Enabling Flexible Algorithm in IP Networks

- IGP Flexible Algorithm in IP Networks, on page 1
- Flexible Algorithm Configuration, on page 4
- Associating the IP Address to Flexible Algorithm, on page 5
- Example: Configuring IS-IS IP Flexible Algorithm, on page 6
- Verifying IP Flexible Algorithm, on page 7
- Protecting Flexible Algorithm IP Prefixes, on page 9
- Example: Enabling Flexible Algorithm Protection, on page 9
- Flexible-Algorithm Redistribution in IP Networks, on page 15

IGP Flexible Algorithm in IP Networks

Table 1: Feature History Table

Feature Name	Release Information	Feature Description

IGP IP Flexible Algorithm for IS-IS Protocol	Release 7.6.1	With IS-IS protocol extensions supporting Interior Gateway Protocol (IGP) Flexible Algorithm (Flex-Algorithm) on the IP data plane, you can now use the Algorithm to calculate IGP paths in an IP network without running Segment Routing. The IGP Flex-Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraints. Earlier, you could calculate IGP only using the Shortest Path First (SPF), which meant that you didn't have any choice except to use the default IGPpath calculated based on a native IGP metric. The following command is introduced: • **data-plane ip** The following commands are modified: • **ipv4 address** • **ipv6 address** • **show isis topology**

Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. The Flexible Algorithm allows Link State IGPs (OSPF, ISIS) to compute paths using various constraints. Each Flexible Algorithm represents the triplet (Calculation-Type, Metric-Type, Constraints) which all routers in the area consistently elect using the defined selection algorithm.

In IP networks, the router uses IGP Flexible Algorithm by computing the paths to the IPv4 address [RFC0791] and IPv6 address [RFCS8200].

Each interface may be associated with one or more IP addresses, and each IP address may be associated with one Flexible Algorithm.

- Packets sent to an address that is associated with a Flexible Algorithm follow the constraint-based path as calculated by the Flexible Algorithm.
- Packet sent to an address, that is *not* associated with a Flexible Algorithm, follow the IGP least-cost path to the egress node.

IGP IP Flexible Algorithm allows forwarding of the voice traffic and data traffic over different paths in the IP network. It can, for example, provide a low latency path for the voice traffic.

Note This feature is supported on Cisco Network Convergence System 5700 Series routers and routers with the Cisco NC57 line cards operating in native mode.

This document describes the IS-IS protocol extensions to support IGP Flexible Algorithm in IP networks.

Prerequisites for IP Flexible Algorithm

Configure the **data-plane ip** command under IGP flex-algo sub-mode to enable participation of the router with the IP Flexible Algorithm.

Flexible Algorithm Definition

The set consisting of (a) calculation-type, (b) metric-type and (c) a set of constraints is referred to as a Flexible-Algorithm Definition.

Flexible-Algorithm is a numeric identifier in the range 128-255 that is associated via provisioning with the Flexible-Algorithm Definition.

To guarantee the loop free forwarding for paths computed for a particular Flex-Algorithm, all routers that are configured to participate in a particular Flex-Algorithm, and in the same Flex-Algorithm definition advertisement scope must agree on the definition of the Flex-Algorithm as in ietf-lsr-flex-algo.

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 agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints
- Exclude SRLG constraints

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

IP Flexible Algorithm Prefix Advertisement

The IPv4 and IPv6 Algorithm Prefix Reachability TLVs defined in draft-ietf-lsr-ip-flexalgo are used to advertise prefix reachability associated with a Flexible Algorithm.

IP Flexible Algorithm Participation

Each application using the Flexible Algorithm must use its own participation signaling. IP Flexible Algorithm uses ISIS IP Algorithm Sub-TLV as specified in draft-ietf-lsr-ip-flexalgo.

Computing IP Flexible Algorithm Paths

Each Flexible Algorithm maintains a separate set of paths—One set per each data-plane.

The IP Flexible Algorithm computation uses only the IP Flex-Algorithm prefixes that are advertised in IPv4 and IPv6 Algorithm Prefix Reachability TLV.

Note The performance impact is proportional to the number of IP Flexible Algorithms in the participating router. Routers using the algorithm may use additional CPU cycles to:

- Process new TLVs.
- Calculate the primary and backup paths for the IP Flex-Algorithm prefixes.
- Advertise the IPv4 or IPv6 Algorithm Prefix Reachability TLVs.

IP Flexible Algorithm Forwarding

IP Flexible Algorithm uses the base IPv4 and IPv6 packets for forwarding.

IP Flexible Algorithm prefixes are advertised in IGPs. The forwarding plane installs the IP Flex-Algorithm prefixes advertised by the receiving routers participating in the associated topology and algorithm.

The IP Flex-Algorithm prefixes can be protected by local LFA. When the prefix is associated with an algorithm, the LFA paths to such a prefix are calculated using the Flexible Algorithm in the associated topology. Thus, ensuring that the algorithm follows the same constraints as the calculation of the primary paths.

The IP Flex-Algorithm prefixes can be protected by TI-LFA. For more information on protecting the IP Flex-Algorithm prefixes, see Protecting Flexible Algorithm IP Prefixes, on page 9.

Flexible Algorithm Configuration

The following workflow enables IP Flexible Algorithm:

1. Configure the IP data-plane to participate in the Flexible Algorithm. See Enabling the Flexible Algorithm Participation, on page 4.
2. Define the parameters for FAD. See Flexible Algorithm Definitions, on page 5.
3. Advertise the definition. See Flexible Algorithm Advertisement, on page 5.
4. Define the admin groups. See Configuring Affinity, on page 5.
5. Verify the Flexible Algorithm configuration.
6. Associate the IP address of the interface to the Flexible Algorithm. See Associating the IP Address to Flexible Algorithm, on page 5.

Enabling the Flexible Algorithm Participation

The following command in IS-IS flex-algo submode configuration enables Flexible Algorithm participation on the native IP data plane:

```
router isis instance
flex-algo algo
data-plane ip
```

Note Segment Routing is the default data-plane. To use the IP data-plane, you must enable the **data-plane ip** command.

Flexible Algorithm Definitions

The following commands configure the Flexible Algorithm definition under the flex-algo submode configuration:

- **router isis** *instance*
  flex-algo *algo*
  metric-type {**delay** | **te**}
- **affinity** {**include-any** | **include-all** | **exclude-any**} *name1, name2, …*

  name—Name of the affinity map.
- **priority** *priority value*

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

Flexible Algorithm Advertisement

The following command enables advertisement of the Flexible Algorithm definition in IS-IS:

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

Configuring Affinity

The following command defines the affinity-map. Affinity-map associates the name with the particular Bit positions in the Extended Admin Group bitmask.

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

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

The following command associates the Flexible Algorithm-specific affinities with an interface:

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

name—Name of the affinity-map.

Associating the IP Address to Flexible Algorithm

To associate the Flexible Algorithm with the interface global IPv4 or IPv6 address, use the following commands:

- **ipv4 address** *prefix-length* [**secondary** | **algorithm** *algo-no*]
- **ipv6 address** *prefix-length* **algorithm** *algo-no*]

Note If you do not associate an algorithm, the default algorithm value (0) is associated.

The following example shows how to associate IPv4 address with a Flexible Algorithm for loopback interface 1:

```
Router(config)# interface loopback 1
Router(config-if#) ipv4 address 1.1.1.1/32
Router(config-if) #ipv4 address 1.1.1.1/32 algorithm 128 ----------> adds the algo value
128
Router(config-if)#commit
```

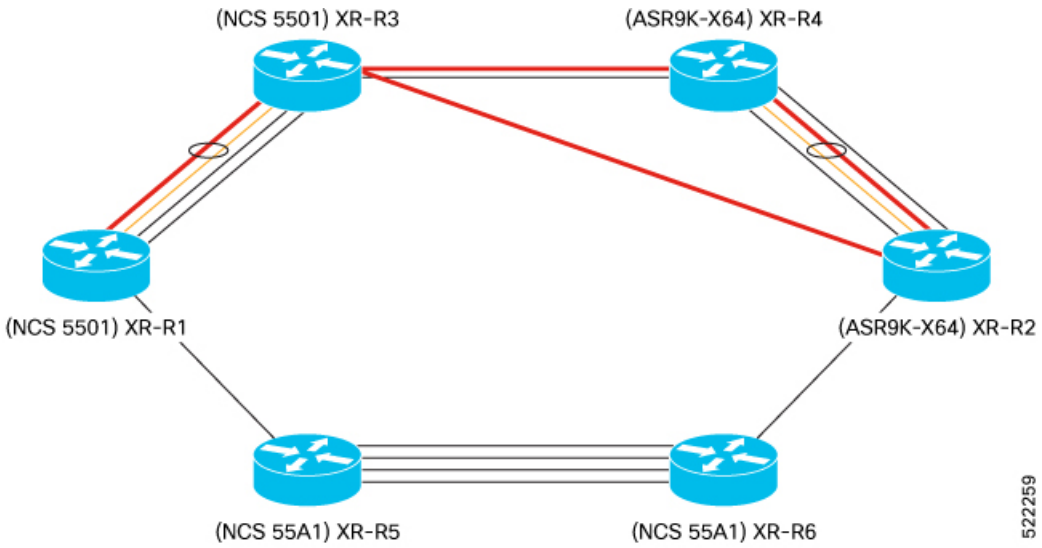
The following example associates the IPv6 address with a Flexible Algorithm for loopback interface 0:

```
Router(config)# interface loopback 0
Router(config-if)# ipv6 address 1::1/64 ----------------> add ipv6 add without algo
Router(config-if)# ipv6 address 1::1/64 algorithm 128 --------> add algo value 128
Router(config-if)#commit
```

Example: Configuring IS-IS IP Flexible Algorithm

The following figure is a six-node topology from R1 to R6 with configuration.

- Node R2 defines the FAD.
- IP Flexible Algorithm on node R2 ► R4 ► R3 ►R1.



This example shows the Flexible Algorithm 151 associated to IPv4 address at Loopback interface 51.

```
RP/0/RSP0/CPU0:Router# show run router isis
router isis 1
 is-type level-2-only
 net 49.0000.0000.0000.0012.00
 affinity-map RED bit-position 4
flex-algo 151
  data-plane ip
  metric-type delay
```

```
  advertise-definition
  affinity include-any RED
 !
!
 interface Loopback51
  passive
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
```

This example shows the IP address association with the Flexible Algorithm on Loopback interface 51.

```
RP/0/RSP0/CPU0:Router# show running interface Loopback 51
Mon Nov 15 14:02:09.832 IST
interface Loopback51
 ipv4 address 2.2.2.51 255.255.255.255 algorithm 151
 ipv6 address 2:2:2::51/128 algorithm 151
!
RP/0/RSP0/CPU0:Router# show running interface HundredGigE 0/1/0/0.301
Mon Nov 15 14:02:38.800 IST
interface HundredGigE0/1/0/0.301
 ipv4 address 200.3.2.1 255.255.255.0 secondary algorithm 151
 ipv6 address 200::3:2:1/112 algorithm 151
 encapsulation dot1q 301
```

Verifying IP Flexible Algorithm

Use the **show isis flex-algo** *algo-no* to verify data-plane used with Flexible Algorithm configuration.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 151
Mon Nov 15 15:54:06.291 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 151:

  Level-2:
    Definition Priority: 128
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: No
  Data Plane IP: Yes
```

This example verifies the IP Algo Prefix Advertisements in ISIS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime/Rcvd  ATT/P/OL
```

```
RouterR2.00-00          0x0000004c   0xd97e        773  /1200         0/0/0
  Area Address:   49.0000
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.31/32 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.51/32 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.1.0/24 D:0 Metric: 10 Algorithm: 131

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.2.0/24 D:0 Metric: 10 Algorithm: 151

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::31/128 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::51/128 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  NLPID:          0xcc
  NLPID:          0x8e
  IP Address:     2.2.2.2
  Router ID:      2.2.2.2
  Metric: 0          IP-Extended 2.2.2.2/32
    Prefix-SID Index: 2, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 102, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1 E:0 A:0
    Source Router ID: 2.2.2.2
  Metric: 0          MT (IPv6 Unicast) IPv6 2:2:2::2/128
    Prefix-SID Index: 12, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 112, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
```

This example verifies Flexible Algorithm 151 is present in only the topology—IP data-plane

```
RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 151 data-plane segment-routing
Mon Nov 15 16:20:23.553 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             **
RouterR3             **
RouterR4             **
RouterR5             **
RouterR6             **

RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 151 data-plane ip
Mon Nov 15 16:20:30.097 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR3             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             **        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             **        RouterR5             Te0/0/0/14.2    *PtoP*
```

These examples verifies the flexible Algorithm 151 IP data-plane IP route.

```
RP/0/RP0/CPU0:RouterR3# show isis route flex-algo 151 2.2.2.51/32 detail
Mon Nov 15 16:01:50.553 IST

L2 2.2.2.51/32 [32/115] Label: None, medium priority
   Installed Nov 15 12:57:39.377 for 03:04:11
     via 32.1.15.2, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight: 0
```

```
     src RouterR2.00-00, 2.2.2.2

RP/0/RP0/CPU0:RouterR3# show isis fast-reroute flex-algo 151 2.2.2.51/32 detail
Mon Nov 15 16:02:04.619 IST

L2 2.2.2.51/32 [32/115] Label: None, medium priority
   Installed Nov 15 12:57:39.377 for 03:04:25
     via 32.1.15.2, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight: 0
       Backup path: LFA, via 32.1.13.2, Bundle-Ether1, RouterR2, SRGB Base: 17000, Weight:
 0, Metric: 40
       P: No, TM: 40, LC: No, NP: Yes, D: Yes, SRLG: No
     src RouterR2.00-00, 2.2.2.2

RP/0/RP0/CPU0:RouterR3# show isis ipv6 route flex-algo 151 2:2:2::51/128 detail
Mon Nov 15 16:02:50.749 IST

L2 2:2:2::51/128 [32/115] Label: None, medium priority
   Installed Nov 15 15:45:17.416 for 00:17:33
     via fe80::28a:96ff:fee7:f400, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight:
0
     src RouterR2.00-00, 2:2:2::2

RP/0/RP0/CPU0:RouterR3# show isis ipv6 fast-reroute flex-algo 151 2:2:2::51/128 detail
Mon Nov 15 16:03:02.722 IST

L2 2:2:2::51/128 [32/115] Label: None, medium priority
   Installed Nov 15 15:45:17.416 for 00:17:45
     via fe80::28a:96ff:fee7:f400, TenGigE0/0/0/14.2, RouterR4, SRGB Base: 17000, Weight:
0
       Backup path: LFA, via fe80::2bc:60ff:fe04:74dc, Bundle-Ether1, RouterR2, SRGB Base:
 17000, Weight: 0, Metric: 40
       P: No, TM: 40, LC: No, NP: Yes, D: Yes, SRLG: No
     src RouterR2.00-00, 2:2:2::2
```

Protecting Flexible Algorithm IP Prefixes

Topology-Independent Loop-Free Alternate (TI-LFA) and Microloop avoidance features provide protection to link, node, and Shared Risk Link Groups (SRLG) using Segment Routing.

The IP Flexible Algorithm does not support TI-LFA and Microloop avoidance on its own. However, you can protect the IP Algorithm prefixes in the IP network by enabling Segment Routing in the network. With Segment Routing Flexible Algorithm, you can also protect the IP Flexible Algorithm traffic on its backup path.

Following conditions are required for TI-LFA protection and/or Microloop avoidance of the IP Flexible Algorithm traffic:

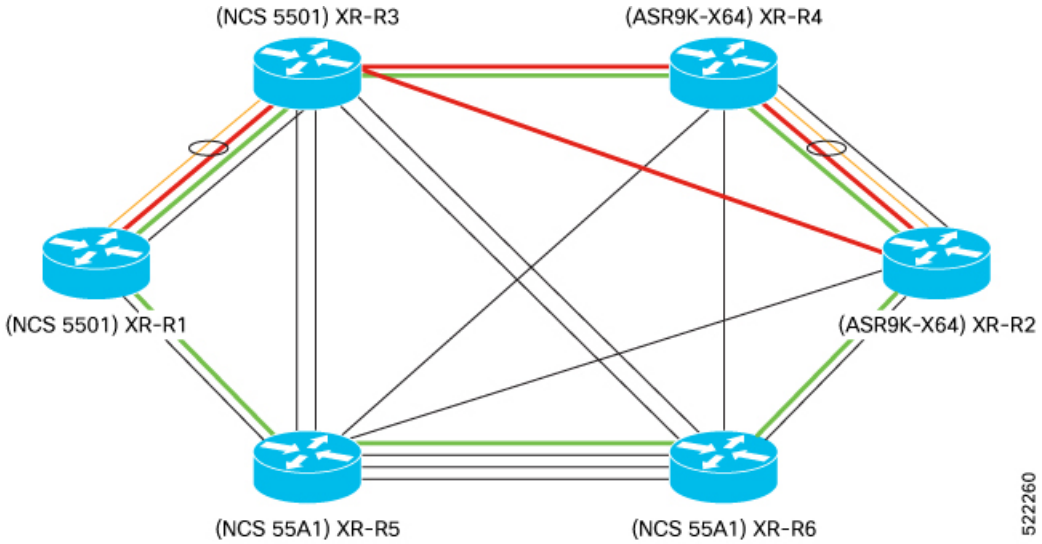
- Both Segment Routing and IP Flexible Algorithms data-planes must be enabled for a particular Flexible Algorithm X in an IS-IS area.
- The exact same set of reachable routers in the IS-IS area is participating in the Segment Routing and IP data-planes for Flexible Algorithm X.

Example: Enabling Flexible Algorithm Protection

The following figure is a six-node topology from R1 to R6 with configurations.

- Node R2 is winning the FAD.

- IP Flexible Algorithm with Flexible Algorithm 151 on nodes R2 ►R4 ► R3 ►R1.
- Segment Routing Flexible Algorithm with Flexible Algorithm 131 on nodes R2 ►R6 ► R5► R1 ► R3 ► R4.
- Both Segment Routing and IP Algorithm with Flexible Algorithm 131 on node R1 ► R3 ►R4 ►R2.



This example shows the Flexible Algorithm 131 and 151 configuration on R2.

```
RP/0/RSP0/CPU0:RouterR2# show run router isis
router isis 1
 is-type level-2-only
 net 49.0000.0000.0000.0012.00
 affinity-map GREEN bit-position 4
flex-algo 131
  data-plane segment-routing ip
  metric-type delay
  advertise-definition
  affinity include-any GREEN
 !
 flex-algo 151
  data-plane ip
  metric-type delay
  advertise-definition
  affinity include-any RED
 !
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls sr-prefer
  segment-routing mpls unlabeled protection route-policy ip_fa
!
 address-family ipv6 unicast
  metric-style wide
  segment-routing mpls sr-prefer
  segment-routing mpls unlabeled protection route-policy ip_fa

interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid index 2
   prefix-sid algorithm 131 index 102
```

```
 !
 address-family ipv6 unicast
  prefix-sid index 12
  prefix-sid algorithm 131 index 112
 !
!

interface Loopback31
 passive
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
 !
!
interface Loopback32
 passive
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
 !
```

This example shows the Loopback interfaces 31 and 51 associated to Flexible Algorithms 131 and 151 respectively.

```
RP/0/RSP0/CPU0:RouterR2# show run int Loopback 31
Mon Nov 15 14:02:03.386 IST
interface Loopback31
 ipv4 address 2.2.2.31 255.255.255.255 algorithm 131
 ipv6 address 2:2:2::31/128 algorithm 131
!

RP/0/RSP0/CPU0:Router# show run int Loopback 51
Mon Nov 15 14:02:09.832 IST
interface Loopback51
 ipv4 address 2.2.2.51 255.255.255.255 algorithm 151
 ipv6 address 2:2:2::51/128 algorithm 151
!
```

This example shows the 100-Gigabit Ethernet interface associated to the Flexible Algorithms 131 and 151.

```
RP/0/RSP0/CPU0:Router# show run int HundredGigE 0/1/0/0.301
Mon Nov 15 14:02:38.800 IST
interface HundredGigE0/1/0/0.301
 ipv4 address 200.3.1.1 255.255.255.0 algorithm 131
 ipv4 address 200.3.2.1 255.255.255.0 secondary algorithm 151
 ipv6 address 200::3:1:1/112 algorithm 131
 ipv6 address 200::3:2:1/112 algorithm 151
 encapsulation dot1q 301
```

Verification Examples

Use the **show isis flex-algo** command to verify Flexible Algorithm configurations.

This example verifies Flexible Algorithm 131 configuration on both Segment Routing and IP data-planes.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 131
Mon Nov 15 15:54:01.104 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 131:

  Level-2:
    Definition Priority: 128
```

```
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: Yes
  Data Plane IP: Yes
```

This example verifies the Flexible Algorithm 151 configuration on the IP data-plane.

```
RP/0/RSP0/CPU0:RouterR2# show isis flex-algo 151
Mon Nov 15 15:54:06.291 IST

IS-IS 1 Flex-Algo Database

Flex-Algo 151:

  Level-2:
    Definition Priority: 128
    Definition Source: Router.00, (Local)
    Definition Equal to Local: Yes
    Definition Metric Type: Delay
    Definition Flex-Algo Prefix Metric: No
    Exclude Any Affinity Bit Positions:
    Include Any Affinity Bit Positions:  4
    Include All Affinity Bit Positions:
    Exclude SRLGs:
    Disabled: No

  Local Priority: 128
  FRR Disabled: No
  Microloop Avoidance Disabled: No

  Data Plane Segemnt Routing: No
  Data Plane IP: Yes
```

This example verifies the Loopback interface 32 configuration.

```
RP/0/RSP0/CPU0:RouterR2#show isis interface Loopback 31          VERIFY ALGO AND IP IN ISIS
 INTERFACE
Mon Nov 15 16:42:30.819 IST

Loopback31                 Enabled
  Adjacency Formation:     Disabled (Passive in IS-IS cfg)
  Prefix Advertisement:    Enabled
 !

  IPv4 Address Family:     Enabled
    Protocol State:        Up
    Forwarding Address(es): Unknown (Intf passive in IS-IS cfg)
    Global Prefix(es):     2.2.2.31/32 (131)
  IPv6 Address Family:     Enabled
    Protocol State:        Up
    Forwarding Address(es): Unknown (Intf passive in IS-IS cfg)
    Global Prefix(es):     2:2:2::31/128 (131)
!
```

This example verifies the IP Algo Prefix Advertisements in ISIS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime/Rcvd  ATT/P/OL
RouterR2.00-00          0x0000004c   0xd97e        773  /1200         0/0/0
  Area Address:   49.0000
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.31/32 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 2.2.2.51/32 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
    Source Router ID: 2.2.2.2
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.1.0/24 D:0 Metric: 10 Algorithm: 131

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 200.3.2.0/24 D:0 Metric: 10 Algorithm: 151

    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::31/128 D:0 Metric: 0 Algorithm: 131
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 2:2:2::51/128 D:0 Metric: 0 Algorithm: 151
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  NLPID:          0xcc
  NLPID:          0x8e
  IP Address:     2.2.2.2
  Router ID:      2.2.2.2
  Metric: 0          IP-Extended 2.2.2.2/32
    Prefix-SID Index: 2, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 102, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1 E:0 A:0
    Source Router ID: 2.2.2.2
  Metric: 0          MT (IPv6 Unicast) IPv6 2:2:2::2/128
    Prefix-SID Index: 12, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 112, Algorithm:131, R:0 N:1 P:0 E:0 V:0 L:0
```

This example verifies the Flexible Algorithm participation in Segment Routing and IP data-plane in IS-IS database on R2.

```
RP/0/RP0/CPU0:RouterR1# show isis database detail verbose RouterR2
Mon Nov 15 16:05:22.188 IST

Router Cap:     2.2.2.2 D:0 S:0
    Segment Routing: I:1 V:1, SRGB Base: 17000 Range: 7000
    SR Local Block: Base: 15000 Range: 1000
    Node Maximum SID Depth:
      Label Imposition: 10
    SR Algorithm:
      Algorithm: 0
      Algorithm: 131
    IP Algorithm:
      Algorithm: 131
      Algorithm: 151
    Flex-Algo Definition:
      Algorithm: 131 Metric-Type: 1 Alg-type: 0 Priority: 128
      Flex-Algo Include-Any Ext Admin Group:
        0x00000010
    Flex-Algo Definition:
      Algorithm: 151 Metric-Type: 1 Alg-type: 0 Priority: 128
      Flex-Algo Include-Any Ext Admin Group:
        0x00000010
```

This exanple verifies Flexible Algorithm 131 is present in both topologies—Segment Routing and IP data-plane

```
RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 131 data-plane segment-routing
Mon Nov 15 16:20:05.271 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             12        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             22        RouterR5             Te0/0/0/14.2    *PtoP*

RP/0/RP0/CPU0:RouterR1# show isis topology flex-algo 131 data-plane ip
Mon Nov 15 16:20:14.015 IST

IS-IS 1 paths to IPv4 Unicast (Level-2) routers
System Id          Metric    Next-Hop           Interface       SNPA
RouterR1             --
RouterR2             32        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR3             20        RouterR3             BE1             *PtoP*
RouterR4             30        RouterR3             BE1             *PtoP*
RouterR5             12        RouterR5             Te0/0/0/14.2    *PtoP*
RouterR6             22        RouterR5             Te0/0/0/14.2    *PtoP*
```

This example shows Segment Routing Flexible Algorithm 131 prefix with Prefix SIDs.

```
RP/0/RSP0/CPU0:RouterR2# show isis route flex-algo 131 1.1.1.1/32 detail            SR FA
131 PREFIX WITH PREFIX-SID
Mon Nov 15 15:32:53.170 IST

L2 1.1.1.1/32 [31/115] Label: 17101, medium priority
   Installed Nov 15 13:46:40.902 for 01:46:13
     via 32.1.24.2, TenGigE0/1/0/3/6, Label: 17101, RouterR4, SRGB Base: 17000, Weight: 0
     src RouterR1.00-00, 1.1.1.1, prefix-SID index 101, R:0 N:1 P:0 E:0 V:0 L:0, Alg:131
RP/0/RSP0/CPU0:Router#
RP/0/RSP0/CPU0:Router2# show isis fast-reroute flex-algo 131 1.1.1.1/32 detail
Mon Nov 15 15:32:58.794 IST

L2 1.1.1.1/32 [31/115] Label: 17101, medium priority
   Installed Nov 15 13:46:40.902 for 01:46:19
     via 32.1.24.2, TenGigE0/1/0/3/6, Label: 17101, RouterR4, SRGB Base: 17000, Weight: 0
       Backup path: TI-LFA (link), via 32.1.26.2, TenGigE0/1/0/3/5.2 tb5-r6, SRGB Base:
17000, Weight: 0, Metric: 120
         P node: RouterR6.00 [6.6.6.6], Label: ImpNull
         Q node: RouterR5.00 [5.5.5.5], Label: 25009
         Prefix label: 17101
         Backup-src: RouterR1.00
       P: No, TM: 120, LC: No, NP: No, D: No, SRLG: No
     src RouterR1.00-00, 1.1.1.1, prefix-SID index 101, R:0 N:1 P:0 E:0 V:0 L:0, Alg:131
```

This example shows IP Flexible Algorithm 131 prefix with IP Prefixes.

```
RP/0/RSP0/CPU0:RouterR2#show isis route flex-algo 131 1.1.1.31/32 detail
Mon Nov 15 15:33:15.970 IST

L2 1.1.1.31/32 [31/115] Label: None, medium priority
   Installed Nov 15 13:46:34.923 for 01:46:42
     via 32.1.24.2, TenGigE0/1/0/3/6, tb5-r4, SRGB Base: 17000, Weight: 0
     src RouterR1.00-00, 1.1.1.1


!!TI-LFA backup is seen only if both IP AND SR TOPOLOGY ARE CONGRUENT
```

```
RP/0/RSP0/CPU0:RouterR2#show isis fast-reroute flex-algo 131 1.1.1.31/32 detail
Mon Nov 15 15:33:27.404 IST

L2 1.1.1.31/32 [31/115] Label: None, medium priority
   Installed Nov 15 13:46:34.923 for 01:46:54
     via 32.1.24.2, TenGigE0/1/0/3/6, tb5-r4, SRGB Base: 17000, Weight: 0
       Backup path: TI-LFA (link), via 32.1.26.2, TenGigE0/1/0/3/5.2 tb5-r6, SRGB Base:
17000, Weight: 0, Metric: 120
         P node: tb5-r6.00 [6.6.6.6], Label: ImpNull
         Q node: tb5-r5.00 [5.5.5.5], Label: 25009
         Prefix label: None
         Backup-src: RouterR1.00
       P: No, TM: 120, LC: No, NP: No, D: No, SRLG: No
     src RouterR1.00-00, 1.1.1.1
```

Flexible-Algorithm Redistribution in IP Networks

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Flexible-Algorithm Redistribution in IP Networks	Release 7.9.1	You can now select or filter the prefix-matching algorithm number during route redistribution, so that only the Flex-Algorithms that you configured for specific addresses are redistributed. By setting Flex-Algorithms for prefixes that are redistributed into IS-IS protocols, you can get better control of traffic that flows between networks running on different routing protocols. This feature introduces the **set algorithm** command.

Earlier, there was no way to redistribute routers into a particular Flex-Algorithm from another domain. Due to this, when you were running IP Flex-Algorithms in a domain and you were redistributing Flex-Algorithms prefixes from another domain, all prefixes would go to base algorithm (algorithm 0). For more information on Flex-Algorithm and the algorithm numbers, see *IGP Flexible Algorithm in IP Networks* in the *Routing Configuration Guide for Cisco NCS 5500 Series Routers*.

Starting from Release 7.9.1, you can now redistribute prefixes into a particular Flex-Algorithm between 128-255. This allows Link State IGPs (OSPF, ISIS) to compute paths using various constraints. You can have more control over the traffic that moves between networks using various routing protocols by configuring Flex-Algorithms for prefixes that are redistributed into IS-IS protocols.

This feature allows you to:

- Set an algorithm in the prefixes
- Match the prefixes based on the algorithm

Set an Algorithm

The set algorithm is added in the route policy mechanism. By using this **set algorithm** command, you can set the Flex-Algorithm for the prefixes that are redistributed into IS-IS protocols. The prefix advertisement after route redistribution can be altered via **set algorithm** in the routing policy.

Limitation:

- The Route Policy accepts algorithm number from 0 to 255.
- The IS-IS protocol only handles algorithm 0, 128-255.
- The algorithm number 1 through 127 in IS-IS protocol are misconfigured algorithms and treated as algorithm 0. Due to this, if you are redistributing Flex-Algorithms prefixes (1 through 127) from another domain, all prefixes would go to base algorithm (algorithm 0).

Use Set Algorithm in RPL

Configuration Example

1. Set the prefixes (that are redistributed into IS-IS protocol) into Flex-Algorithm 128 using the route-policy.

```
Router(config)#prefix-set PFX_ALGO128
Router(config-pfx)#44.44.44.128/32,
Router(config-pfx)#44:44:44::128/128
Router(config-pfx)#end-set
```

2. Define the route-policy using set algorithm to set Flex-Algorithm 128 for prefix-set *PFX_ALGO128*.

```
Router(config)#route-policy BGP_TO_ISIS
Router(config-rpl)# if destination in PFX_ALGO128 then
Router(config-rpl-if)# set tag 200
Router(config-rpl-if)# set algorithm 128
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# drop
Router(config-rpl-else)# endif
Router(config-rpl)#end-policy
Router(config)#commit
```

3. Use the route-policy while redistributing Flex-Algorithm 128 into IS-IS protocol.

```
Router(config)#router isis 100
Router(config-isis)#address-family ipv4 unicast
Router(config-isis-af)#redistribute bgp 100 route-policy BGP_TO_ISIS metric-type
rib-metric-as-external
Router(config-isis-af)#exit
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#redistribute bgp 100 route-policy BGP_TO_ISIS metric-type
rib-metric-as-external
Router(config-isis-af)#commit
Router(config-isis-af)#end
```

Running Configuration

```
prefix-set PFX_ALGO128
  44.44.44.128/32,
  44:44:44::128/128
end-set
!
route-policy BGP_TO_ISIS
  if destination in PFX_ALGO128 then
    set tag 200
    set algorithm 128
    pass
```

```
  else
    drop
  endif
end-policy

router isis 100
 address-family ipv4 unicast
  metric-style wide
  redistribute bgp 100 route-policy BGP_TO_ISIS metric-type rib-metric-as-external
  segment-routing mpls
 !

router isis 100
 address-family ipv6 unicast
  metric-style wide
  redistribute bgp 100 route-policy BGP_TO_ISIS metric-type rib-metric-as-external
  segment-routing mpls
```

Verification

Run the **show isis database** command in IS-IS protocol database to check whether RPL is applied and redistributed prefixes has Flex-Algorithm 128 attached.

```
Router#show isis instance 100 database R4.00-01 verbose | begin 44.44.44.128/32
Tue Dec 13 09:12:41.395 UTC
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 2 Algorithm:
128
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
      algo 128 due to RPL with set algo.
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 129
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 130
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 131
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv4 Algo Prefix: MT (Standard (IPv4 Unicast)) 44.44.44.128/32 D:0 Metric: 10 Algorithm:
 132
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  Metric: 0          MT (IPv6 Unicast) IPv6-External 6:6:6:6::6/128
    Admin. Tag: 200
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  Metric: 10         MT (IPv6 Unicast) IPv6-External 7:7:7:7::7/128
    Admin. Tag: 200
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::132/128 D:0 Metric: 0 Algorithm: 132
    Prefix Attribute Flags: X:0 R:0 N:0 E:0 A:0
  IPv6 Algo Prefix: MT (IPv6 Unicast) 66:66:66::128/128 D:0 Metric: 0 Algorithm: 128
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::128/128 D:0 Metric: 2 Algorithm: 128
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
    Admin. Tag: 200
  IPv6 Algo Prefix: MT (IPv6 Unicast) 44:44:44::129/128 D:0 Metric: 10 Algorithm: 129
    Prefix Attribute Flags: X:1 R:0 N:0 E:0 A:0
```

```
    Admin. Tag: 200
```

Match an Algorithm

You can configure the match algorithms in RPL and use them to filter out based on algorithms during redistribution. This mechanism acts as a filter based on the algorithm number in the prefixes while redistributing.

Use Match Algorithm in RPL

You can use the match algorithm as a conditional expression to select or filter the prefix matching algorithm during route redistribution.

Configuration Example

1. Set a route-policy that is used to filter out Flex-Algorithm 128 and algorithm 0 prefixes.

```
Router(config)#route-policy ISIS_TO_BGP
Router(config-rpl)#  if algorithm is 128 or algorithm is 0 then
Router(config-rpl-if)# pass
Router(config-rpl-if)# else
Router(config-rpl-else)# drop
Router(config-rpl-else)# endif
Router(config-rpl)#end-policy
Router(config)#commit
```

2. Apply the route-policy to filter out Flex-Algorithm 128 and algorithm 0 prefixes while redistributing IS-IS protocol into BGP.

```
Router(config)#router bgp 100
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#  redistribute isis 100 route-policy ISIS_TO_BGP
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)# redistribute isis 100 route-policy ISIS_TO_BGP
Router(config-bgp-af)#commit
```

Running Configuration

```
route-policy ISIS_TO_BGP
  if algorithm is 128 or algorithm is 0 then
    pass
  else
    drop
  endif
end-policy
!

router bgp 100
 address-family ipv4 unicast
  redistribute isis 100 route-policy ISIS_TO_BGP
 !

router bgp 100
 address-family ipv6 unicast
  redistribute isis 100 route-policy ISIS_TO_BGP
 !
```

Verification

The **show isis route flex-algo <*algorithm number*>** command displays the routes available into Flex-Algorithm 128:

```
Router#show isis route flex-algo 128
Tue Dec 13 09:34:09.502 UTC
IS-IS 100 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: 8
L2 11.11.11.128/32 [20/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
L2 22.22.22.128/32 [10/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
L2 33.33.33.128/32 [30/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
C  44.44.44.128/32
     is directly connected, Loopback129
L2 55.55.55.128/32 [40/115]
     via 2.4.0.2, HundredGigE0/0/0/1, R2, SRGB Base: 16000, Weight: 0
```

The **show route *prefix* detail** command displays the prefix `11.11.11.128/32` prefix belongs to Flex-Algorithm 128.

```
Router#show route 11.11.11.128/32 detail
Tue Dec 13 09:35:16.681 UTC
Routing entry for 11.11.11.128/32
  Known via "isis 100", distance 115, metric 4 (algo 128), type level-2
  Installed Dec 13 07:56:46.972 for 01:38:29
  Routing Descriptor Blocks
    4.5.0.5, from 1.1.1.1, via HundredGigE0/0/0/2, Backup (Local-LFA)
      Route metric is 54
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:65       Path ref count:1
      NHID:0x3(Ref:29)
    2.4.0.2, from 1.1.1.1, via HundredGigE0/0/0/1, Protected
      Route metric is 4
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x2(Ref:32)
      Backup path id:65
  Route version is 0x2 (2)
  No local label
  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 170
  No advertising protos.
```

Use the **show bpg ipv4 unicast advertised summary** command to verify RPL filtering.

```
Router#show bgp ipv4 unicast advertised summary
Tue Dec 13 09:37:53.596 UTC
Network            Next Hop        From            Advertised to
1.1.1.1/32         4.6.0.4         Local           4.6.0.6
1.2.0.0/24         4.6.0.4         Local           4.6.0.6
1.3.0.0/24         4.6.0.4         Local           4.6.0.6
2.2.2.2/32         4.6.0.4         Local           4.6.0.6
2.4.0.0/24         4.6.0.4         Local           4.6.0.6
3.3.3.3/32         4.6.0.4         Local           4.6.0.6
3.5.0.0/24         4.6.0.4         Local           4.6.0.6
4.4.4.4/32         4.6.0.4         Local           4.6.0.6
4.5.0.0/24         4.6.0.4         Local           4.6.0.6
4.6.0.0/24         4.6.0.4         Local           4.6.0.6
5.5.5.5/32         4.6.0.4         Local           4.6.0.6
5.6.0.0/24         4.6.0.4         Local           4.6.0.6
11.11.11.128/32    4.6.0.4         Local           4.6.0.6
22.22.22.128/32    4.6.0.4         Local           4.6.0.6
33.33.33.128/32    4.6.0.4         Local           4.6.0.6
44.44.44.128/32    4.6.0.4         Local           4.6.0.6
55.55.55.128/32    4.6.0.4         Local           4.6.0.6
Processed 17 prefixes, 17 paths
```