple FAD TLVs

Configuration

FlexAlgo is configured through the following steps:

- **Configure the definition of an algorithm:** The constraints for the algorithm need to be defined and a name is assigned to the algorithm itself. In most cases, you will want to replicate the definition (and name) on all systems within the area so that management is simplified and route resolution through FlexAlgo paths can take place everywhere.
- **Bind the algorithm to IS-IS:** Each node that will participate in the algorithm must be configured to use that algorithm for the specific IS-IS levels where it will be used. The binding also specifies which systems will advertise the definition of the algorithm. For robustness, we recommend advertising the definition of the algorithm on at least three systems within the area.
- **Configure route resolution against the FlexAlgo results:** FlexAlgo does not directly affect traffic paths. The results are placed in the colored tunnel RIB or system tunnel RIB. Other protocols, usually BGP, are configured to resolve paths using these RIBs and in turn, make use of the FlexAlgo paths.

Configure algorithm definitions

FlexAlgo can configure up to 128 different algorithms concurrently, using identifiers 128-255. Each algorithm identifier corresponds to an algorithm name, configured under ***"router traffic-engineering"*** mode, underneath the ***"flex-algo"*** submode. This must be configured on each system that will participate in the algorithm. As previously already mentioned, for consistency,

we strongly recommend using the same algorithm name for the same algorithm identifier throughout the network. The syntax for assigning a name to an algorithm identifier is:

```
[no | default] flex-algo <algo-id> <name>
```

For example, a simple algorithm definition might be configured with:

```
router traffic-engineering
   flex-algo
      flex-algo 128 HFT
```

Within an algorithm definition, you may optionally configure one or more constraints on the path computed by the algorithm and properties of the algorithm. All of the constraints and properties listed below are configured under the algorithm definition mode.

Configure algorithm definition priority

Multiple nodes within the network may (and should) advertise definitions for the algorithm. Normally, all of these definitions should be identical. However, when making configuration changes to the network, it is useful to have the newer definition have priority over the older definition. The priority is configured as a value 0-255 within the algorithm definition with higher values being preferred. The default priority is 237. The syntax of the priority configuration is:

```
[no | default] priority <prio>
```

Configure algorithm color

The results of a FlexAlgo computation are MPLS switched paths to node SIDs advertised by other nodes. These are placed in the colored tunnel RIB if color is configured for an algorithm or the system tunnel RIB if no color is configured. Color is a number 0-4294967295 and can be configured using the syntax:

```
[no | default] color <value>
```

The algorithm color is not advertised as part of its definition, so the color must be configured on any node that will produce results that should be placed in the colored tunnel RIB.

Configure algorithm metric

FlexAlgo can make use of the normal IGP metric, the minimum delay for a link or the TE metric of a link. The default is to use the normal IGP metric. Which metric to use can be configured with:

```
[no | default] metric <metric>
```

The metric value can be ***"igp-metric"*** , ***"min-delay"*** , ***"te-metric"*** , ***"0"*** (IGP metric), ***"1"*** (min-delay) or ***"2"*** (TE metric).

Configure administrative groups

Links in the topology can be assigned to one or more administrative groups. The possible groups are numbered 0-31.

As of EOS-4.31.2F, it is possible to use administrative groups in the range of 0-127.

Constraints on a FlexAlgo path can then be specified by rejecting links with a given administrative group, requiring a set of administrative groups, or permitting the usage of any of a set of administrative groups.

A set of administrative groups is described as a comma-separated list of individual group numbers or ranges. Each range has the syntax ***"<number1>-<number2>"*** , where the value of <number1> must be less than the value of <number2> . An example set of administrative groups might appear as:

```
0,3,5-7,9,11-13,27
```

As of EOS-4.31.2F an example set of administrative groups might appear as:

```
0,5-7,9,11-13,27,100,105-108,127
```

- Links may be rejected from consideration by configuring a set of administrative groups with the ***"exclude"*** keyword. If a link is configured with an administrative group that is one of the groups listed in the ***"exclude"*** set, then the link will not be considered during path computation.

- Algorithms may also be configured using an ***"include all"*** set of administrative groups. If a link is configured with all of the administrative groups within this set, then it is eligible to be part of the resulting path.

- Links may be required to belong to a set of administrative groups by configuring the ***"include any"*** constraint. A link must have at least one of the administrative groups in this

set to be considered as part of the path.

The syntax for configuring all of these sets of administrative groups is:

```
[no | default] administrative-
group [include all <set>] [include any <set>] [exclude <set>]
```

An example of a configuration of administrative groups might be:

```
administrative-group include all 1-3 include any 4-6 exclude 7-9
```

Configure Shared Risk Link Groups

A Shared Risk Link Group is a set of links that are likely to fail together. This may be because, for example, they are different wavelengths multiplexed over the same fiber, separate fibers in the same cable, or fibers run through the same conduit. SRLGs are similar to ***"exclude"*** administrative groups. An SRLG can be a number, 0-4294967295, or a name. An algorithm can be configured to exclude a set of SRLGs with the syntax:

```
srlg exclude <list>
```

The ***"list"*** is a list of SRLG numbers and names separated by spaces.

Example of a complex definition

The constraints and parameters above can be combined under an algorithm definition. A complex algorithm might look like this:

```
router traffic-engineering
   flex-algo
      flex-algo 128 HFT
         priority 85
         color 47
         metric min-delay
         administrative-group include all 0,3,5 include any 2,4,6 exclude 7-15
         srlg exclude flag marea warf 101 680 880 17
```

Configure algorithm bindings

Once an algorithm has been defined, it must be invoked by IS-IS to become operational. An algorithm is bound to IS-IS under the ***"router isis"*** mode, within the ***"segment-routing mpls"*** submode using the syntax:

```
[no | default] flex-algo <name> [level-1 | level-2 | level-1-2] [advertised]
```

If a level qualifier is specified, then the algorithm will be operational within that level. If no level is specified, then the algorithm will be operational in both level 1 and level 2.

If the ***"advertised"*** qualifier is specified, then the local definition of the algorithm will be advertised to the rest of the network. If it is the selected definition (as you might have competing definitions), it will be used by all participants in the algorithm. For example,

```
router isis Amun
   ...
   segment-routing mpls
      flex-algo HFT level-1 advertised
```

This configuration would enable algorithm HFT within level 1 IS-IS and advertise the definition above to the network.

Configure interface delay

The one-way minimum delay on an interface can be configured under the interface mode using the syntax:

```
[no | default] traffic-engineering min-delay static <value> <units>
```

In this command, ***"<value>"*** is a number 0-16777215 and ***"<units>"*** is either ***"microseconds"*** or ***"milliseconds"*** . This is a statically configured value and will not change even if the latency of the link changes.

An easy way to estimate the one-way delay on the interface is to use the ***"ping"*** command to ping the interface on the opposite side of the link when the link is uncongested and divide the minimum resulting time by two. For example:

```
#ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 72(100) bytes of data.
80 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=2.77 ms
80 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=1.67 ms
```

```
80 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=1.76 ms
80 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=1.78 ms
80 bytes from 10.0.1.2: icmp_seq=5 ttl=64 time=1.67 ms

--- 10.0.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 1.672/1.933/2.776/0.427 ms, ipg/ewma 2.750/2.340 ms
```

This suggests a one-way delay of 1672/2 = 836 microseconds. Point-to-point links are usually symmetric and the other end of the link should be configured with the same value. LAN links should all have the same value configured, usually.

Configure interface administrative groups

Administrative groups can be assigned to an interface within interface mode using the syntax:

```
[no | default] traffic-engineering administrative-group <set>
```

In this configuration, ***"<set>"*** is a comma-separated list of group numbers or ranges, as described above or a hexadecimal bit-mask, 0x0-0xFFFFFFFF, of the groups to be included with the least significant bit representing group 0 and the most significant bit representing bit 31.

As of EOS-4.31.2F it is possible to configure administrative groups on an interface in the extended range of 0-127.

The hex value in the set can appear once and will only specify values for the 0-31 range, not the extended administrative group range (>31). Each range has the syntax "<number1>-<number2>", where the value of <number1> must be less than the value of <number2>.

Example, as of EOS-4.31.2F:

```
interface eth1
   traffic-engineering administrative-group 0xA,RED,31-33,127
```

Point-to-point links are normally configured with the same administrative groups on both ends. LANs are usually configured with the same administrative groups on all connected interfaces.

Configure interface Shared Risk Link Groups (SRLGs)

Shared Risk Link Groups (SRLGs) can be assigned to an interface within interface mode using the syntax:

```
[no | default] traffic-engineering srlg (<name> | <number>)
```

This command can be repeated if there are multiple SRLGs for an interface. If a name is specified, it should be defined using the ***"srlg"*** command under ***"router traffic-engineering"*** . (See the TOI for IS-IS Traffic Engineering)

Point-to-point links are normally configured with the same SRLGs on both ends. LANs are usually configured with the same SRLGs on all connected interfaces.

Configure interface Traffic Engineering Metric

Traffic Engineering Metric can be assigned to an interface within interface mode using the syntax:

```
[no | default] traffic-engineering metric <value>
```

In this command, ***"<value>"*** is a number 1-16777215.

Point-to-point links are normally configured with the same TE Metric on both ends. LANs are usually configured with the same TE Metric on all connected interfaces.

Configure fallback to Legacy TE SubTLV

In the case of devices in the network that do not support the advertisement of ASLA SubTLVs, this feature makes it possible to fall back to Legacy TE SubTLVs. See mode of operation:

	Fallback mechanism configuration	ASLA Received	SubTLV used
1	Off	No	ASLA
2	Off	Yes	ASLA
3	On	No	Legacy
4	On	Yes	ASLA

The fallback mechanism works on a per-link basis, meaning that if any ASLA SubTLV is received for a given link, then fallback will not take place. The fallback mechanism can be configured under ***"router traffic-engineering"*** mode, underneath the ***"flex-algo"*** submode using

the syntax:

```
[ no | default ] link-attributes asla fallback
```

Configure Node-SID for Flex Algo

Node Segment Identifiers (SIDs) are the endpoints for FlexAlgo path computations. Each system in the network will compute a path to each node SID that is participating in the algorithm, using the constraints specified in the selected FlexAlgo definition. A node SID is typically configured on a loopback interface that is configured with a /32 IPv4 address or a /128 IPv6 address. A node SID can be configured to participate in a particular algorithm within interface configuration mode using the syntax:

```
[no | default] node-segment (ipv4 | ipv6) index <value> [flex-algo <name>]
```

In this command ***"ipv4"*** or ***"ipv6"*** selects the address family for the SID. The ***"<value>"*** specifies the index of the SID relative to the node's global block, and the ***"<name>"*** specifies the mandatory algorithm name if the SID is to be used with FlexAlgo. We allow one SID to be configured for each algorithm for each interface. Different SIDs for different algorithms can be configured on the same interface.

Configure Prefix-SID for Flex-Algo

As of 4.34.1F, EOS supports FlexAlgo path computation for prefix-segments.

Prefix Segment Identifiers for Flex-Algo are segments associated with an IS-IS prefix for which a router is originating an IP Reachability TLV in that specific Flex-Algo.

Similar to Node-SIDs, Prefix-SIDs are also the endpoints for FlexAlgo path computation. Each system in the network will compute a path to each Prefix-SID that is participating in the algorithm, using the constraints specified in the selected FlexAlgo definition.

Prefix Segments are configured under segment-routing mpls configuration mode in IS-IS.

```
[no | default] prefix-
segment <ipv4-prefix/ipv6-prefix> (index <value>/label <value) flex-algo <name>
```

In this command, ipv4-prefix or ipv6-prefix specifies the ipv4 prefix or ipv6 prefix endpoint for which path is being computed and ***"index <value>"*** represents the index relative to the node's global block or it represents the absolute label if configured with ***"label <value>"*** and ***"<name>"*** specifies the mandatory algorithm name if the SID is to be used with FlexAlgo. We allow one

Prefix-SID to be configured for each algorithm for a specific prefix.

Example Config

```
router isis Arista
   ...
   segment-routing mpls
      prefix-segment 1.1.1.1/32 index 10 flex-algo HFT
      prefix-segment 200::1/128 label 900456 flex-algo HFT
      flex-algo HFT
```

Configure Anycast-SID for Flex-Algo

As of 4.34.1F, EOS supports FlexAlgo path computation for anycast-segments.

Anycast-SID is a prefix segment which does not identify a specific router, but a set of routers. Anycast-SID enforces the ECMP-aware shortest-path forwarding towards the closest node of the anycast set.

Anycast-SID for FlexAlgo are same prefix-segments configured on multiple routers with same SID under same FlexAlgo such that the path to this anycast prefix would be via the topologically nearest node in the anycast group participating in that FlexAlgo.

Note that for Anycast-SID to work as expected, the SRGB(Segment Routing Global Block) on the members of the anycast group should be the same.

As of 4.34.1F, Prefix/Anycast segment with FlexAlgo should be configured with /32 or /128 mask length.

Configuring TI-LFA protection for FlexAlgo node segments

As of 4.30.2F , we will compute the TI-LFA backup path for FlexAlgo node segments. There is no specific knob to turn on TI-LFA support for FlexAlgo node segments, so FlexAlgo paths will automatically get protected once a system is upgraded from an older release to 4.30.2 and TI-LFA is configured.

As of 4.34.1F, we will compute the TI-LFA backup paths for FlexAlgo Prefix/Anycast segments as well. There is no specific knob to turn on the TI-LFA support for FlexAlgo Prefix/Anycast segments.

To enable link/node protection for node segments of a specific address-family learned on all IS-IS interfaces or for prefix/anycast segments learned over IS-IS, the following command is used in the address-family sub-mode of the router isis mode. Note that FRR using TI-LFA is disabled globally by default in the router IS-IS address-family sub-modes.

```
[no | default] fast-reroute ti-lfa mode [( [level-1 | level-2]) | disabled]
```

To enable SRLG protection on all interfaces, the following command is used. This command is used in addition to configuring link-protection or node-protection. If SRLG protection is enabled, the backup paths will be computed after excluding all the links that share the same SRLG with the active link that is being used by all flex-algo node-segments and/or prefix/anycast segments .

```
[no | default] fast-reroute ti-lfa srlg [strict | disabled]
```

Selective TI-LFA protection can be enabled on a particular interface instead of applying it to all interfaces. The following commands are used in the interface configuration mode.

```
DUT(config-if-Et1)# [no | default] isis [ipv4 | ipv6] fast-reroute ti-
lfa mode  [level-1 | level-2]

DUT(config-if-Et1)# [no | default] isis [ipv4 | ipv6] fast-reroute ti-
lfa srlg [strict | disabled]
```

Note that when TI-LFA protection is disabled on a specific interface while being activated globally, tunnels passing through that interface will not be protected. However, protection will still be applied to all the tunnels that traverse other interfaces.

Please refer to “ Support for TI-LFA FRR using IS-IS Segment Routing” TOI for more information on other configurations.

Configuring tunnel preference for FlexAlgo tunnels

By default, IS-IS SR tunnels and IS-IS FlexAlgo tunnels have the same preference of 65. This means that when the same node advertises a segment with the default algorithm (algorithm 0) and a colorless FlexAlgo segment then the tunnels corresponding to the default algorithm segment and the FlexAlgo segment will form ECMP paths and any route resolving to this next hop will have both of these tunnels in its path.

To make the FlexAlgo tunnel more preferred or less preferred than the SR tunnel, the preference of the FlexAlgo tunnel can be updated using the following command:

```
[ default ] source-protocol isis flex-algo preference <preference>
```

For the preference, smaller is more preferred.

For example:

```
tunnel-ribs
   tunnel-rib system-tunnel-rib
      source-protocol nexthop-group
      source-protocol rsvp-ler
      source-protocol bgp labeled-unicast
      source-protocol static
      source-protocol ldp
      source-protocol isis flex-algo preference 50
      source-protocol isis segment-routing
```

Since the IS-IS FlexAlgo tunnel preference is changed to 50, it will be preferred over the IS-IS SR tunnel which has a default preference of 65.

Preference can also be configured per Algorithm using the following command:

```
[ default ] source-protocol isis flex-algo ALGO_NAME preference <preference>
```

An algo-preference takes precedence over the overall “flex-algo” preference. Example:

```
tunnel-ribs
   tunnel-rib system-tunnel-rib
      source-protocol isis flex-algo preference 50
      source-protocol isis flex-algo ALG1 preference 60
      source-protocol isis segment-routing
```

In this case, tunnels corresponding to algorithm ALG1 will use the preference value of 60 whereas other flex-algo tunnels that might have been configured on the switch will use a preference value of 50.

Configure route resolution

IS-IS FlexAlgo tunnels will be installed either in the system tunnel or system colored tunnel RIB, depending on the “***color***” configuration. The key for tunnel entries in the system tunnel RIB is just the prefix, whereas the key for tunnel entries in the system colored tunnel RIB is prefix + color. A BGP route associated with an extended color community (a.k.a color) will first try to resolve over the system colored tunnel RIB and then over the system tunnel RIB. To add an

extended color community to a BGP route, add a route map and then associate the route-map to the BGP neighbor command, for example:

```
route-map bgp100 permit 10
   set extcommunity color 100


router bgp 100
   router-id 252.252.1.252
...
   neighbor 1.0.5.1 route-map bgp100 out
   neighbor 1.0.5.1 send-community
```

The above command sets extended community 100 to all routes advertised to neighbor 1.0.5.1. The ***"neighbor <neighbor> send-community"*** command is necessary to advertise the extended community. It can also be applied on the inward direction using ***"neighbor 1.0.5.1 route-map bgp100 in"*** which will apply extended community 100 to all routes received from peer 1.0.5.1

Such routes associated with the extended community will resolve over FlexAlgo tunnel present in the system colored tunnel RIB with a color that matches the extended community.

Multiple Router Capability TLV handling behaviour

- On reception of more than one Router Capability TLVs which may each contain one or more Flex-algo Definition TLVs, in accordance with RFC, we combine the unique values inside Exclude-SRLG sub-TLVs and for other sub-TLVs, only their first advertisement in lowest numbered LSP is processed .

Show commands

Show commands for algorithm bindings and definitions

The command ***"show isis flex-algo"*** lists the algorithms that the local system is participating in, whether or not the local definition is advertised, the IS-IS levels that are using the algorithm, the metric being used for the path computation, and which node advertised the selected definition of the algorithm:

```
#show isis flex-algo
IS-IS Instance: Amun VRF: default

Algorithm Advertised Level Metric    Selected
--------- ---------- ----- --------- --------
HFT       yes        L1    min-delay ip8
```

```
VPN       yes        L1    default   ip8
BULK      yes        L1    TE        ip6
```

The details about the algorithm definition can be viewed with the ***"show isis flex-algo algorithm <name>"*** command.

```
#show isis flex-algo algorithm HFT

IS-IS Instance: Amun VRF: default

Algorithm Priority Level Advertiser
--------- -------- ----- ----------
HFT              8 L1    ip8

Parameters:
advertised
metric min-delay
administrative-group include any 5,10
color 1

Other advertisers: ip1, ip2, ip3, ip4, ip5, ip6
```

This shows the definition of the algorithm that has been selected for the algorithm, the priority of the advertisement, and which system is advertising the algorithm. The parameters for the definition are shown, as well as the local color. The names of other systems in the network that are also advertising a definition for the algorithm are also shown.

The definition being advertised by another system can be shown by adding the ***"system <name>"*** qualifier:

```
#show isis flex-algo algorithm HFT system ip5

IS-IS Instance: Amun VRF: default

Algorithm Priority Level Advertiser
--------- -------- ----- ----------
HFT              5 L1    ip5

Parameters:
advertised
metric min-delay
administrative-group include any 5,10
```

```
color 1

Other advertisers: ip1, ip2, ip3, ip4, ip6, ip8
```

As of 4.34.2F, on reception of flex-algo definition TLVs with differing fixed portions, a warning is printed as shown below -

```
#show isis flex-algo algorithm red system host1

!Ignored conflicting flex-algo definition sub-
TLV with fixed portion mismatch from 1111.1111.1002 (host1)
IS-IS Instance: 1 VRF: default
Algorithm Priority Level Advertiser
--------- -------- ----- ----------
red              0 L1    host1

Parameters:
advertised
metric min-delay
administrative-group include all 0 include any 0 exclude 0
srlg exclude 111 112

Other advertisers: host2, host3
```

To see the priorities of the algorithm advertisers, you can use the ***"routers"*** qualifier:

```
#show isis flex-algo algorithm HFT routers

IS-IS Instance: Amun VRF: default

Router Level Advertising Priority
------ ----- ----------- --------
ip1    L1    yes                1
ip2    L1    yes                2
ip3    L1    yes                3
ip4    L1    yes                4
ip5    L1    yes                5
ip6    L1    yes                6
ip8    L1    yes                8
```

If an algorithm is active in multiple IS-IS levels, you can use the ***"level-1"*** or ***"level-2"*** qualifiers to restrict what is displayed.

To see the priorities and algorithm advertisers across all algorithms, you can use the ***"show isis flex-algo routers"*** command:

```
#show isis flex-algo routers

IS-IS Instance: Amun VRF: default

Algorithm: Bulk

Router Level Advertising Priority
------ ----- ----------- --------
ip6    L1    yes                6
ip7    L1    yes                7
ip8    L1    yes                8
ip6    L2    yes                6
ip7    L2    yes                7
ip8    L2    yes                8

Algorithm: HFT

Router Level Advertising Priority
------ ----- ----------- --------
ip1    L1    yes                1
ip2    L1    yes                2
ip3    L1    yes                3
ip4    L1    no
ip5    L1    yes                5
ip6    L1    yes                6
ip8    L1    yes                8
ip1    L2    yes                1
```

Show traffic engineering interface attributes

The traffic engineering attributes that have been configured on an interface can be seen with ***"show traffic-engineering interfaces"*** :

```
#show traffic-engineering interfaces
Interface Ethernet1:
Traffic engineering metric: 30
Administrative groups: 5
Shared Risk Link Groups: stl, marea, ecfs
```

```
Minimum delay: 2 milliseconds (static)

Interface Ethernet2:
Administrative groups: 10
Shared Risk Link Groups: stl, marea, ecfs
Minimum delay: 10 milliseconds (static)
```

Show the FlexAlgo segments

The command ***"show isis segment-routing prefix-segments"*** will show the algorithms advertised along with a prefix-segment.

```
#show isis segment-routing prefix-segment
System ID: rtrrsvp1                    Instance: 'Amun'
SR supported Data-plane: MPLS                  SR Router ID: 10.0.2.1

Node: 3      Proxy-Node: 0      Prefix: 0       Total Segments: 3

Flag Descriptions: R: Re-advertised, N: Node Segment, P: no-PHP
                   E: Explicit-NULL, V: Value, L: Local
Segment status codes: * - Self originated Prefix, L1 - level 1, L2 - level 2, ! - SR-
unreachable,
                      # - Some IS-IS next-hops are SR-unreachable
  Prefix                       SID Type       Flags                    System ID
 Level Protection  Algorithm
   ------------------------- ----- ---------- ---------------------- ---------------
 ----- ----------- -------------
   20.0.0.1/32                 100 Node       R:0 N:1 P:0 E:0 V:0 L:0 rtrrsvp4
 L1    unprotected SPF
   20.0.0.1/32                 200 Node       R:0 N:1 P:0 E:0 V:0 L:0 rtrrsvp4
 L1    unprotected HFT
   20.0.0.1/32                 300 Node       R:0 N:1 P:0 E:0 V:0 L:0 rtrrsvp4
 L1    unprotected VPN
```

The last column shows the algorithm associated with the segment. The SPF algorithm is the default algorithm, i.e., algorithm 0. HFT and VPN are the algorithms defined on this node.

Show the paths that have been computed

The results of the CSPF path computation including the next hops to the node advertising a FlexAlgo segment can be shown using ***"show isis flex-algo path."***

```
#show isis flex-algo path
Flex algo paths for IPv4 address family
Topology ID: Level-1
   Destination       Algorithm       Next Hop    Interface
----------------- --------------- -------------- ---------
   rtrrsvp4          HFT             10.0.1.3    Ethernet2

   rtrrsvp4          VPN             10.0.0.2    Ethernet1
```

In this case, rtrrsvp4 is the node that advertised two segments, one with the HFT algorithm and another with the VPN algorithm. The outgoing interface and next-hop information are shown in the output above.

As of 4.34.1F, ***"show isis flex-algo path"*** shows flex-algo paths computed for prefix/anycast segment.

```
#show isis flex-algo path
Flex algo paths for IPv4 address family
Topology ID: Level-1
   Destination           Algorithm       Next Hop         Interface
--------------------- --------------- ------------------ ------------
   1.1.1.1/32            HFT             10.0.1.3         Ethernet2

   2.2.2.2/32            VPN             10.0.0.2         Ethernet1
```

As of 4.30.2, ***"show isis ti-lfa path"*** shows the repair path for FlexAlgo(s) also, with the list of all the system IDs from the P-node to the Q-node for every destination and algorithms.

```
#show isis ti-lfa path 1111.1111.1004
TI-LFA paths for IPv4 address family
   Topology ID: Level-1
   Destination       Algorithm        Constraint                      Path
----------------- ---------------- --------------------------------- --------------

   1111.1111.1004    HFT              exclude link Ethernet2         1111.1111.1003
                                                                     1111.1111.1001

                     VPN              exclude link Ethernet1         1111.1111.1002
```

As of 4.34.1F, ***"show isis ti-lfa path"*** shows repair path for FlexAlgos computed for prefix/anycast

segment.

```
#show isis ti-lfa path 9.9.9.9/32
TI-LFA paths for IPv4 address family
   Topology ID: Level-1
   Destination       Algorithm       Constraint              Path
----------------- --------------- -------------------------- -----
   9.9.9.9/32        HFT            exclude Ethernet38/1     1111.1111.1003
```

To see the MPLS LFIB route corresponding to the FlexAlgo segments use the “show mpls lfib route” command:

```
#show mpls lfib route
--- snip ---
IP    900200   [1], 20.0.0.1/32, algorithm HFT
               via M, 10.0.1.3, swap 900200
                payload autoDecide, ttlMode uniform, apply egress-acl
                interface Ethernet2
IP    900300   [1], 20.0.0.1/32, algorithm VPN
               via TI-LFA tunnel index 1, pop
                 payload autoDecide, ttlMode uniform, apply egress-acl
                    via 10.0.1.2, Vlan2968, label imp-null(3)
                    backup via 10.0.0.2, Vlan2387, label 900102

--- snip ---
```

Show the tunnel RIBs

Every FlexAlgo segment will install a tunnel entry in the system tunnel or system colored tunnel RIB, depending on the ***“color”*** configuration.

If there are multiple algorithms advertised by the same node for the same prefix, then they will form parallel paths as shown in the ***“show tunnel rib brief”*** output:

```
#show tunnel rib brief
Tunnel RIB: system-tunnel-rib
   Endpoint          Tunnel Type          Index(es)       Tunnel Preference       IGP
 Preference       IGP Metric    Metric Type
----------------- ------------------- -------------- ---------------------- ------
-------------- ---------------- -----------
   20.0.0.1/32       IS-IS FlexAlgo       1, 2            65                      0
                  2000          metric
```

The tunnel with indexes 1 and 2 can be seen in the ***"show tunnel fib"*** output:

```
#show tunnel fib
Type 'IS-
IS FlexAlgo', index 1, endpoint 20.0.0.1/32, algorithm HFT, forwarding Primary
   via 10.0.1.3, 'Ethernet2' label 900200

Type 'IS-
IS FlexAlgo', index 2, endpoint 20.0.0.1/32, algorithm VPN, forwarding Primary
   via TI-LFA tunnel index 1, pop
      via 10.0.1.2, Vlan2968, label imp-null(3)
      backup via 10.0.0.2, Vlan2387, label 900102
```

The FlexAlgo tunnel will be moved from the system tunnel to the system colored tunnel table when a color configuration is present for the particular algorithm.

If a color value of 115 is configured for the HFT algorithm, then the FlexAlgo tunnel corresponding to the HFT algorithm will be moved to the colored tunnel RIB with color set as 115:

```
Tunnel RIB: system-colored-tunnel-rib
 Endpoint       Color    Tunnel Type       Index(es)    Tunnel Preference    IGP Pref
erence    IGP Metric   Metric Type
-------------- -------- ----------------- ------------ -------------------- ---------
-------- ------------ -----------
 20.0.0.1/32    115      IS-IS FlexAlgo    1            65                   0
          2000         metric
```

Show the resolved routes

Routes can be resolved over tunnel RIBs. The ***"show ip route"*** command shows the details of the resolved routes. For example, the output of a static route resolving over a FlexAlgo tunnel is shown below:

```
S        43.12.4.1/32 [1/0] via 20.0.0.1/32, algorithm HFT, IS-
IS FlexAlgo tunnel index 1
                              via 10.0.1.3, Ethernet2, label 900200
                           via TI-LFA tunnel index 1, pop
                              via 10.0.1.2, Vlan2968, label imp-null(3)
                              backup via 10.0.0.2, Vlan2387, label 900102
```

Troubleshooting

FlexAlgo is a complex, distributed feature that can be difficult to troubleshoot. We recommend the following steps:

- Determine which algorithm is not producing the expected results.
- Verify the configuration of the algorithm on all of the nodes participating in the algorithm.
- Are all of the definitions for the algorithm the same?
- Are all of the nodes selecting the same definition?
- Are the link attributes being advertised correctly?
- Are the link-state databases of all nodes synchronized?
- Using the ***"show isis flex-algo path"*** command, does the next hop appear to be correct everywhere along the path?
- Does it match the path shown by MPLS traceroute for our algorithm?
- Do the LFIB entries at each node look to be correct?
- Consider enabling tracing (see the "Tracing" paragraph below for further information).

MPLS PING

Use the ping command to verify the connectivity of a FlexAlgo tunnel:

```
# ping mpls segment-routing  <destination prefix> [algorithm <FlexAlgo name>]
```

The FlexAlgo name is provided to the ping command using the ***"algorithm"*** option. If the algorithm option is not provided, the ping will go through the normal SPF path.

Example output:

```
# ping mpls segment-routing ip 1.0.12.1/32 algorithm RED repeat 3
LSP ping to Segment-Routing route 1.0.12.1/32, algorithm RED
   timeout is 5000ms, interval is 1000ms
Via 1.0.0.2, Ethernet32/1, label stack: [900000]
   Reply from 1.0.10.2: seq=1, time=18ms, success: egress ok
Via 1.0.0.2, Ethernet32/1, label stack: [900000]
   Reply from 1.0.10.2: seq=2, time=19ms, success: egress ok
Via 1.0.0.2, Ethernet32/1, label stack: [900000]
   Reply from 1.0.10.2: seq=3, time=20ms, success: egress ok

--- Segment-Routing target fec 1.0.12.1/32, algorithm RED : lspping statistics ---
Via 1.0.0.2, Ethernet32/1, label stack: [900000]
   3 packets transmitted, 3 received, 0% packet loss, time 2174ms
   3 received from 1.0.10.2, rtt min/max/avg 18/20/19 ms
```

If parallel paths exist to the destination, the ping command will select one path. To verify all paths, use the traceroute command described next.

MPLS Traceroute

Use the traceroute command to verify all paths of a FlexAlgo tunnel:

```
#traceroute mpls segment-routing  <destination prefix> [algorithm <FlexAlgo name>]
```

For each path, the traceroute output shows downstream information of each intermediate node. From a successful traceroute, we should see ***"success: label switched"*** from all intermediate nodes and ***"success: egress ok"*** from the end node.

Example output:

```
#traceroute mpls segment-routing ip 1.0.12.1/32 algorithm RED
LSP traceroute to 1.0.12.1/32, algorithm RED
  via 1.0.0.2, label stack: [900000]
  1  1.0.0.2           MTU 1500  5ms       success: label switched
     downstream information (DSMAP) 1:
        interface address: 1.0.4.2
        IP address: 1.0.4.2
        label stack: [900000]
  2  1.0.4.2           MTU 1500  13ms      success: label switched
     downstream information (DSMAP) 1:
        interface address: 1.0.10.2
        IP address: 1.0.10.2
        label stack: [implicit-null]
  3  1.0.10.2          21ms      success: egress ok
  via 1.0.1.2, label stack: [900000]
  1  1.0.1.2           MTU 1500  5ms       success: label switched
     downstream information (DSMAP) 1:
        interface address: 1.0.6.2
        IP address: 1.0.6.2
        label stack: [900000]
  2  1.0.6.2           MTU 1500  14ms      success: label switched
     downstream information (DSMAP) 1:
        interface address: 1.0.11.2
        IP address: 1.0.11.2
        label stack: [implicit-null]
  3  1.0.11.2          20ms      success: egress ok
```

This example shows two paths to the destination ***1.0.12.1/32*** over the FlexAlgo tunnel ***RED : via 1.0.0.2*** and ***via 1.0.1.2*** .

Tracing

Disclaimer: In some cases, enabling tracing can seriously impact the performance of the switch. Please use it cautiously and seek advice from an Arista representative before enabling this in any production environment.

We recommend the following tracing settings:

```
trace Isis setting Rib::Isis/*cf,Rib::Mio/*cf,Rib::Isis-<instance>::Lsp/*cf,Rib::Isis
-Amun::Database/*cf,Rib::Isis-<instance>::Normal/*cf,Rib::Isis-<instance>::Debug/*cf,
Rib::Isis-<instance>::Adjacency/*cf,Rib::Isis-<instance>::Sr/*cf,SegmentRoutingImpl/*
cf,TopoDbImpl/*cf
trace Cspf setting CspfRoot/*cf,CspfVrfRoot/*cf,CspfAgent/*cf,TopoDb/*cf
```

In this command, ***“<instance>”*** is the name of the IS-IS instance.

Limitations

- FlexAlgo is only supported in multi-agent mode.
- There is no interaction between FlexAlgo at different levels or in different instances.
- Active measurement of link delay (e.g., TWAMP) is not supported.
- Interaction with LDP is not supported.
- Using PBR for traffic marshaling is not supported.
- Interaction with SR-TE is not supported.
- Prefix SIDs are not supported under FlexAlgo.
- Proxy node SIDs are not supported.
- Extended administrative groups are not supported.
- IP forwarding plane FlexAlgo is not supported.
- FlexAlgo is only supported in the default VRF.
- The implementation generates only a single Router Capability TLV, with a capacity of 255 bytes. All advertised FlexAlgo definitions must fit in this TLV. Typically, this is room for about 8 algorithm definitions.

Resources

- Segment Routing Traffic Engineering Policy (SR-TE) TOI
- IS-IS Traffic Engineering TOI
- Support for TI-LFA FRR using IS-IS Segment Routing TOI

- IGP Flexible Algorithm (Internet-Draft)
- IS-IS Application-Specific Link Attributes (Internet-Draft)
- Extended Administrative Groups TOI