Cisco IOS XRv 9000 Router Control Plane Specific Features

This chapter covers information about the Cisco XRv 9000 Control Plane Specific features.

BGP Optimal Route Reflector

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
BGP ORR 6PE	Release 7.3.1	This feature is introduced. If there is no path selectable as bestpath for a given ORR table, you can assign the default table's bestpath as ORR group's bestpath. This feature enables IPv6 label-unicast with IPv4 nexthop and fallback default path. New keyword introduced in this release: • **fallback-default-bestpath**

BGP-ORR (optimal route reflector) enables virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view.

BGP ORR calculates the best path by:

1. Running SPF multiple times in the context of its RR clients or RR clusters (set of RR clients)
2. Saving the result of different SPF runs in separate databases
3. Using these databases to manipulate BGP best path decision and thereby allowing BGP to use and announce best path that is optimal from the client's point of view

In an autonomus system, a BGP route reflector acts as a focal point and advertises routes to its peers (RR clients) along with the RR's computed best path. Since the best path advertised by the RR is computed from the RR's point of view, the RR's placement becomes an important deployment consideration.

With network function virtualization (NFV) becoming a dominant technology, service providers (SPs) are hosting virtual RR functionality in a cloud using servers. A vRR can run on a control plane device and can be placed anywhere in the topology or in a SP data center. Cisco IOS XRv 9000 Router can be implemented as vRR over a NFV platform in a SP data center. vRR allows SPs to scale memory and CPU usage of RR deployments significantly. Moving a RR out of its optimal placement requires vRRs to implement ORR functionality that calculates the best path from a RR client's point of view.

BGP ORR offers these benefits:

- calculates the bestpath from the point of view of a RR client.
- enables vRR to be placed anywhere in the topology or in a SP data center.
- allows SPs to scale memory and CPU usage of RR deployments.

Note Enabling the ORR feature increases the memory footprint of BGP and RIB. With increased number of vRR configured in the network, ORR adversely impacts convergence for BGP.

BGP ORR Configuration Methods

You can configure the BGP ORR as follows:

- BGP ORR without path advertisement policy.
- BGP ORR with path advertisement policy.

BGP ORR without Path Advertisement Policy

The vRR performs the following steps to calculate and advertise the BGP ORR best path to all the vRR sites and ORR groups:

1. The vRR calculates the BGP ORR best path for each ORR group from the global Address Family Indicator (AFI) policy.
2. The vRR advertises the calculated best path along with selected additional paths.

If you haven't assigned a policy to the ORR group, by default, the vRR calculates the best paths using the Global-AFI's default selection policy for that ORR group. For more information, refer to BGP ORR without Path Advertisement Policy Topology, on page 3.

BGP ORR with Path Advertisement Policy

The vRR performs the following steps to calculate and advertise the BGP ORR best and backup/additional paths to all the vRR sites and ORR groups:

1. You can configure an ORR group add path policy using the **optimal-route-reflection** command. For more information, refer to BGP Optimal Route Reflector with Path Advertisement Policy, on page 10.
2. The vRR calculates the best path for each ORR group based on the assigned policy.
3. The vRR advertises the best and backup/additional paths to PE routers within the ORR group that has addpath route-policy.

Note If required you can enable the vRR to advertise all the paths to the ORR groups using the *addpath-all policy* in the **optimal-route-reflection** command.

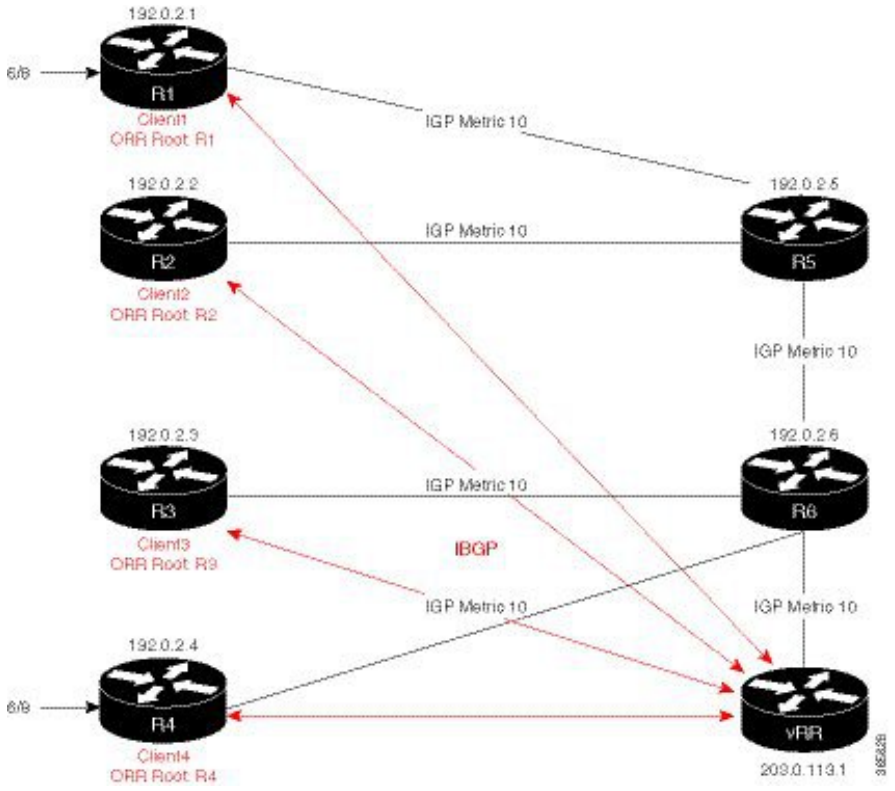
4. The vRR advertises the calculated best path along with all the known paths to all vRR sites within the Intra-Region Groups.
5. The vRR advertises only the client path and the necessary peering path to all the vRR sites within the Inter-Region Group.

BGP ORR without Path Advertisement Policy Topology

Consider a BGP ORR without path advertisement policy topology where:

- Router R1, R2, R3, R4, R5 and R6 are route reflector clients
- Router R1 and R4 advertise 6/8 prefix to vRR

Figure 1: BGP-ORR without Path Advertisement Policy Topology



vRR receives prefix 6/8 from R1 and R4. Without BGP ORR configured in the network, the vRR selects R4 as the closest exit point for RR clients R2, R3, R5, and R6, and reflects the 6/8 prefix learned from R4 to these RR clients R2, R3, R5, and R6. From the topology, it is evident that for R2 the best path is R1 and not R4. This is because the vRR calculates best path from the RR's point of view.

When the BGP ORR is configured in the network, the vRR calculates the shortest exit point in the network from R2's point of view and determines that R1 is the closest exit point to R2. vRR then reflects the 6/8 prefix learned from R1 to R2.

Restrictions and Limitations for BGP ORR without Path Advertisement Policy

The following are the restrictions and limitations for BGP ORR without path advertisement policy:

- If there are multiple paths added to a ORR group, the vRR chooses only one path among them to calculate the best path for a vRR site.
- The vRR calculates and advertises same type and set of additional paths to all the clients.

Configuring BGP ORR without Path Advertisement Policy

Perform the following steps to configure the BGP ORR without path advertisement policy.

Step 1 Configure an ORR globally under router BGP mode.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# optimal-route-reflection ipv4 foo 10.1.1.1 10.1.1.2 10.1.1.3
Router(config-bgp)# optimal-route-reflection ipv6 bar abcd::1 abcd::2 abcd::3
```

Step 2 Enable the ORR group under address-family mode.

Example:

```
Router(config-bgp-route)# address-family ipv4 unicast
Router(config-bgp-route-af)# optimal-route-reflection apply foo
Router(config-bgp-route-af)# exit
Router(config-bgp-route)# address-family ipv6 unicast
Router(config-bgp-route-af)# optimal-route-reflection apply bar
Router(config-bgp-route-af)# allocate-label {all | route-policy <>}
Router(config-bgp-route-af)# commit
```

Step 3 Specify a neighbor as an ORR client.

Example:

```
Router# configure
Router(config)# neighbor 2.2.2.2
Router(config-nbr)# remote-as 100
Router(config-nbr)# address-family ipv4 unicast
Router(config-nbr-af)# optimal-route-reflection foo
Router(config-nbr-af)# exit
Router(config-nbr)# address-family ipv6 label-unicast
Router(config-nbr-af)# optimal-route-reflection foo
Router(config-nbr-af)# commit
```

Step 4 Enable the selection of default table's bestpath in the absence of bestpath for an ORR group.

Example:

```
Router# configure
Router(config)# router bgp 65000
Router(config-bgp)# bgp router-id 10.1.1.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# optimal-route-reflection fallback-default-bestpath
```

Step 5 View the running configuration to verify the configuration that you have configured.

Example:

```
router bgp 100
 optimal-route-reflection ipv4 foo 10.1.1.1 10.1.1.2 10.1.1.3
 optimal-route-reflection ipv6 bar abcd::1 abcd::2 abcd::3
 address-family ipv4 unicast
  optimal-route-reflection apply foo
 address-family ipv6 unicast
  optimal-route-reflection apply foo
  allocate-label {all | route-policy <>}
 neighbor 2.2.2.2
  remote-as 100
  address-family ipv4 unicast
   optimal-route-reflection foo
  address-family ipv6 label-unicast
   optimal-route-reflection foo
```

Step 6 Perform the following steps to verify the BGP ORR configuration for IPv4:

a) Verify whether R2 received the best exit, execute the **show bgp <prefix>** command (from R2) in EXEC mode. In the above example, R1 and R4 advertise the 6/8 prefix; run the **show bgp 6.0.0.0/8** command:

Example:

```
Router# sh bgp 200.1.1.0/24 path-elements
BGP routing table entry for 200.1.1.0/24
Versions:
    Process     bRIB/RIB SendTblVer
    Speaker           84         84
        Flags: 0x043e3028+0x00000000;
Last Modified: Sep 7 20:21:38.000 for 00:11:45
Paths: (2 available, best #1)
Path-elements: 2
    Path ID: 1
        Gateway metric 0, Version 84
    Path: Nexthop 11.1.1.2, flags 0x4000000001440207
          Neighbor 11.1.1.2, Received Path ID 0
    Flags: 0x00000001
           status: valid
           path type: bestpath
           add-path action:
    Opaque: pelem=0xebf81d10
            net=0xabe435c, tblattr=0xa962f64 (ver 84)
            path=0xabf4450, path-tblattr=0xa962f64 (ver 84)
                       nobestpath-tblattr=0xa963234 (ver 0)
                        noaddpath-tblattr=0xa9631e4 (ver 0)
            bitfields=0xace9f64 (val=0x1a, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0xac145d8 (val=0x0, size=0)
            orr-ap-bitfields=0xac145d8 (val=0x2, size=0). <<<<<<<<<<<<<<
            net-next=0xebf81cd4, tblattr-prev=0xebf81cd4, tblattr-next=0xebf81c98
Path ID: 2
    Gateway metric 0, Version 107
    Path: Nexthop 192.168.0.10, flags 0x4000000000040007
          Neighbor 192.168.0.2, Received Path ID 2
    Flags: 0x00000042
           status: valid
           path type: backup
           add-path action: advertise
    Opaque: pelem=0x10a0bba8
            net=0x109cbb14, tblattr=0x10399c84 (ver 156)
            path=0x109ebcd4, path-tblattr=0x10399c84 (ver 156)
                        nobestpath-tblattr=0x10399fa4 (ver 0)
                            noaddpath-tblattr=0x10399f54 (ver 0)
            bitfields=0x10c00624 (val=0x10, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0x10a2b8cc (val=0x0, size=1)
            orr-ap-bitfields=0x10a2b8a8 (val=0x6, size=1) <<<<<<<<<<<<<<
            net-next=0x10a0b528, tblattr-prev=0x10a0bbe8, tblattr-next=0x0
```

The above show output states that the best path for R2 is through R1, whose IP address is 192.0.2.1 and the metric of the path is 20.

b) Execute the **show bgp** command from the vRR to determine the best path calculated for R2 by ORR. R2 has its own update-group because it has a different best path (or different policy configured) than those of other peers:

Example:

```
VRR# show bgp 6.0.0.0/8
Thu Apr 28 13:36:42.744 UTC
BGP routing table entry for 6.0.0.0/8
Versions:
Process bRIB/RIB SendTblVer
Speaker 13 13
```

```
Last Modified: Apr 28 13:36:26.909 for 00:00:15
Paths: (2 available, best #2)
Advertised to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.1 (metric 30) from 192.0.2.1 (192.0.2.1)
Origin incomplete, metric 0, localpref 100, valid, internal, add-path
Received Path ID 0, Local Path ID 2, version 13
Path #2: Received by speaker 0
Advertised to update-groups (with more than one peer):
0.2
ORR addpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.4 (metric 20) from 192.0.2.4 (192.0.2.4)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
Received Path ID 0, Local Path ID 1, version 13
```

Note Path #1 is advertised to update-group 0.1. R2 is in update-group 0.1.

c) Execute the **show bgp** command for update-group 0.1 verify whether R2 is in update-group 0.1.

Example:

```
VRR# show bgp update-group 0.1
Thu Apr 28 13:38:18.517 UTC

Update group for IPv4 Unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 65000
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): g1; Index: 0
4-byte AS capable
Non-labeled address-family capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 5, replicated: 5
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1
Neighbors in filter-group: 0.2(RT num: 0)
192.0.2.2
```

d) For further verification, check the contents of the table created on vRR as a result of configuring the g1 policy. From R2's point of view, the cost of reaching R1 is 20 and the cost of reaching R4 is 30. Therefore, the closest and best exit for R2 is through R1.

Example:

```
VRR# show orrspf database g1
Thu Apr 28 13:39:20.333 UTC

ORR policy: g1, IPv4, RIB tableid: 0xe0000011
Configured root: primary: 192.0.2.2, secondary: NULL, tertiary: NULL
Actual Root: 192.0.2.2, Root node: 2000.0100.1002.0000

Prefix Cost
203.0.113.1 30
192.0.2.1 20
192.0.2.2 0
192.0.2.3 30
192.0.2.4 30
192.0.2.5 10
192.0.2.6 20

Number of mapping entries: 8
```

Step 7 Perform the following steps to verify the BGP ORR configuration for IPv6 Provider Edge (6PE):

a) Verify the BGP ORR configuration using the **show bgp** command.

Example:

```
show bgp ipv6 labeled-unicast 1111::1/128
Tue Mar 2 10:25:00.748 PST
BGP routing table entry for 1111::1/128
Versions:
Process bRIB/RIB SendTblVer
Speaker 4 4
Last Modified: Mar 2 10:18:53.000 for 00:06:08
Paths: (3 available, best #3)
Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.168.0.3 (metric 75) from 192.168.0.3 (192.168.0.3)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, add-path, labeled-unicast
Received Path ID 0, Local Path ID 2, version 4
Path #2: Received by speaker 0
Not advertised to any peer
Local, (Received from a RR-client)
192.168.0.4 (metric 190) from 192.168.0.4 (192.168.0.4)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, labeled-unicast
Received Path ID 0, Local Path ID 0, version 0
Path #3: Received by speaker 0
Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
0.2
Local, (Received from a RR-client)
192.168.0.5 (metric 65) from 192.168.0.5 (192.168.0.5)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, labeled-unicast
Received Path ID 0, Local Path ID 1, version 3
```

b) Verify the ipv6 group configuration using the **show bgp** command.

Example:

```
show bgp ipv6 labeled-unicast update-group
Tue Mar 2 10:25:51.308 PST
```

```
Update group for IPv6 Labeled-unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 1
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): orr-grp-1; Index: 0
4-byte AS capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 1, replicated: 2
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1
Neighbors in filter-group: 0.2(RT num: 0)
192.168.0.2 192.168.0.4

Update group for IPv6 Labeled-unicast, index 0.2:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 1
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
4-byte AS capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 1, replicated: 4
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.1, Filter-Groups num:1
Neighbors in filter-group: 0.1(RT num: 0)
192.168.0.3 192.168.0.5
```

c) Verify the IPv6 unicast routes in the BGP configuration using the **show bgp** command.

Example:

```
show bgp ipv6 unicast orr-group all
Tue Mar 2 10:26:41.072 PST
Name Tableid Nbrcnt Index Root
orr-grp-1 0xe0000019 2 0 192.168.0.3
```

d) Verify the BGP speaker global ORR policy group.

Example:

The following show command displays the BGP speaker global ORR policy group table.

```
Router# show bgp orr-group global all
Wed Apr  8 16:46:29.929 PDT
Name            Policy-afi Global     Tableid          AFI-count  Root
orr-grp-3       IPv4       Yes        0xe0000014        1         1.1.2.1
orr-grp-2       Ipv6       Yes        0xe0800013        0         1::1
orr-grp-1       IPv4       Yes        0xe0000012        2         192.168.0.3
```

The following show command displays the details of a global ORR group entry for the given ORR name.

```
Router# show bgp orr-group global orr-grp-1
Wed Apr  8 16:46:51.596 PDT
  ORR Name : orr-grp-1
       policy afi : IPv4
global Defined : Yes
          tableid : 0xe0000012
           aficnt : 2
IPv4 unicast used : Yes
IPv6 unicast used : Yes
           root : 192.168.0.3
```

The following show command displays the BPM ORR policy group table:

```
Router# show bgp orr-group bpm all
Wed Apr  8 16:49:44.223 PDT
Name            Policy-afi Global     AFI-cnt    Nbr-af-cnt      Root
orr-grp-3       IPv4       Yes        1          0               1.1.2.1
orr-grp-2       IPv6       Yes        0          0               1::1
orr-grp-1       IPv4       Yes        2          4               192.168.0.3
```

The following show command displays the detail of a BPM ORR group entry for the given ORR name.

```
Router# show bgp orr-group bpm orr-grp-1
Wed Apr  8 16:50:02.437 PDT
    ORR Name : orr-grp-1
        v4 policy : Yes
   global Defined : Yes
        AFI count : 2
 total nbr af cnt : 4
IPv4 unicast used : Yes
IPv6 unicast used : Yes
  IPv4 nbr af cnt : 2
  IPv6 nbr af cnt : 2
           root : 192.168.0.3
```

BGP Optimal Route Reflector with Path Advertisement Policy

Table 2: Feature History Table

Feature Name	Release Information	Feature Description

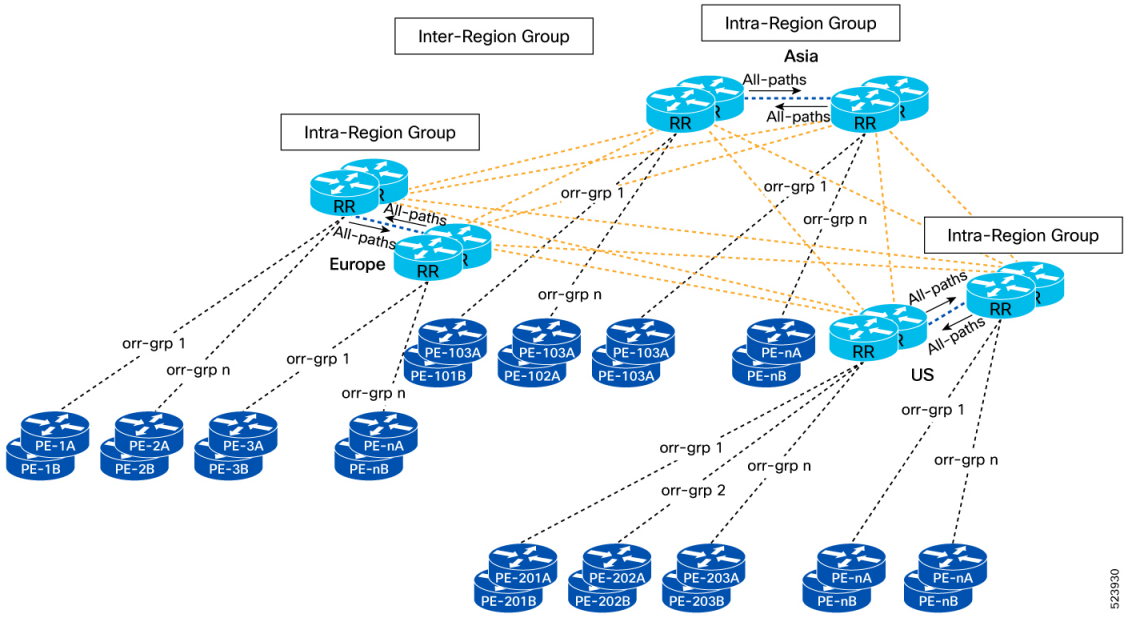
BGP Optimal Route Reflector with Path Advertisement Policy	Release 24.2.1	This feature allows you to assign an independent routing policy to an individual Optimal Route Reflector (ORR) group. The virtual route reflector (vRR) calculates the additional paths for that ORR group based on the assigned policy. This results in having a different best and backup path for each ORR group which is the optimal path from the client point of view. This feature introduces these changes: **CLI:** • The *route-policy* keyword is introduced in the **optimal-route-reflection** command. **YANG Data Model:** • `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg.yang` (see GitHub, YANG Data Models Navigator)

BGP-ORR (optimal route reflector) enables the virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view. For more information, refer to BGP Optimal Route Reflector, on page 1.

Traffic Flow Topology

Consider a topology where the network traffic flows from ORR groups, through vRR sites, through an Intra-Region Group, and finally through an Inter-Region Group. Refer to the figure Figure 2: Advertising BGP-ORR Best Path, on page 11.

Figure 2: Advertising BGP-ORR Best Path



The following are the definition of the hierarchical structure of network elements within a BGP ORR environment, as shown in the figure Figure 2: Advertising BGP-ORR Best Path, on page 11:

- ORR Groups: Multiple sets of routers and provider edge (PE) routers connect to form an ORR group. For example, orr-grp n.
- vRR Sites: Multiple ORR groups connect to form a vRR site.
- Intra-Region Group: Multiple vRR sites connect to form an Intra-Region Group. For example, Europe.
- Inter-Region Group: Multiple Intra-Region groups connect to form an Inter-Region Group.

This feature allows you to configure add-path selection policies for each vRR site and ORR group. The vRR calculates and advertises the BGP ORR best and backup/additional paths to all the vRR sites and ORR groups. For more information, refer to the BGP ORR with Path Advertisement Policy, on page 3.

You can configure the following router policy to enable the vRR to advertise the best, backup, or additional paths:

- **"Is-destination"**: This is a term assigned to the policy that advertises a specific type of path, such as best, backup, or add-path, to a designated neighbor. This policy doesn't apply to ORR neighbors.
- **"Set path-selection backup 1 advertise"**: This is a term assigned to the policy that enables the vRR to advertise only the best and backup path to the vRR sites and ORR groups.
- **"Set path-selection all advertise"**: This is a term assigned to the policy that enables the vRR to advertise the best and additional paths to the vRR sites and ORR groups.

Restrictions and Limitations for BGP ORR with Path Advertisement Policy

The following are the restrictions and limitations for BGP ORR with path advertisement policy:

- By default, the vRR advertises the calculated best and backup paths to the ORR group. If required you can enable vRR to advertise all the paths to the ORR group using the *addpath-all* keyword in the **additional-paths selection** command.
- By default, the vRR advertises all the paths to all vRR sites.
- If you have not assigned a policy to the ORR group and if the AFI router policy is not assigned with a default path, then the vRR does not calculate the best paths for that ORR group.
- The router policy **"Is-destination"** option is not supported with ORR route-policy attach-point. For example, if you have configured the "is-destination" path, the "is-destination" path is advertised only to the vRR but not for PEs.
- The **"Set path-selection backup 1 install"** option doesn't support ORR groups as vRR is not part of the forwarding path; therefore, you cannot install ORR and backup paths on the forwarding table. If you configure this policy for an ORR group, the system ignores the install command and, by default, advertises the backup paths to the specified ORR groups.

Configuring BGP ORR with Path Advertisement Policy

Perform the following steps to configure the BGP ORR with path advertisement policy.

Step 1 Configure RPL policy "Is-destination".

Example:

```
Router# configure
Router(config)# route-policy Bestpath
Router(config-rpl)# if destination is-best-path then
Router(config-rpl)# pass
Router(config-rpl)# else
Router(config-rpl)# drop
Router(config-rpl)# endif
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

This policy allows vRR to pass the path provided by you only if it is the best path; otherwise, it will drop the route. This policy allows vRR to advertise only the best path to the vRR sites and it doesn't apply to ORR neighbors.

Step 2 Configure a RPL policy to advertise the best and back up path.

Example:

```
Router# configure
Router(config)# route-policy Backup
Router(config-rpl)# set path-selection backup 1 advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

This policy allows vRR to advertise the best and backup path.

Step 3 Configure a RPL policy to advertise all paths.

Example:

```
Router# configure
Router(config)# route-policy addpath-all
Router(config-rpl)# set path-selection all advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# route-policy pass
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# route-policy addpath-backup
Router(config-rpl)# set path-selection backup 1 advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# end
Router#
```

This policy allows vRR to advertise all the paths to the ORR groups.

Step 4 Configure an ORR group add path policy using the **optimal-route-reflection** command, and assign the address-family to the ORR group using the **address-family** command.

Example:

```
Router# configure
Router(config)# router bgp 13
Router(config-bgp)# optimal-route-reflection ipv4 orr-grp-1 1.1.1.1 route-policy addpath-backup
Router(config-bgp-router)# address-family ipv4 unicast
Router(config-bgp-router-af)# additional-paths selection route-policy addpath-all
Router(config-bgp-router-af)# optimal-route-reflection apply orr-grp-1
Router(config-bgp-router-af)# commit
```

Step 5 Configure the neighbor groups for BGP using the **neighbor-group** command.

Example:

```
Router# configure
Router(config)# neighbor-group PR
Router(config-nbrgrp)# address-family ipv4 unicast
Router(config-nbrgrp-af)# route-reflector-client
Router(config-nbrgrp-af)# optimal-route-reflection orr-grp-1
Router(config-nbrgrp-af)# route-policy pass in
Router(config-nbrgrp-af)# commit
```

Step 6 Apply the configured BGP neighbor group to the BGP neighbors using the **neighbor** command.

Example:

```
Router# configure
Router(config)# neighbor 10.10.10.10
Router(config-nbr)# use neighbor-group PR
```

Step 7 View the running configuration to verify the configuration that you have configured.

Example:

```
/*Enable to BGP Policies*/
route-policy addpath-all
  set path-selection all advertise
end-policy
route-policy pass
  pass
end-policy
route-policy addpath-backup
  set path-selection backup 1 advertise
end-policy

/*Configure ORR Add-Path and create ORR Groups*/
router bgp 13
  optimal-route-reflection ipv4 orr-grp-1 1.1.1.1 route-policy addpath-backup
  optimal-route-reflection ipv4 orr-grp-2 2.2.2.2
  address-family ipv4 unicast
    additional-paths selection route-policy addpath-all
    optimal-route-reflection apply orr-grp-1
    optimal-route-reflection apply orr-grp-2
!

/* Configure BGP neighbor groups*/
neighbor-group PR
  address-family ipv4 unicast
    route-reflector-client
    optimal-route-reflection orr-grp-1
    route-policy pass in
  !
!
neighbor-group BB
  address-family ipv4 unicast
    route-reflector-client
    route-policy PR_BB_COMM in
  !
!

/*Apply the neighbor groups*/
neighbor 10.10.10.10
  use neighbor-group PR
neighbor 3.3.3.3
  use neighbor-group BB
```

Step 8 Verify the BGP ORR with the add path policy configuration using the **show bgp** command.

Example:

```
Router# sh bgp 200.1.1.0/24 path-elements
BGP routing table entry for 200.1.1.0/24
Versions:
    Process     bRIB/RIB SendTblVer
    Speaker           84         84
        Flags: 0x043e3028+0x00000000;
Last Modified: Sep 7 20:21:38.000 for 00:11:45
Paths: (2 available, best #1)
Path-elements: 2
    Path ID: 1
        Gateway metric 0, Version 84
    Path: Nexthop 11.1.1.2, flags 0x4000000001440207
          Neighbor 11.1.1.2, Received Path ID 0
    Flags: 0x00000001
           status: valid
           path type: bestpath
           add-path action:
    Opaque: pelem=0xebf81d10
            net=0xabe435c, tblattr=0xa962f64 (ver 84)
            path=0xabf4450, path-tblattr=0xa962f64 (ver 84)
                      nobestpath-tblattr=0xa963234 (ver 0)
                       noaddpath-tblattr=0xa9631e4 (ver 0)
            bitfields=0xace9f64 (val=0x1a, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0xac145d8 (val=0x0, size=0)
            orr-ap-bitfields=0xac145d8 (val=0x2, size=0). <<<<<<<<<<<<<<<
            net-next=0xebf81cd4, tblattr-prev=0xebf81cd4, tblattr-next=0xebf81c98
Path ID: 2
    Gateway metric 0, Version 107
    Path: Nexthop 192.168.0.10, flags 0x4000000000040007
          Neighbor 192.168.0.2, Received Path ID 2
    Flags: 0x00000042
           status: valid
           path type: backup
           add-path action: advertise
    Opaque: pelem=0x10a0bba8
            net=0x109cbb14, tblattr=0x10399c84 (ver 156)
            path=0x109ebcd4, path-tblattr=0x10399c84 (ver 156)
                        nobestpath-tblattr=0x10399fa4 (ver 0)
                            noaddpath-tblattr=0x10399f54 (ver 0)
            bitfields=0x10c00624 (val=0x10, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0x10a2b8cc (val=0x0, size=1)
            orr-ap-bitfields=0x10a2b8a8 (val=0x6, size=1) <<<<<<<<<<<<<<<
            net-next=0x10a0b528, tblattr-prev=0x10a0bbe8, tblattr-next=0x0
```

Support for bgp bestpath igp-metric ignore Command

The **bgp bestpath igp-metric ignore** command enables the system to ignore Interior Gateway Protocol (IGP) metric while selecting the best path.

By default, BGP always prefers a path with the lowest IGP metric. When there are two paths, one with the IGP metric and the other without, then executing the **bgp bestpath igp-metric ignore** command results in BGP performing best path computation as if neither paths has the IGP metric.

The following example shows how to configure the software to ignore the interior gateway protocol (IGP) metric when performing best-path selection:. In this example, the command is configured in router BGP VRF configuration mode.

```
RP/0/0/CPU0:router#configure
RP/0/0/CPU0:router(config)#router bgp 50000
RP/0/0/CPU0:router(config-bgp)#vrf 1
RP/0/0/CPU0:router(config-bgp-vrf)#bgp bestpath igp-metric ignore
```

BFD for Multihop Paths

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Multihop BFDv4 and BFDv6 for iBGP and eBGP	Release 7.3.1	This feature provides sub-second forwarding failure detection for a destination more than one hop, and up to 255 hops away. This feature is supported on all the media-types that support BFD singlehop.

BFD multihop (BFD-MH) is a BFD session between two addresses that are not on the same subnet. An example of BFD-MH is a BFD session between PE and CE loopback addresses or BFD sessions between routers that are several TTL hops away. External and internal BGP applications support BFD multihop. BFD multihop supports BFD on arbitrary paths, which can span multiple network hops.

The BFD for Multihop Paths feature is supported on all the media-types that support BFD singlehop.

Note The Multihop BFDv4 and BFDv6 for iBGP and eBGP feature is not supported over MPLS/GRE Tunnel/SR.

Setting up BFD Multihop Session

A BFD multihop session is set up between a unique source-destination address pair provided by the client. A session can be set up between two endpoints that have IP connectivity. IPv4 addresses in both global routing table and in a VRF table are supported.

Note Aggressive timer is not recommended to use for the BFD Multipath sessions and the Multihop sessions. The recommend time is more than 100 ms x 3 = 300 ms.

Configure BFD IPv6 Multihop Session

When BFD is used with BGP, the BFD session type (singlehop or multihop) is configured based on the BGP configuration. If you configure eBGP-multihop keyword, the BFD session will also run in multihop mode; otherwise the session will run in singlehop mode.

Use the **bfd multihop ttl-drop-threshold** command to drop BFD packets coming from neighbors exceeding a certain number of hops.

- Configure BFD IPv6 multihop for eBGP neighbors
- Configure BFD IPv6 multihop for iBGP neighbors
- Enable BFD on BGP neighbor

Configure BFD IPv6 multihop for eBGP neighbors

```
Router# configure
Router(config)# bfd multipath include location 0/7/CPU0
Router(config)# router bgp 65001
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2 ebgp-multihop 255
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2 bfd fast-detect
```

Configure BFD IPv6 multihop for iBGP neighbors

```
Router# configure
Router(config)# bfd multipath include location 0/7/CPU0
Router(config)# router bgp 65001
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2
```

Enable BFD on a BGP neighbor

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# bfd minimum-interval 6500
Router(config-bgp)# bfd multiplier 7
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# bfd fast-detect
```

Running Configuration

The following is the running configuration for BFD IPv6 Multihop for eBGP neighbors:

```
Router# show running-configuration
bfd multipath include location 0/7/CPU0
router bgp 65001
neighbor 21:1:1:1:1:1:1:2 ebgp-multihop 255
neighbor 21:1:1:1:1:1:1:2 bfd fast-detect
```

The following is the running configuration for BFD IPv6 Multihop for iBGP neighbors:

```
Router# show running-configuration

bfd multipath include location 0/7/CPU0
router bgp 65001
neighbor 21:1:1:1:1:1:1:2
```

The following is the running configuration for BFD IPv6 Multihop for iBGP neighbors:

```
Router# show running-configuration

router bgp 120
 bfd minimum-interval 6500
 bfd multiplier 7
 neighbor 172.168.40.24
 remote-as 2002
 bfd fast-detect
```

Verification

```
Router# show bfd session
Tue Apr  7 06:16:36.982 UTC

Src Addr  Dest Addr VRF Name H/W NPU Local det time(int*mult)State

10.1.1.1  192.0.2.1 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.2  192.0.2.2 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.3  192.0.2.3 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.4  192.0.2.4 default  No  n/a  n/a  150ms(50ms*3)    UP


Router# show bfd ipv6 session
Tue Apr  7 06:16:45.012 UTC
Src Addr     Dest Addr         VRF Name Local det time(int*mult) State  Echo  Async
---------- -----------------  -------- ----------------------  -----  ----  -----
2001:DB8::1 2001:DB8:0:ABCD::1 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::2 2001:DB8:0:ABCD::2 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::3 2001:DB8:0:ABCD::3 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::4 2001:DB8:0:ABCD::4 default  0s(0s*0) 150ms(50ms*3)    UP
```

CVAC - Bootstrap Configuration Support

Cisco Virtual Appliance Configuration (CVAC) is an out-of-band configuration mechanism supported by several Cisco virtual routers. CVAC receives the configuration injected into the virtual router environment on a CD-ROM, disk image, or USB drive provided by the hypervisor. The configuration is detected and applied at startup time.

This allows the user to combine a new virtual router with a startup (bootstrap) configuration for initial deployment and is a very quick and convenient way of configuring many basic requirements (for example, the management ip address) that typically would have to be done manually.

Note CVAC works if there is no existing configuration, including the initial username and password you are prompted for on a new system.

Cisco IOS XRv 9000 Router fully supports CVAC with native KVM, Openstack Config Drive, and Virsh. CVAC is not supported on VMware ESXi.

Building the Bootstrap Configuration ISO

Cisco IOS XRv 9000 Router supports a plain-text configuration file on a single CD-ROM drive:

- **iosxr_config.txt**—provides standard XR configuration

This text file provides a simple list of configuration CLIs for CVAC to apply automatically. This operation is functionally equivalent to manually issuing a **copy iosxr_config.txt running-config** command.

Given one or more configuration files, you can create an ISO image suitable for insertion into Cisco IOS XRv 9000 Router with the following command:

```
mkisofs -output bootstrap.iso -l -V config-1 --relaxed-filenames --iso-level 2
iosxr_config.txt
```

Here is the sample output for **mkisofs** command on Ubuntu:

```
Warning: creating filesystem that does not conform to ISO-9660.
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
175 extents written (0 MB)
```

CVAC and KVM

In order to have CVAC process the config, add an extra drive to the Qemu command line (as the last drive):

```
-drive file=./bootstrap.iso,if=virtio,media=cdrom,index=3
```

If the configuration files are correctly provided and CVAC runs successfully, you will see these syslog messages:

```
  RP/0/0/CPU0:Dec 14 09:10:22.719 : config[1]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
committed by user  'CVAC'
Use 'show configuration commit changes 1000000001' to view the  changes.


  RP/0/0/CPU0:Dec 14 09:10:23.619 : cvac[2]:  %MGBL-CVAC-5-CONFIG_DONE :
```

Configuration was applied from file /disk0:/iosxr_config.txt.

If any configuration from the config file(s) is rejected, you will additionally see this syslog message:

```
RP/0/0/CPU0:Dec 14 09:10:23.619 : cvac[2]: %MGBL-CVAC-3-CONFIG_ERROR :  Errors were
encountered while applying configs from file  /etc/sysconfig/iosxr_config.txt. Please inspect
 'show configuration  failed' for details
```

Any configuration that did not fail is committed. Further debugging is available in the disk0:/cvac.log file.

CVAC and Virsh

1. Create an ISO image suitable for insertion into Cisco IOS XRv 9000 Router and follow the procedure explained in section Building the Bootstrap Configuration ISO.
2. The Virsh.xml file that is downloaded along with router's image has a Bootstrap section as shown here:

```
<!-- BootstrapSection -->
<!-- Example Bootstrap CLI ISO -->
<!-- <disk type='file' device='cdrom'> -->
<!-- <driver name='qemu' type='raw'/> -->
<!-- <source file='<ISO with file iosxr_config.txt'/> -->
<!-- <target dev='vdc' bus='virtio'/> -->
<!-- <readonly/> -->
<!-- <alias name='bootstrap_CLI'/> -->
<!-- </disk> -->
```

3. Uncomment the Bootstrap section and locate the source file to the absolute path of the bootstrap ISO file on the machine launching the instance. For example:

```
<!--BootstrapSection -->
 <disk type='file' device='cdrom'>
 <driver name='qemu' type='raw'/>
```

```
<source file='/production/bootstrap.iso'/>
<target dev='vdc' bus='virtio'/>
<readonly/>
<alias name='bootstrap_CLI'/>
</disk>
```

CVAC and OpenStack Config-Drive

Config-drive is a mechanism to bootstrap a VM orchestrated in OpenStack with an initial configuration. Click here to know more about OpenStack config-drive.

If config-drive support is desired, a plaintext file with the initial XR configuration can be passed using this command line:

```
config-drive true user-data /iosxr_config.txt file /iosxr_config.txt=/iosxr_config.txt
```

The file *iosxr_config.txt* is the raw text file with the XR commands, not an ISO file used for the KVM command line or Virsh in earlier sections.

CVAC on Boot

On a newly installed VM instance with no configuration, CVAC behavior is straightforward: all configurations accepted through the parser are committed. CVAC maintains a signature (CRC) of the last applied config file.

On subsequent reboots, any CVAC config file passed into the system has its CRC checked against the last applied CVAC configuration CRC. If there is no change, nothing is done. This means that after an initial CVAC applied configuration, and subsequent configuration changes, if a CVAC configuration file is passed and it hasn't change, then the system configuration is not changed (or reverted).

If there is a change, the new configuration is applied over any existing configuration. This allows an initial CVAC configuration, subsequent configuration changes, and additional CVAC configuration changes on an already configured system.

CVAC does not try to rationalize any of the configurations are just fed into the parser over the already committed configuration. Any commands that pass the parser are committed, and, as mentioned previously, errors are noted in the log file.

ORR Support for FlexAlgo

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
ORR Support for FlexAlgo	Release 7.5.1	This feature 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.

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 extension to support Segment Routing Flexible Algorithm on an MPLS data-plane.

Prerequisites for Flexible Algorithm

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

Building Blocks of Segment Routing Flexible Algorithm

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

Flexible Algorithm Definition

Computing a path over a network require the use of many possible constraints. Some networks require the deployment of multiple planes. 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. You can define the mapping between the algorithm value and its meaning to provide maximum flexibility. When all the routers in the domain have the common understanding what the particular algorithm value represents, the computation for such an algorithm is consistent and the traffic is not subject to looping. In Flexible Algorithm, you define the meaning of the algorithm, not a standard.

Flexible Algorithm Membership

An algorithm defines how IGP computes the best path. Routers advertise the support for the algorithm as a node capability. Routers advertise the prefix-SIDs with algorithm value. Routers tightly couple the prefix-SIDs with the algorithm itself.

An algorithm is a one octet value. Routers reserve values 128 -255 for user-defined values. The routers use them 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 one or more dedicated routers advertising the definition of each Flexible Algorithm. Such an 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

Use the **advertise-definition** command to enable the router to advertise the definition for the particular Flexible Algorithm. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. The Flexible Algorithm will not be functional unless the router advertises the valid definition.

Flexible Algorithm Link Attribute Advertisement

The router used various link attributes during the Flexible Algorithm path calculation. For example, include or exclude rules that are based on link affinities can be part of the Flexible Algorithm definition, as defined in IETF draft.

Link attribute advertisements that the router uses during Flexible Algorithm calculation must use the Application-Specific Link Attribute (ASLA) advertisements, as defined in RFC8919 (IS-IS). In IS-IS, if the router sets the L-Flag in the ASLA advertisement, then the router uses legacy advertisements (IS-IS Extended availability TLV) instead.

The mandatory use of ASLA advertisements applies to the following link attributes:

- Minimum Unidirectional Link Delay
- TE Default Metric
- Administrative Group

Flexible Algorithm Prefix-SID Advertisement

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

Calculation of Flexible Algorithm Path

A router may compute the paths for multiple Flexible Algorithms. Configure a router to support a 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 it uses the Flexible Algorithm.

When computing the shortest path tree for a particular Flexible Algorithm:

- The routers prune from the topology all the nodes that do not advertise support for the Flexible Algorithm.
- If the Flexible Algorithm definition includes excluded affinities, then the routers prune all the links from the topology for which the routers advertise any of such affinities.
- Router uses the metric that is part of the Flexible Algorithm definition. If the routers do not advertise the metric for the particular link, the routers prune the link from the topology.

The routers compute Loop Free Alternate (LFA) paths, TI-LFA backup paths, and Microloop Avoidance paths for a particular Flexible Algorithm. The routers use the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use Prefix-SIDs advertised specifically for such Flexible Algorithm in order to enforce a back-up or microloop avoidance path.

Configuring Microloop Avoidance for Flexible Algorithm

By default, the Microloop Avoidance per Flexible Algorithm instance follows the Microloop Avoidance configuration for algo-0.

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, a LFA/TI-LFA per Flexible Algorithm instance follows LFA/TI-LFA configuration for algo-0.

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
```

Installation of Forwarding Entries for Flexible Algorithm Paths

The routers must install a Flexible Algorithm path to any prefix in the forwarding using the Prefix-SID that the routers advertised for such Flexible Algorithm. If you do not know the Prefix-SID for Flexible Algorithm, the routers do not install such Flexible Algorithm path in forwarding for such prefix.

The routers install only MPLS to MPLS entries for a Flexible Algorithm path. The routers do not install IP to IP or IP to MPLS entries. These follow the native IPG paths computed based on the default algorithm and regular IGP metrics.

Flexible Algorithm Prefix-SID Redistribution

Previously, the routers limited the prefix redistribution from IS-IS to another IS-IS instance or protocol to SR algorithm 0 (regular SPF) prefix SIDs. The routers did not redistribute SR algorithm 1 (Strict SPF) and SR algorithms 128–255 (Flexible Algorithm) prefix SIDs 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. The routers automatically enable this feature when you configure redistribution of IS-IS Routes with strict or Flexible Algorithm SIDs.

Flexible Algorithm Prefix Metric

A limitation of the existing Flexible Algorithm functionality in IS-IS is the inability to compute the best path to a prefix in a remote area or remote IGP domain. The routers advertise prefixes between IS-IS areas or between protocol domains, but the existing prefix metric does not reflect any of the constraints of the Flexible Algorithm path. Although you can compute the best Flexible Algorithm path to the interarea or redistribute the 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 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.

Configuring Flexible Algorithm

Use the following IS-IS configuration submode to configure the Flexible Algorithm:

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

Note Always use the IGP metric. If you enable the delay or TE metric, use the advertised delay or TE metric on the link as a metric for Flexible Algorithm computation.

```
router isis  instance flex-algo algo affinity { include-any | include-all | exclude-any}
name1, name2, …
```

name—Name of the affinity map

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

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

Configure the following command to include the Flexible Algorithm prefix metric in the advertised Flexible Algorithm definition in IS-IS :

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

Configure the following command to enable advertisement of the Flexible Algorithm definition in IS-IS:

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

Configuring Affinity

Configure the following command 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
```

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

Configure the following command to associate the affinity with an interface:

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

name—Name of the affinity-map

Configuring Prefix-SID Advertisement

Configure the following command to advertise prefix-SID for the 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
```

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

Configuring Flexible Algorithm Definitions

Configure the following commands to configure the Flexible Algorithm definition under the flex-algo submode:

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

Example: Configuring IS-IS Flexible Algorithm

```
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
!
```

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 1.1.1.1
  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
```

Configuring Flexible Algorithm

The following ISIS 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:

- **metric-type delay**

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

- **affinity exclude-any** *name1, name2, …*

  name—name of the affinity map

- **priority** *priority value*

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

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

```
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.

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

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

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

name—name of the affinity-map

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

IPv4 and IPv6 Traffic Redirect using Policy based Routing

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
IPv4 and IPv6 traffic redirect using Policy Based Routing (PBR)	Release 7.3.3	This feature allows you to redirect IPv4 and IPv6 subscriber traffic to a destination of your choice instead of the one destined originally. With this functionality, you get the flexibility to assign specific traffic through specialized paths, allowing you to efficiently manage service networks carrying voice, video, and data.

In today's converged networks carrying voice, video and data, the requirement to route traffic through specific paths instead of using paths that are from the routing protocols is increasing. The PBR Redirect feature caters to this need by redirecting subscriber traffic to a destination other than the original. This is an additional functionality to the Policy Based Routing (PBR) feature where the packet forwarding decisions are based on the policy configuration, instead of routing protocols.

For example, a service provider may want voice traffic to traverse through certain specialized link and data traffic through the regular routing path. Policy based redirect feature provides a mechanism that lets the service providers to redirect traffic based on a set of preconfigured match criteria into the IPv4 or IPv6 next-hop address.

General Guidelines

- PBR redirect supports IPv4 and IPv6 traffic types.
- PBR redirect and GRE features are mutually exclusive.
- PBR redirect supports destination and source address match.
- The router drops all the matching traffic when the next-hop address for PBR redirect isn't reachable.
- PBR redirect supports the following redirect types:
  - IPv4 and IPv6 next-hop.
  - Default VRF from named VRF

- Named VRF from default VRF

- PBR redirect supports VRF scaling.
- You can use the following commands to view the PBR policies and associated class maps:
  - **show pbr-pal km policy** *name* **vmr interface** *type* **sw**
  - **show pbr-pal km policy** *name* **info location** *ID*

Configuration

Use the following sample configuration to configure ACLs with PBR.

```
/* Configure an access list */

Router(config)# ipv4 access-list Test
Router(config-ipv4-acl)# 10 permit ipv4 any host 10.1.1.10
Router(config-ipv4-acl)# 20 permit ipv4 any host 10.2.3.4
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* Configure a class map for the access list */
Router(config)# class-map type traffic match-any Test A
Router(config-cmap)# match access-group ipv4 Test
Router(config-cmap)# end-class-map
Router(config)# commit

/* Configure an PBR policy map with the class map */
Router(config)# policy-map type pbr Test B
Router(config-pmap)# class type traffic Test A
Router(config-pmap-c)# redirect nexthop 192.168.10.1
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
```

Running Configuration

Validate the configuration by using the show command.

```
Router(config)# show running-config
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Mon Nov  6 17:31:23 2017 by UNKNOWN
!
ipv4 access-list Test
 10 permit ipv4 any host 10.1.1.10
 20 permit ipv4 any host 10.2.3.4
!
!
class-map type traffic match-any Test A
 match access-group ipv4 Test
 end-class-map
!
!
policy-map type pbr Test B
 class type traffic Test A
  redirect ipv4 nexthop 192.168.10.1
 !
```

```
 end-policy-map
!
```

Verification

The following show commands help in better debuggability for IPv4 and IPv6 traffic redirect using PBR.

```
Router#show pbr-pal km policy policy1 vmr interface GigabitEthernet 0/0/0/0 sw
Wed Feb 16 16:38:03.096 UTC
KM ifname GigabitEthernet0_0_0_0, ul_ifname NULL, policy info name policy1 pnum = 0
ingress_format = 0  egress_format = 1 km policy flags = 0x00001100  class# = 2 ref# = 1
num_intfs = 1 is_bvi 0 is_typhoon_tomahawk 0 np_start 255 np_end 255
================================================================================
B : type & id      E : ether type     VO : vlan outer     VI : vlan inner
Q : tos/exp/group  X : Reserved       DC : discard class  Fl : flags
F2: L2 flags       F4: L4 flags       SP/DP: L4 ports
T : IP TTL         D : DFS class#      L : leaf class#
Pl: Protocol       G : QoS Grp         M : V6 hdr ext.     C : VMR count

================================================================================
policy name policy1 and format type 0
Total Ingress TCAM entries: 2
 |B    Q  T  Fl Pl SP   DP   G  IPv4/v6 SA IPv4/v6 DA
================================================================================
V|0006 00 00 00 11 0000 0000 00 AC020400 00000000
M|FF00 00 00 80 FF 0000 0000 00 FFFFFF00 00000000
R| C=0 D=0 L=0
V|0006 00 00 00 00 0000 0000 00 00000000 00000000
M|FF00 00 00 00 00 0000 0000 00 00000000 00000000
R| C=1 D=1 L=1

================================================================================
Total Ingress and Egress TCAM entries: 2
```

```
Router#show pbr-pal km policy policy1 info location 0/0/CPU0
Wed Feb 16 16:39:44.503 UTC
KM ifname NULL, ul_ifname NULL, policy info name policy1 pnum = 0 ingress_format = 0
egress_format = 1 km policy flags = 0x00001100  class# = 2 ref# = 1 num_intfs = 1 is_bvi 0
 is_typhoon_tomahawk 0 np_start 255 np_end 255KM policy info
        name policy1
       pnum = 0  ingress_format = 0  egress_format = 1 km policy flags = 0x00001100  class#
 = 2
        ref# = 1 num_intfs = 1
Interface Details
=================
No.  Interface Name                VMR ID   Ingress
1    GigabitEthernet0/0/0/0        1
```

gNMI Bundling of Telemetry Updates

Table 6: Feature History Table

Feature Name	Release Information	Description
gNMI Bundling Size Enhancement	Release 7.8.1	With gRPC Network Management Interface (gNMI) bundling, the router internally bundles multiple gNMI `Update` messages meant for the same client into a single gNMI `Notification` message and sends it to the client over the interface. You can now optimize the interface bandwidth utilization by accommodating more gNMI updates in a single notification message to the client. We have now increased the gNMI bundling size from 32768 to 65536 bytes, and enabled gNMI bundling size configuration through Cisco native data model. Prior releases allowed only a maximum bundling size of 32768 bytes, and you could configure only through CLI. The feature introduces new XPaths to the `Cisco-IOS-XR-telemetry-model-driven-cfg.yang` Cisco native data model to configure gNMI bundling size. To view the specification of gNMI bundling, see Github repository.

To send fewer number of bytes over the gNMI interface, multiple gNMI `Update` messages pertained to the same client are bundled and sent to the client to achieve optimized bandwidth utilization.

The router internally bundles multiple gNMI `Update` messages in a single gNMI `Notification` message of gNMI `SubscribeResponse` message. Cisco IOS XR software Release 7.8.1 supports gNMI bundling size up to 65536 bytes.

Router bundles multiple instances of the same client. For example, a router bundles interfaces `MgmtEth0/RP0/CPU0/0`, `FourHundredGigE0/0/0/0`, `FourHundredGigE0/0/0/1`, and so on, of the following path.

- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`

Router does not bundle messages of different client in a single gNMI `Notification` message. For example,

- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`
- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/protocols`

  Data under the container of the client path cannot be split into different bundles.

The gNMI `Notification` message contains a timestamp at which an event occurred or a sample is taken. The bundling process assigns a single timestamp for all bundled `Update` values. The notification timestamp is the first message of the bundle.

Note

- ON-CHANGE subscription mode does not support gNMI bundling.
- Router does not enforce bundling size in the following scenarios:
  - At the end of (N-1) message processing, if the notification message size is less than the configured bundling size, router allows one extra instance which could result in exceeding the bundling size.
  - Data of a single instance exceeding the bundling size.
- The XPath: `network-instances/network-instance/afts` does not support bundling.

QoS on IPv4 GRE Tunnels

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
QoS on IPv4 GRE Tunnels	Release 7.3.3	This feature, which enables the capability to define and control the QoS for both incoming and outgoing customer traffic on provider edge (PE) routers in a service provider network, is introduced.

QoS support on IPv4 GRE tunnels enables applying the policy map directly on the IPv4 GRE interface. This enables aggregate policing and marking on the tunnel. The policy can be applied on the ingress side of the tunnel to mark and police payload traffic going into the tunnel. QoS is not supported on traffic egressing out of the tunnel.

For information on GRE tunnels, see the chapter *Implementing Generic Routing Encapsulation* in the*Cisco ASR 9000 Series Aggregation Services Router MPLS Layer 3 VPN Configuration Guide*.

Note For HQOS license on XRv9000, the license is calculated based on both input and output traffic on the ingress and egress interface only when the service policy is configured on both ingress and egress interface.

Restrictions

The following restrictions apply to QoS on GRE tunnels:

- Any other payload traffic except that of IPv4, IPv6, and MPLS isn't supported.
- QoS on GRE tunnels currently supports only one line card and one network processor.
- The QoS on GRE service policy is supported in the egress direction. You apply an egress service policy at the line card and an ingress policy at the GRE Tunnel.
- The QoS on GRE service policy isn’t supported in the ingress direction.
- The QoS on GRE tunnels functionality doesn't support shaping and queuing actions.

- The QoS on GRE tunnels feature doesn't support marking and conditional marking actions.
- You can't attach a policy map that has percentage-based policing configured.
- The QoS on GRE tunnels feature supports only policing actions. The policer can be a single-rate, two-color policer (1R2C).
- Conform traffic is transmitted, and exceed action is dropped.

Classification for IPv4 GRE tunnel traffic

The following table indicates the support for various payload traffic QoS fields on an IPv4 GRE tunnel for classification.

QoS field	Classification	
	Ingress	Egress
Precedence	No	Yes
Tunnel Precedence	No	No
VLAN	No	No
Class of Service (CoS)	No	No
Drop Eligibility Indicator (DEI)	No	No
IPv4 L3 field	No	Yes

Example

The following example shows application of a two-level hierarchical policy with single-rate, two-color policer (1R2C) on a GRE tunnel.

```
Router(config)#class-map match-any mydata
Router(config-cmap)#match mpls experimental topmost 3 4
Router(config-cmap)#end-class-map
Router(config)#class-map match-any mycontrol
Router(config-cmap)#match mpls experimental topmost 1 2
Router(config-cmap)#end-class-map
Router(config)#policy-map child
Router(config-pmap)#class mycontrol
Router(config-pmap-c)#police rate 2 gbps
Router(config-pmap-c-police)#conform-action transmit
Router(config-pmap-c-police)#exceed-action transmit
Router(config-pmap-c-police)#class class-default
Router(config-pmap-c)#police rate 1 gbps
Router(config-pmap-c-police)#conform-action transmit
Router(config-pmap-c-police)#exceed-action transmit
Router(config-pmap-c-police)#end-policy-map
Router(config)#policy-map parent_gre
Router(config-pmap)#class class-default
Router(config-pmap-c)#service-policy child
Router(config-pmap-c)#police rate 5 gbps
Router(config-pmap-c-police)#child-conform-aware
Router(config-pmap-c-police)#end-policy-map
Router(config)#interface tunnel-ip1
Router(config-if)#service-policy output parent_gre
```

```
Router(config-if)#ipv4 address 12.0.0.1 255.255.255.0
Router(config-if)#tunnel source TenGigE0/0/0/1
Router(config-if)#tunnel destination 15.1.1.2
```

Accessing the Networking Stack

The Cisco IOS XR Software serves as a networking stack for communication. This section explains how applications on IOS XR can communicate with internal processes, and with servers or outside devices.

Note We strongly recommend setting the MTU value of 1514 (default) or higher to avoid dropping fragment packets.

Communication Outside Cisco IOS XR

Table 8: Feature History Table

Feature Name	Release Information	Description
Virtual IP address in the Linux networking stack	Release 7.5.4	Virtual IP addresses allow a single IP address to connect to the current active RP after an RP switchover event. In addition, this functionality enables your network stack to support virtual IP addresses for third-party applications and IOS XR applications that use the Linux networking stack.

To communicate outside Cisco IOS XR, applications use the `fwdintf` interface address that maps to the `loopback0` interface or a configured Gigabit Ethernet interface address. For information on the various interfaces on IOS XR, see Application Hosting on the Cisco IOS XR Linux Shell, on page 40.

To have an iPerf or Chef client on IOS XR communicate with its respective server outside IOS XR, you must configure an interface address as the source address on XR. The remote servers must configure this route address to reach the respective clients on IOS XR.

Virtual addresses can be configured to access a router from the management network such as gRPC using a single virtual IP address. On a device with two or more RPs, the virtual address refers to the management interface that is currently active. This functionality can be used across RP failover without the information of which RP is currently active. This is applicable to the Linux packet path.

This section provides an example of configuring a Gigabit Ethernet interface address as the source address for external communication.

Using a Gigabit Ethernet Interface for External Communication

To configure a GigE interface on IOS XR for external communication, use these steps:

1. Configure a GigE interface.

```
RP/0/RP0/CPU0:ios(config)# interface GigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:ios(config-if)# ipv4 address 192.57.43.10 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# commit
Fri Oct 30 07:51:14.785 UTC
RP/0/RP0/CPU0:ios(config-if)# exit
RP/0/RP0/CPU0:ios(config)# exit
```

2. Verify whether the configured interface is up and operational on IOS XR.

```
RP/0/RP0/CPU0:ios# show ipv4 interface brief
Fri Oct 30 07:51:48.996 UTC

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1         Up                    Up
Loopback1                      8.8.8.8         Up                    Up
GigabitEthernet0/0/0/0         192.164.168.10  Up                    Up
GigabitEthernet0/0/0/1         192.57.43.10    Up                    Up
GigabitEthernet0/0/0/2         unassigned      Shutdown              Down
MgmtEth0/RP0/CPU0/0            192.168.122.197 Up                    Up
RP/0/RP0/CPU0:ios#
```

3. Enter the Linux bash shell and verify if the configured interface is up and running.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash


[xr-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Gi0_0_0_1 Link encap:Ethernet  HWaddr 52:46:2e:49:f6:ff
          inet addr:192.57.43.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:2eff:fe49:f6ff/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:294 (294.0 B)  TX bytes:504 (504.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
```

```
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:392 (392.0 B)  TX bytes:532 (532.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:672 (672.0 B)  TX bytes:672 (672.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

4. Exit the Linux bash shell and configure the GigE interface as the source address for external communication.

```
[xr-vm_node0_RP0_CPU0:~]$ exit

RP/0/RP0/CPU0:ios# config
Fri Oct 30 08:55:17.992 UTC
RP/0/RP0/CPU0:ios(config)# tpa address-family ipv4 update-source gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:ios(config)# commit
Fri Oct 30 08:55:38.795 UTC
```

Note By default, the `fwdintf` interface maps to the `loopback0` interface for external communication. This is similar to binding a routing process or router ID to the `loopback0` interface. When you use the `tpa address-family ipv4 update-source` command to bind the `fwdintf` interface to a Gigabit Ethernet interface, network connectivity can be affected if the interface goes down.

5. Enter the Linux bash shell and verify whether the GigE interface address is used by the `fwdintf` interface for external communication.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash

[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 192.57.43.10
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.197
[xr-vm_node0_RP0_CPU0:~]$
```

External communication is successfully enabled on IOS XR.

East-West Communication for Third-Party Applications

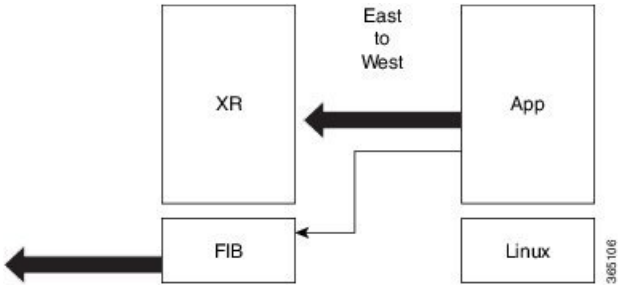
East-West communication on IOS XR is a mechanism by which applications hosted in containers interact with native XR applications (hosted in the XR control plane).

The following figure illustrates how a third-party application hosted on IOS XR interacts with the XR Control Plane.

The application sends data to the Forwarding Information Base (FIB) of IOS XR. The application is hosted in the east portion of IOS XR, while the XR control plane is located in the west region. Therefore, this form of communication between a third-party application and the XR control plane is termed as East-West (E-W) communication.

Third-party applications such as Chef Client and Puppet Agent use this mode of communication to configure and manage containers, packages, and applications on IOS XR. In the future, this support could be extended to IOS XR, configured and managed by such third-party applications.

Figure 3: East-West Communication on IOS XR



For a third-party application to communicate with IOS XR, the Loopback1 interface must be configured. This is explained in the following procedure.

1. Configure the Loopback1 interface on IOS XR.

```
RP/0/RP0/CPU0:ios(config)# interface Loopback1
RP/0/RP0/CPU0:ios(config-if)# ipv4 address 8.8.8.8/32
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# commit
RP/0/RP0/CPU0:ios(config-if)# exit
RP/0/RP0/CPU0:ios(config)#
```

2. Verify the creation of the Loopback1 interface.

```
RP/0/RP0/CPU0:ios# show ipv4 interface brief
Thu Nov 12 10:01:00.874 UTC

Interface                  IP-Address      Status                Protocol
Loopback0                  1.1.1.1         Up                    Up
Loopback1                  8.8.8.8         Up                    Up
GigabitEthernet0/0/0/0     192.164.168.10  Up                    Up
GigabitEthernet0/0/0/1     192.57.43.10    Up                    Up
GigabitEthernet0/0/0/2     unassigned      Shutdown              Down
MgmtEth0/RP0/CPU0/0        192.168.122.197 Up                    Up
RP/0/RP0/CPU0:ios#
```

3. Enter the third-party network namespace or global VRF depending on the version of IOS XR version you are using for your network.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash
```

```
/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash
```

4. Verify whether the Loopback1 interface address has been mapped to the E-W interface.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 192.57.43.10
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.197
[xr-vm_node0_RP0_CPU0:~]$
```

Application Hosting on the Cisco IOS XR Linux Shell

Linux supports an entire ecosystem of applications and tools that have been created, tested, and deployed by system administrators, developers, and network engineers over the last few decades. Linux is well suited for hosting servers with or without applications, because of its stability, security, scalability, reduced cost for licensing, and the flexibility it offers to customize applications for specific infrastructure needs.

With a growing focus on DevOps style workflows that focus on automation and ease of integration, network devices need to evolve and support standard tools and applications that make the automation process easier. A standardized and shared tool chain can boost speed, efficiency, and collaboration. IOS XR is developed from a Yocto-based Wind River Linux 7 distribution. The OS is RPM based and well suited for embedded systems.

IOS XR enables hosting of 64-bit Linux applications on the box, and has the following advantages:

- Seamless integration with configuration management applications
- Easy access to file systems
- Ease of operation

To host a Linux application on IOS XR, you must be familiar with the Linux shell on XR.

A typical Linux OS provides a single set of network interfaces and routing table entries that are shared across the OS. With the introduction of network namespaces, Linux provides multiple instances of network interfaces and routing tables that operate independently.

Note Support for network namespaces varies across different distributions of the Linux OS. Ensure that the distribution you are planning to use for application hosting supports network namespaces.

Network Namespaces on IOS XR

There are two ways of accessing the IOS XR Linux shell, depending on the version of Cisco IOS XR that you are using in your network.

- If you are using **Cisco IOS XR Version 6.0.0**, then you must use the procedure in Accessing the Third-Party Network Namespace on Cisco IOS XR Linux Shell, on page 44. Accessing the XR Linux shell takes you to the default network namespace, XRNNS. You must navigate from this namespace to access the third-party network namespace (TPNNS), where all the third-party application interfaces reside. There is a difference between what you can access and view at the XR router prompt, and what you can access and view at the XR Linux Shell.

- If you are using **Cisco IOS XR Version 6.0.2** and higher, then you must use the procedure in Accessing Global VRF on the Cisco IOS XR Linux Shell, on page 41. Accessing the XR Linux shell takes you directly to the third-party network namespace, renamed as global VRF. You can run bash commands at the XR router prompt itself to view the interfaces and IP addresses stored in global VRF. Navigation is faster and more intuitive in this version of IOS XR.

Accessing Global VRF on the Cisco IOS XR Linux Shell

The Third-Party Network Namespace (TPNNS) is renamed as Global VRF (global-vrf) in Cisco IOS XR Version 6.0.2 and higher. When you access the Cisco IOS XR Linux shell, you directly enter global VRF. This is described in the following procedure.

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

   You have reached the IOS XR router prompt.

2. View the ethernet interfaces on IOS XR.

```
RP/0/0/CPU0:ios# show ipv4 interface brief
...

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1/32      Up                    Up
GigabitEthernet0/0/0/0         10.1.1.1/24     Up                    Up
...


RP/0/RP0/CPU0:ios# show interfaces gigabitEthernet 0/0/0/0
...

GigabitEthernet0/0/0/0 is up, line protocol is up
Interface state transitions: 4
Hardware is GigabitEthernet, address is 5246.e8a3.3754 (bia
5246.e8a3.3754)
Internet address is 10.1.1.1/24
MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation ARPA,
Duplex unknown, 1000Mb/s, link type is force-up
output flow control is off, input flow control is off
loopback not set,
Last link flapped 01:03:50
ARP type ARPA, ARP timeout 04:00:00
Last input 00:38:45, output 00:38:45
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
12 packets input, 1260 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 2 broadcast packets, 0 multicast packets
0 runts, 0 giants, 0 throttles, 0 parity
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
12 packets output, 1224 bytes, 0 total output drops
Output 1 broadcast packets, 0 multicast packets
```

   The output displays the IP and MAC addresses of the `GigabitEthernet0/0/0/0` interface.

3. Verify whether the bash command runs in global VRF by running the **bash -c ifconfig** command to view the network interfaces.

```
RP/0/RP0/CPU0:ios# bash -c ifconfig
...
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

The presence of the following two interfaces confirms that you are in Global VRF:

`fwd_ew` is the interface used for communication (east to west) between third-party applications and IOS XR.

`fwdintf` is the interface used for communication between third-party applications and the network outside IOS XR.

4. Access the Linux shell by running the **bash** command.

```
RP/0/RP0/CPU0:ios# bash
Tue Aug 02 13:44:07.627 UTC
[xr-vm_node0_RP0_CPU0:~]$
```

5. (Optional) View the IP routes used by the `fwd_ew` and `fwdintf` interfaces.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 1.1.1.1
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.213
```

Alternative Method of Entering Global VRF on IOS XR

To directly enter global VRF on logging to IOS XR, without entering the **bash** command, you can use the `sshd_operns` service, as explained in the steps that follow. The procedure involves the creation of a non-root user in order to access the service. (Root users cannot access this service.)

Note On IOS XR, prior to starting a service that binds to an interface, ensure that the interface is configured, up, and operational.

To ensure that a service starts only after an interface is configured, include the following function in the service script:

```
. /etc/init.d/operns-functions
operns_wait_until_ready
```

The addition of the **operns_wait_until_ready** function ensures that the service script waits for one or more interfaces to be configured before starting the service.

1. (Optional) If you want the operns service to start automatically on reload, add the `sshd_operns` service and verify its presence.

```
bash-4.3# chkconfig --add sshd_operns
bash-4.3# chkconfig --list sshd_operns
sshd_operns      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bash-4.3#
```

2. Start the `sshd_operns` service.

```
bash-4.3# service sshd_operns start
Generating SSH1 RSA host key: [  OK  ]
Generating SSH2 RSA host key: [  OK  ]
Generating SSH2 DSA host key: [  OK  ]
  generating ssh ECDSA key...
Starting sshd: [  OK  ]

bash-4.3# service sshd_operns status
sshd (pid  6224) is running...
```

3. Log into the `sshd_operns` session as the non-root user created in Step 1.

```
host@fe-ucs36:~$ ssh devops@192.168.122.222 -p 57722
devops@192.168.122.222's password:
Last login: Tue Sep  8 20:14:11 2015 from 192.168.122.1
XR-vm_node0_RP0_CPU0:~$
```

4. Verify whether you are in global VRF by viewing the network interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

You are ready to use the IOS XR Linux shell for hosting applications.

Accessing the Third-Party Network Namespace on Cisco IOS XR Linux Shell

The Cisco IOS XR Linux shell provides a Third-Party Network Namespace (TPNNS) that provides the required isolation between third-party applications and internal XR processes, while providing the necessary access to XR interfaces for the applications. You can use the steps mentioned in this section to access the IOS XR Linux shell and navigate through the XRNNS (default XR Network Namespace) and the TPNNS.

Note This procedure is applicable only on Cisco IOS XR Versions 5.3.2 and 6.0.0. For accessing this namespace on other versions of Cisco IOS XR, see Accessing Global VRF on the Cisco IOS XR Linux Shell, on page 41.

Use these steps to navigate through the XR Linux shell.

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

You have reached the IOS XR router prompt.

2. View the ethernet interfaces on IOS XR.

```
RP/0/0/CPU0:ios# show ipv4 interface brief
...

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1/32      Up                    Up
GigabitEthernet0/0/0/0         10.1.1.1/24     Up                    Up
...


RP/0/RP0/CPU0:ios# show interfaces gigabitEthernet 0/0/0/0
...

GigabitEthernet0/0/0/0 is up, line protocol is up
Interface state transitions: 4
Hardware is GigabitEthernet, address is 5246.e8a3.3754 (bia
5246.e8a3.3754)
Internet address is 10.1.1.1/24
MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation ARPA,
Duplex unknown, 1000Mb/s, link type is force-up
output flow control is off, input flow control is off
loopback not set,
Last link flapped 01:03:50
ARP type ARPA, ARP timeout 04:00:00
Last input 00:38:45, output 00:38:45
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
12 packets input, 1260 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 2 broadcast packets, 0 multicast packets
0 runts, 0 giants, 0 throttles, 0 parity
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
12 packets output, 1224 bytes, 0 total output drops
Output 1 broadcast packets, 0 multicast packets
```

The output displays the IP and MAC addresses of the `GigabitEthernet0/0/0/0` interface.

3. Enter the **run** command to launch the IOS XR Linux bash shell.

You can also check the version of IOS XR when you are at the bash prompt.

```
RP/0/RP0/CPU0:ios# run
Wed Oct 28 18:45:56.168 IST
```

```
[xr-vm_node0_RP0_CPU0:~]$ uname -a
Linux xr-vm_node0_RP0_CPU0 3.10.19-WR7.0.0.2_standard #1 SMP Mon Jul 6
13:38:23 PDT 2015 x86_64 GNU/Linux
[xr-vm_node0_RP0_CPU0:~]$
```

Note To exit the Linux bash shell and launch the IOS XR console, enter the **exit** command:

```
[xr-vm_node0_RP0_CPU0:~]$ exit
exit
RP/0/RP0/CPU0:ios#
```

4. Locate the network interfaces by running the **ifconfig** command.

```
[xr-vm_node0_RP0_CPU0:~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:280 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31235 (30.5 KiB)  TX bytes:20005 (19.5 KiB)

eth-vf0   Link encap:Ethernet  HWaddr 52:54:00:34:29:44
          inet addr:10.11.12.14  Bcast:10.11.12.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe34:2944/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1566 (1.5 KiB)  TX bytes:1086 (1.0 KiB)

eth-vf1   Link encap:Ethernet  HWaddr 52:54:00:ee:f7:68
          inet6 addr: fe80::5054:ff:feee:f768/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:326483 errors:0 dropped:3 overruns:0 frame:0
          TX packets:290174 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24155455 (23.0 MiB)  TX bytes:215862857 (205.8 MiB)

eth-vf1.1794 Link encap:Ethernet  HWaddr 52:54:01:5c:55:8e
          inet6 addr: fe80::5054:1ff:fe5c:558e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:728 (728.0 B)  TX bytes:1234 (1.2 KiB)

eth-vf1.3073 Link encap:Ethernet  HWaddr e2:3a:dd:0a:8c:06
          inet addr:192.0.0.4  Bcast:192.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::e03a:ddff:fe0a:8c06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:317735 errors:0 dropped:3560 overruns:0 frame:0
          TX packets:257881 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:18856325 (17.9 MiB)  TX bytes:204552163 (195.0 MiB)

eth-vf1.3074 Link encap:Ethernet  HWaddr 4e:41:50:00:10:01
          inet addr:172.0.16.1  Bcast:172.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::4c41:50ff:fe00:1001/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
```

```
          RX packets:8712 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32267 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:723388 (706.4 KiB)  TX bytes:11308374 (10.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1635360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1635360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:182532711 (174.0 MiB)  TX bytes:182532711 (174.0 MiB)

tap123    Link encap:Ethernet  HWaddr c6:13:74:4b:dc:e3
          inet6 addr: fe80::c413:74ff:fe4b:dce3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:998 (998.0 B)
```

The output displays the internal interfaces (`eth0` through `eth-vf1.3074`) used by IOS XR. These interfaces exist in XR Network Namespace (XRNNS) and do not interact with the network outside IOS XR. Interfaces that interact with the network outside IOS XR are found in the Third Party Network Namespace (TPNNS).

5. Enter the TPNNS on the IOS XR bash shell.

```
[XR-vm_node0_RP0_CPU0:~]$ ip netns exec tpnns bash
```

6. View the TPNNS interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

The interfaces displayed in the output are replicas of the IOS XR interfaces in the Linux environment. (They have the same MAC and IP addresses.)

- `Gi0_0_0_0` is the IOS XR GigabitEthernet 0/0/0/0 interface.
- `Mg0_RP0_CPU0_0` is the IOS XR management interface, used for administrative operations on XR.
- `fwd_ew` is the interface used for communication (east to west) between third-party applications and IOS XR.
- `fwdintf` is the interface used for communication between third-party applications and the network outside IOS XR.
- `lo:0` is the IOS XR loopback0 interface used for communication between third-party applications and the outside network through the `fwdintf` interface. The loopback0 interface must be configured for applications to communicate outside XR. Alternatively, applications can also configure a GigE interface for external communication, as explained in the Communication Outside Cisco IOS XR, on page 36 section.

All interfaces that are enabled (with the **no shut** command) are added to TPNNS on IOS XR.

7. (Optional) View the IP routes used by the `fwd_ew` and `fwdintf` interfaces.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 1.1.1.1
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.213
```

Alternative Method of Entering the Third Party Network Namespace on IOS XR

To directly enter the TPNNS on logging to IOS XR, without entering the **ip netns exec tpnns bash** command, you can use the `sshd_tpnns` service, as explained in the steps that follow. The procedure involves the creation of a non-root user in order to access the service. (Root users cannot access this service.)

Note On IOS XR, prior to starting a service that binds to an interface, ensure that the interface is configured, up, and operational.

To ensure that a service starts only after an interface is configured, include the following function in the service script:

```
. /etc/init.d/tpnns-functions
tpnns_wait_until_ready
```

The addition of the **tpnns_wait_until_ready** function ensures that the service script waits for one or more interfaces to be configured before starting the service.

1. (Optional) If you want the TPNNS service to start automatically on reload, add the `sshd_tpnns` service and verify its presence.

```
bash-4.3# chkconfig --add sshd_tpnns
bash-4.3# chkconfig --list sshd_tpnns
sshd_tpnns      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bash-4.3#
```

2. Start the `sshd_tpnns` service.

```
bash-4.3# service sshd_tpnns start
Generating SSH1 RSA host key: [  OK  ]
Generating SSH2 RSA host key: [  OK  ]
Generating SSH2 DSA host key: [  OK  ]
  generating ssh ECDSA key...
Starting sshd: [  OK  ]

bash-4.3# service sshd_tpnns status
sshd (pid  6224) is running...
```

3. Log into the `sshd_tpnns` session as the non-root user created in Step 1.

```
host@fe-ucs36:~$ ssh devops@192.168.122.222 -p 57722
devops@192.168.122.222's password:
Last login: Tue Sep  8 20:14:11 2015 from 192.168.122.1
XR-vm_node0_RP0_CPU0:~$
```

4. Verify whether you are in TPNNS by viewing the interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
```

```
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

You are ready to use the IOS XR Linux shell for hosting applications.