

Implementing Cisco Express Forwarding

- Implementing Cisco Express Forwarding, on page 1

Implementing Cisco Express Forwarding

Cisco Express Forwarding (CEF) is an advanced, Layer 3 IP switching technology. CEF optimizes network performance and scalability for networks with large and dynamic traffic patterns, such as the Internet, on networks characterized by intensive web-based applications, or interactive sessions. CEF is an inherent feature and the users need not perform any configuration to enable it. If required, the users can change the default route purge delay and static routes. Cisco NCS 5500 Series Routers supports only single stage forwarding.

Components

Cisco IOS XR software CEF always operates in CEF mode with two distinct components:

- Forwarding Information Base (FIB) database: The protocol-dependent FIB process maintains the forwarding tables for IPv4 and IPv6 unicast in the route processor and line card (LC). The FIB on each node processes Routing Information Base (RIB) updates, performing route resolution and maintaining FIB tables independently in the route processor and line card (LC). FIB tables on each node can be slightly different.
- Adjacency table—a protocol-independent adjacency information base (AIB)

CEF is a primary IP packet-forwarding database for Cisco IOS XR software. CEF is responsible for the following functions:

- Software switching path
- Maintaining forwarding table and adjacency tables (which are maintained by the AIB) for software and hardware forwarding engines

The following features are supported for CEF on Cisco IOS XR software:

- Bundle interface support
- Multipath support
- Route consistency
- High availability features such as packaging, restartability, and Out of Resource (OOR) handling
- OSPFv2 SPF prefix prioritization

- BGP attributes download

CEF Benefits

- Improved performance—CEF is less CPU-intensive than fast-switching route caching. More CPU processing power can be dedicated to Layer 3 services such as quality of service (QoS) and encryption.
- Scalability—CEF offers full switching capacity at each line card.
- Resilience—CEF offers an unprecedented level of switching consistency and stability in large dynamic networks. In dynamic networks, fast-switched cache entries are frequently invalidated due to routing changes. These changes can cause traffic to be process switched using the routing table, rather than fast switched using the route cache. Because the Forwarding Information Base (FIB) lookup table contains all known routes that exist in the routing table, it eliminates route cache maintenance and the fast-switch or process-switch forwarding scenario. CEF can switch traffic more efficiently than typical demand caching schemes.

The following CEF forwarding tables are maintained in Cisco IOS XR software:

- IPv4 CEF database—Stores IPv4 Unicast routes for forwarding IPv4 unicast packets
- IPv6 CEF database—Stores IPv6 Unicast routes for forwarding IPv6 unicast packets
- MPLS LFD database—Stores MPLS Label table for forwarding MPLS packets

Verifying CEF

To view the details of the IPv4 or IPv6 CEF tables, use the following commands:

- show cef {*ipv4 address* | *ipv6 address*} hardware egress npu location {node-id}

Displays the IPv4 or IPv6 CEF table. The next hop and forwarding interface are displayed for each prefix. The output of the **show cef** command varies by location.

```
Router# show cef 203.0.1.2 hardware egress
  203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0),
0x0 (0x0)
 Updated Nov 20 13:33:23.557
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
   via 203.0.1.2/32, HundredGigE0/0/0/9, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
```

```
Router# show cef ipv6 fccc:0:2:ce00:: hardware egress npu location 0/0/CPU0
Using NPU option is resource intensive and may result in system
instability and possibly result in traffic loss.
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] : y
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] :Do you really want to
continue[confirm with only 'y' or 'n'] [y/n] : y
Do you really want to continue[confirm with only 'y' or 'n'] [y/n] :Do you really want to
continue[confirm with only 'y' or 'n'] [y/n] :y
fccc:0:2:ce00::/64, version 725, SRv6 Headend, internal 0x1000001 0x220 (ptr 0x8dace0a8)
[1], 0x400 (0x8daa1138), 0x0 (0x8e664250)
 Updated Oct 12 20:35:51.433
 local adjacency to Bundle-Ether1301

 Prefix Len 64, traffic index 0, precedence n/a, priority 2
```

```
  gateway array (0x8d93b398) reference count 2, flags 0x0, source rib (7), 0 backups
                [4 type 3 flags 0x8401 (0x8d9efe68) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8daa1138, sh-ldi=0x8d9efe68]
  gateway array update type-time 1 Oct 10 11:22:17.595
 LDI Update time Oct 10 11:22:17.595
 LW-LDI-TS Oct 12 20:35:51.450
 Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
   via fe80::11/128, Bundle-Ether1301, 13 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x20049 [0x901b88b0 0x0]
    next hop fe80::11/128
    local adjacency
    Accounting: 0/0 packets/bytes output


 LEAF - HAL pd context :
 sub-type : IPV6, ecd_marked:0, has_collapsed_ldi:0
 collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
    PI:0x308dace0a8 PD:0x308dace140 rev:2474 type: IPV6 (1) TBL: 0xe0800000
    LEAF location: LPM
    FEC key: 0x14b40000a0d0
    LEAF ip6route HW: <<<< New information
            npu:0x0 fec index:0x2000ffce

    LWLDI:
        PI:0x308daa1138 PD:0x308daa1180 rev:2473 p-rev:2381  ldi type:IP
        FEC key: 0x14b40000a0d0 fec index: 0x2000ffce(65486) num paths: 1 bkup paths: 0
        IMP pattern:3
        PI:0x308daa1138 PD:0x308daa1180 rev:2473 p-rev:2381  dpa-rev:1070462
        FEC key: 0x14b40000a0d0 fec index: 0x2000ffce(65486) num paths: 1 bkup paths: 0
        Path:0  fec index: 0x2000ffce(65486) DSP: 0xc000001
                SRV6 Encap key: 0xf1c0000040011844 SRV6 Encap Id: 0x40011844 Remote: 0
                Alloc Size: 1 SID:  Func: 48 (LOC_STATS)
                Hardware Programmed SID: {}
                uSID Shift slots:0
        FEC:
        npu:0x0 fec:0x2000ffce port:0xc000001 encap:0x40011844

        SHLDI:
            PI:0x308d9efe68 PD:0x308d9eff10 rev:2381 dpa-rev:195026 cbf_enabled:0
pbts_enabled:0 surpf_enable:0 flag:0x0
            FEC key: 0x14340000a0d0 fec index: 0x2000ffc5(65477) num paths: 1 bkup paths:
0
            p-rev:2364
            Path:0 fec index: 0x2000ffc5(65477) DSP:0xc000001 Dest fec index: 0x0(0)
            FEC:  <<<< New information
            npu:0x0 fec:0x2000ffc5 port:0xc000001 encap:0x40010001

            TX-NHINFO:
                PI: 0x30901b88b0 PD: 0x30901b8950 rev:2364 dpa-rev:195015 Encap hdl: (nil)
               Encap id: 0x40010001 Remote: 0 L3 int: 45 flags: 0x7 transport_encap_id:0x0

                npu_mask: 0x1 DMAC: fc:82:a0:51:04:82
                ENCAP:  <<<< New information
                npu:0x0 mac:0x820451a082fc vlan:0x2d


    Load distribution: 0 (refcount 4)

    Hash  OK  Interface                Address
    0     Y   Bundle-Ether1301         fe80::11
```

- show cef {ipv4 | ipv6} summary

Displays a summary of the IPv4 or IPv6 CEF table.

```
Router#show cef ipv4 summary
Fri Nov 20 13:50:45.239 UTC

Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 4129
  56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes
  13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default routes
  56 load sharing elements, 24304 bytes, 1 references
  1 shared load sharing elements, 432 bytes
  55 exclusive load sharing elements, 23872 bytes
  0 route delete cache elements
  13 local route bufs received, 1 remote route bufs received,  0 mix bufs received
  13 local routes, 0 remote routes
  13 total local route updates processed
  0 total remote route updates processed
  0 pkts pre-routed to cust card
  0 pkts pre-routed to rp card
  0 pkts received from core card
  0 CEF route update drops, 0 revisions of existing leaves
  0 CEF route update drops due to version mis-match
  Resolution Timer: 15s
  0 prefixes modified in place
  0 deleted stale prefixes
  0 prefixes with label imposition, 0 prefixes with label information
  0 LISP EID prefixes, 0 merged, via 0 rlocs
 28 next hops
  1 incomplete next hop

 0 PD backwalks on LDIs with backup path
```

- show cef { ipv4 address | ipv6 address } detail

Displays the details of the IPv4 or IPv6 CEF table.

```
Router#show cef 203.0.1.2 detail
203.0.1.2/32, version 0, internal 0x1020001 0x0 (ptr 0x8d7db7f0) [1], 0x0 (0x8daeedf0), 0x0
 (0x0)
 Updated Nov 20 13:33:23.556
 local adjacency 203.0.1.2
 Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
  gateway array (0x8d84beb0) reference count 1, flags 0x0, source aib (10), 0 backups
                [2 type 3 flags 0x8401 (0x8d99a598) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8daeedf0, sh-ldi=0x8d99a598]
  gateway array update type-time 1 Nov 20 13:33:23.556
 LDI Update time Nov 20 13:33:23.556
 LW-LDI-TS Nov 20 13:33:23.556
   via 203.0.1.2/32, HundredGigE0/0/0/9, 3 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8cfc81a0 0x0]
    next hop 203.0.1.2/32
    local adjacency
        Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/9        203.0.1.2
```

- show adjacency detail

Displays detailed adjacency information, including Layer 2 information for each interface. The output of the show adjacency command varies by location.

```
Router#show adjacency detail

------------------------------------------------------------------------------
0/5/CPU0
------------------------------------------------------------------------------
Interface                  Address                 Version  Refcount Protocol
Hu0/5/0/12                  (interface)                 13        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/30                  (interface)                 31        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/19                  (interface)                 20        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/16                  (interface)                 17        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/23                  (interface)                 24        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/22                  (interface)                 23        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/1                   (interface)                  2        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/6                   (interface)                  7        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/10                  (interface)                 11        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/31                  (interface)                 32        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4


Hu0/5/0/28                  (interface)                 29        1(    0)
                           (interface entry)
                           mtu: 1500, flags 1 4
```

```
Hu0/5/0/35                 (interface)                  36        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/32                 (interface)                  33        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/15                 (interface)                  16        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/34                 (interface)                  35        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/2                  (interface)                   3        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/26                 (interface)                  27        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/0/0/9                  203.0.1.2                    50        2(    0) mpls
                          00109400000cea285f0b80248847
                          mtu: 8986, flags 1 0

Hu0/0/0/9                  203.0.1.2                    49        2(    0) ipv4
                          00109400000cea285f0b80240800
                          mtu: 8986, flags 1 0

Hu0/5/0/29                 (interface)                  30        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/33                 (interface)                  34        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/20                 (interface)                  21        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/24                 (interface)                  25        1(    0)
                          (interface entry)
                          mtu: 1500, flags 1 4

Hu0/5/0/0                  (interface)                   1        1(    0)
                          (interface entry)
```

```
                        mtu: 1500, flags 1 4

Hu0/5/0/4                (interface)                  5        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/8                (interface)                  9        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/3                (interface)                  4        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/27               (interface)                 28        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/7                (interface)                  8        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/14               (interface)                 15        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/11               (interface)                 12        1(    0)
                        (interface entry)
                        mtu: 1500, flags 1 4

Hu0/5/0/18               (interface)                 19        1(    0)
                        (interface entry)
```

Unicast Reverse Path Forwarding

Configuration of Unicast IPv4 and IPv6 Reverse Path Forwarding (uRPF) enables a router to verify the reachability of the source address in packets being forwarded. Configuring uRPF, both strict and loose modes, helps to mitigate problems caused by the introduction of spoofed IP source addresses into a network. Configuration of uRPF discards IP packets that lack a verifiable IP source address after a reverse lookup in the CEF table.

When **strict uRPF** is enabled, the source address of the packet is checked in the FIB. If the packet is received on the same interface that would be used to forward the traffic to the source of the packet, the packet passes the check and is further processed. Otherwise, the packet is dropped. Configure strict uRPF only where there is natural or configured symmetry. Internal interfaces are likely to have a routing asymmetry, that is, multiple routes to the source of a packet. Therefore, you should not implement strict uRPF on interfaces that are internal to the network.

Implementation of strict mode uRPF requires maintenance of a uRPF interfaces list for the prefixes. The list contains only the interfaces configured with strict mode uRPF. The interfaces are provided by the prefix path. The uRPF interface list is shared among the prefixes wherever possible.

When **loose uRPF** is enabled, the source address of the packet is checked in the FIB. If the source address exists and matches a valid forwarding entry, the packet passes the check and is further processed. Otherwise, the packet is dropped.

Note The behavior of strict uRPF varies slightly on the basis of platforms, the number of recursion levels, and the number of paths in Equal-Cost Multipath (ECMP) scenarios. A platform may switch to loose uRPF check for some or all prefixes, even though strict uRPF is configured. For example, if ECMP Path is eight or more, strict mode is converted to loose mode.

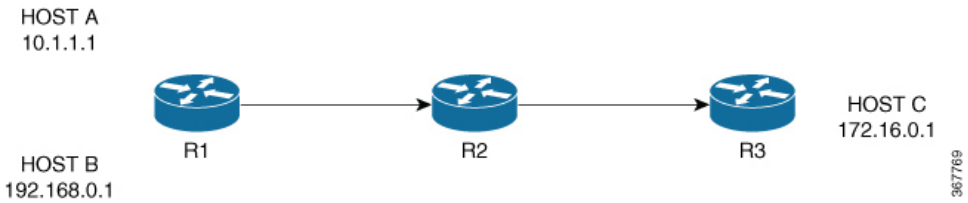
Loose and strict uRPF supports two options: **allow self-ping** and **allow default**. The **allow self-ping** option allows the source of the packet to ping itself. The **allow default** option allows the lookup result to match a default routing entry. When the **allow default** option is enabled with the strict mode of the uRPF, the packet is processed further only if it arrives through the default interface.

Restrictions

Consider the following restrictions when you configure uRPF:

- Global configuration followed by cold reload is required to enable or disable uRPF on the router.
- Configuration of uRPF per interface enables or disables uRPF mode in the hardware of the corresponding interface.
- Configuration of uRPF reduces the route scale to half. Half of the routing table is used for destination lookup and the other half is used for source lookup of received packets.
- Unicast RPF allows packets with 0.0.0.0 source addresses and 255.255.255.255 destination addresses to pass so that Bootstrap Protocol and Dynamic Host Configuration Protocol (DHCP) functions properly.
- Unicast RPF allows packets whose destination IP address is not a unicast address.
- The behavior of strict uRPF varies slightly on the basis of platforms, the number of recursion levels, and the number of paths in Equal-Cost Multipath (ECMP) scenarios. A platform may switch to loose uRPF check for some or all prefixes, even though strict uRPF is configured.
- The **allow self-ping** option is the default option in uRPF configuration for both strict and loose mode. You cannot disable the **allow-self-ping** option. The **allow-default** option needs to be configured specifically per interface.

Figure 1: uRPF Topology



In Figure 1, uRPF is enabled on Router R2 and R2 has the following FIB table entries:

- 10.1.1.0/24 [110/3] via 10.25.24.1, 2d08h, HundredGigE0/0/0/24
- 12.1.1.0/24 [110/3] via 10.25.24.1, 2d08h, HundredGigE0/0/0/25

- 14.0.5.0/24 [110/3] via 20.0.0.2, 2d08h, HundredGigE0/0/0/1

R2 allows the packet from Host 1 to be routed because Host 1 subnet is available in R2's FIB table. However, R2 does not allow Host 3 to be routed because Host 3 subnet is not available in R2's FIB table. If strict uRPF is enabled on R2, then the source address 10.1.1.1/24 should be reachable through the same interface from which it is received. If loose uRPF is enabled on R2, then it is not mandatory that the source address 10.1.1.1/24 be reachable through the same interface from which it is received. The only criteria for a packet to be forwarded is that the host address should be present in R2's FIB table.

Configure Unicast Reverse Path Forwarding

Configuring Unicast Reverse Path Forwarding (uRPF) enables a router to verify the reachability of the source address of packets being forwarded. If the source IP address is not valid, the packet is discarded. This capability can limit the appearance of spoofed addresses in a network.

To configure uRPF on a hardware module, use the following steps:

1. Configure a hardware module in the global configuration mode and enable uRPF.
2. Reboot the router.
3. Configure an interface.
4. Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface.

Configuration

Use the following configuration to configure uRPF in strict mode:

```
/* Configure a hardware module in the global configuration mode and enable uRPF. */
Router# configure
Router(config)# hw-module urpf enable
Mon Sep 17 09:34:00.945 UTC
In order to activate/deactivate urpf, you must manually reboot the box.
Router(config)# commit


/* Reboot the router manually. */

/* Configure an interface. */
Router(config)# interface BVI1
Router(config-ipv6-acl)# description BVI INTERFACE
Router(config-ipv6-acl)# commit

/* Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface. */
Router(config)# ipv4 address 11.1.1.1 255.255.255.0
Router(config-pifib-policer-global)# ipv4 verify unicast source reachable-via rx
Router(config-pifib-policer-global)# commit
```

Use the following configuration to configure uRPF in loose mode:

```
/* Configure a hardware module in the global configuration mode and enable uRPF. */
Router# configure
Router(config)# hw-module urpf enable
Mon Sep 17 09:34:00.945 UTC
In order to activate/deactivate urpf, you must manually reboot the box.
Router(config)# commit
```

```
/* Reboot the router manually. */

/* Configure an interface. */
Router(config)# interface BVI1
Router(config-ipv6-acl)# description BVI INTERFACE
Router(config-ipv6-acl)# commit

/* Configure IPv4 or IPv6 uRPF through strict or loose mode under the interface. */
Router(config)# ipv4 address 11.1.1.1 255.255.255.0
Router(config-pifib-policer-global)# ipv4 verify unicast source reachable-via any
Router(config-pifib-policer-global)# commit
```

Verification

Use the following command to check the uRPF status:

```
Router#show prm server  profile-status software
Software PROFILE STATUS VARIABLES
**************************************
BGP-3107 LB             : 0
LB Seed                 : 0x65c5208d
LB Seed Cfg             : NO
LB Sub-Sel-Offset Cfg   : NO
LB SubSel               : Hash Field B1
LB Offset               : 13
SR-PE Mode              : 0
Current URPF            : YES
Configured URPF         : YES
Current HW Profile      : SP-Profile
Configured HW Profile   : SP-Profile
**************************************
```

Per-Flow Load Balancing

The system inherently supports the 7-tuple hash algorithm. Load balancing describes the functionality in a router that distributes packets across multiple links based on Layer 3 (network layer) and Layer 4 (transport layer) routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.

Per-flow load balancing performs these functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Incoming data traffic is evenly distributed over multiple equal-cost connections member links within a bundle interface.
- Layer 2 bundle and Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows . If it is an IPv4 or an IPv6 payload, then a 7-tuple hashing is done. If it is an MPLS payload with three or less labels, then the hardware parses the payload underneath and identifies whether the payload packet has an IPv4 or an IPv6 header. If it is an IPv4 or IPv6 header, then a 4-tuple hashing is performed based on the IP source, IP destination, router ID, and label stack; otherwise, an MPLS label based hashing is performed. In case of MPLS label-based hashing, the top 4 labels are used in hash computation. However, for Cisco NC57 line cards, all the labels are used for MPLS label-based hashing..
- A 7-tuple hash algorithm provides more granular load balancing and used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or

on a bundle interface. In addition, load balancing over member links can occur within a Layer 2 bundle interface.

- The 7-tuple load-balance hash calculation contains:
  - Source IP address
  - Destination IP address
  - IP Protocol type
  - Router ID
  - Source port
  - Destination port
  - Input interface



Note Cisco NCS 5700 line cards support 6-tuple load-balance hash calculation because input interface is not considered as a parameter for load-balance hash calculations.



Note In ECMP, Cisco NCS 5500 line cards support 6-tuple load-balancing hash calculations because the input interface is not considered a parameter for load-balancing hash calculations. However, in LAG, Cisco NCS 5500 line cards support 7-tuple load-balancing hash calculations, where the input interface is considered a parameter for load-balancing hash calculations.

Load balancing decisions are taken based on a packet header, type of load balancing, type of scenario and platform specifics as follows:

- Packet header can contain one or many MAC, MPLS, IPv4 or IPv6 address, TCPor UDP headers, and so on.
- Load balancing can be done during ECMP or LAG (Bundle-Ether) forwarding.
- Scenarios can include IP forwarding, IP tunnel forwarding or decapsulation, MPLS forwarding or disaggregation, or Ethernet forwarding.
- The chipset type contains a packet's fields. These fields are considered for load balancing.

The following tables includes detailed list of options, list of scenarios, and headers fields to specify how ECMP or LAG load balancing is done.

Note:

- For NCS 5500 and NCS 5700 line cards, the fields superscripted with * are used for load balancing through LAG only. For the NCS 5500 line cards, the fields superscripted with * are used for load balancing through ECMP and LAG both. For example, in an MPLS forwarding scenario for Jericho line cards, for an MPLS packet with three labels, with an IPv4 or IPv6 header, and with L4 (TCP or UDP) ECMP load balancing is done based on:
  - All label values in the MPLS label stack

- Source and destination IPv4 or IPv6 addresses

However, for the same MPLS packet, LAG load balancing is done based on:

- All label values in the MPLS label stack
- Source and destination IPv4 or IPv6 addresses
- L4 source and destination ports

- Only the fields that are highlighted in bold font are used for load balancing hash calculations. For example, for IP forwarding for IPv4 or IPv6 header and L4 (TCP or UDP) header, ECMP or LAG load balancing is done based on:
  - Source and destination IPv4 or IPv6 addresses
  - L4 source and destination ports

- To modify the hashing algorithm that is used for ECMP and bundle member selection, use the **hw-module profile load-balance algorithm** command in XR Config mode.

Table 1: ECMP or LAG Load Balancing for IP Forwarding

Header 4	Header 3	Header 2	Header 1
		IPv4	ETH
		IPv6	ETH
	L4	**IPv4**	ETH
	L4	**IPv6**	ETH
GTP	**L4**	**IPv4**	ETH
GTP	**L4**	**IPv6**	ETH

Table 2: ECMP or LAG Load Balancing for IP Tunnel Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	**IPv4**	ETH
	L4*	**IPv4**	**IPv4**	ETH
		IPv6	**IPv4**	ETH
		IPv6	**IPv4**	ETH
	IPv4	**MPLS[1..3]**	**IPv4**	ETH
L4	**IPv4**	**MPLS[1..3]**	**IPv4**	ETH
	IPv6	**MPLS[1..3]**	**IPv4**	ETH

Header 5	Header 4	Header 3	Header 2	Header 1
L4	**IPv6**	**MPLS[1..3]**	**IPv4**	ETH
IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**IPv4**	ETH
		IPv4	**IPv6**	ETH
	L4*	**IPv4**	**IPv6**	ETH
		IPv6	**IPv6**	ETH
	L4*	**IPv6**	**IPv6**	ETH

Note For MPLS packets, up to four labels are used for hash calculation.

Table 3: ECMP or LAG Load Balancing for IP Tunnel Decapsulation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	IPv4	ETH
	L4	**IPv4**	IPv4	ETH
		IPv6	IPv4	ETH
	L4	**IPv6**	IPv4	ETH
	IPv4	**MPLS[1..3]**	IPv4	ETH
L4	**IPv4**	**MPLS[1..3]**	IPv4	ETH
	IPv6	**MPLS[1..3]**	IPv4	ETH
L4	**IPv6**	**MPLS[1..3]**	IPv4	ETH
IPv4	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH
IPv6	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH
MPLS[7..9]	**MPLS[4..6]**	**MPLS[1..3]**	IPv4	ETH

Note For MPLS packets, up to four labels are used for hash calculation.

Table 4: ECMP or LAG Load Balancing for MPLS Forwarding

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	ETH
	IPv4	**ETH**	**MPLS[1..3]**	ETH
	IPv6	**ETH**	**MPLS[1..3]**	ETH
		IPv4	**MPLS[1..3]**	ETH
		IPv6	**MPLS[1..3]**	ETH
	L4*	**IPv4**	**MPLS[1..3]**	ETH
	L4*	**IPv6**	**MPLS[1..3]**	ETH
	IPv4	**IPv4**	**MPLS[1..3]**	ETH
	IPv6	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv4	**IPv4**	**MPLS[1..3]**	ETH
L4	IPv6	**IPv4**	**MPLS[1..3]**	ETH
		MPLS[4..6]	**MPLS[1..3]**	ETH
IPv4	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	ETH	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv4	**MPLS[4..6]**	**MPLS[1..3]**	ETH
	IPv6	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
L4	**IPv6**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**IPv4**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv4	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH
IPv6	**MPLS[7..9]**	**MPLS[4..6]**	**MPLS[1..3]**	ETH



Note For MPLS packets with multiple labels, hash calculation is done based on the first five labels along other headers.

Table 5: ECMP or LAG Load Balancing for MPLS Deaggregation

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4	MPLS1	ETH
		IPv6	MPLS1	ETH
	L4*	**IPv4**	MPLS1	ETH
	L4*	**IPv6**	MPLS1	ETH
	IPv4	**IPv4**	MPLS1	ETH
	IPv6	**IPv4**	MPLS1	ETH
L4	IPv4	**IPv4**	MPLS1	ETH
L4	IPv6	**IPv4**	MPLS1	ETH
	IPv4	**ETH**	MPLS1	ETH
	IPv6	**ETH**	MPLS1	ETH
L4	**IPv4**	**ETH**	MPLS1	ETH
L4	**IPv6**	**ETH**	MPLS1	ETH

Table 6: ECMP or LAG Load Balancing for Ethernet Forwarding for IPoE Packets

Header 3	Header 2	Header 1
	IPv4	**ETH**
	IPv6	**ETH**
L4*	**IPv4**	**ETH**
L4*	**IPv6**	**ETH**

Table 7: ECMP or LAG Load Balancing Ethernet Forwarding for IPoE Packets with Complex Headers

Header 5	Header 4	Header 3	Header 2	Header 1
		IPv4*	**IPv4**	**ETH**
	L4	**IPv4***	**IPv4**	**ETH**
		IPv6*	**IPv4**	**ETH**
	L4	**IPv6***	**IPv4**	**ETH**
	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv4	**MPLS[1..3]***	**IPv4**	**ETH**

Header 5	Header 4	Header 3	Header 2	Header 1
	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**
L4	IPv6	**MPLS[1..3]***	**IPv4**	**ETH**
IPv4	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
IPv6	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**
MPLS[7..9]	MPLS[4..6]	**MPLS[1..3]***	**IPv4**	**ETH**

Note For MPLS packets with one through three labels, only the first label is used for load balancing along with other headers.

Table 8: ECMP or LAG Load Balancing Ethernet Forwarding for MPLS packets

Header 5	Header 4	Header 3	Header 2	Header 1
			MPLS[1..3]	**ETH**
		IPv4*	**MPLS[1..3]**	**ETH**
		IPv6*	**MPLS[1..3]**	**ETH**
	L4	**IPv4***	**MPLS[1..3]**	**ETH**
	L4	**IPv6***	**MPLS[1..3]**	**ETH**
	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv4	**IPv4***	**MPLS[1..3]**	**ETH**
L4	IPv6	**IPv4***	**MPLS[1..3]**	**ETH**
		MPLS[4..6]*	**MPLS[1..3]**	**ETH**
	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
L4	IPv6	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv4	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv6	IPv4	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**
IPv4	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**

Header 5	Header 4	Header 3	Header 2	Header 1
IPv6	MPLS[7..9]	**MPLS[4..6]***	**MPLS[1..3]**	**ETH**

Note For MPLS packets with multiple labels, hash calculation is done based on first five labels along with other headers.

Per-Destination Load Balancing

Per destination load balancing is used for packets that transit over a recursive MPLS path (for example, learned through BGP 3107). Per-destination load balancing means the router distributes the packets based on the destination of the route. Given two paths to the same network, all packets for destination1 on that network go over the first path, all packets for destination2 on that network go over the second path, and so on. This preserves packet order, with potential unequal usage of the links. If one host receives the majority of the traffic all packets use one link, which leaves bandwidth on other links unused. A larger number of destination addresses leads to more equally used links.

Configuring Static Route

Routers forward packets using either route information from route table entries that you manually configure or the route information that is calculated using dynamic routing algorithms. Static routes, which define explicit paths between two routers, cannot be automatically updated; you must manually reconfigure static routes when network changes occur. Static routes use less bandwidth than dynamic routes. Use static routes where network traffic is predictable and where the network design is simple. You should not use static routes in large, constantly changing networks because static routes cannot react to network changes. Most networks use dynamic routes to communicate between routers but might have one or two static routes configured for special cases. Static routes are also useful for specifying a gateway of last resort (a default router to which all unroutable packets are sent).

Configuration Example

Create a static route between Router A and B over a HundredGigE interface. The destination IP address is 203.0.1.2/32 and the next hop address is 1.0.0.2.




```
Router(config)#router static address-family ipv4 unicast
Router(config-static-afi)#203.0.1.2/32 HundredGigE 0/0/0/9 1.0.0.2
Router(config-static-afi)#commit
```

Running Configuration

```
Router#show running-config router static address-family ipv4 unicast
router static
 address-family ipv4 unicast
 203.0.1.2/32 HundredGigE 0/0/0/9 1.0.0.2
 !
!
```

Verification

Verify that the Next Hop Flags fields indicate COMPLETE for accurate functioning of the configuration.

The database, such as LPM, EXT-TCAM, and LEM, in which a prefix is updated is also provided through the output. Therefore, you can efficiently manage your network resources because you can understand the scaling of prefixes. You can also understand why a particular IP address configuration for a device fails and thereby debug easily.

```
Router#show cef 203.0.1.2/32 hardware egress details location 0/0/CPU0
Wed Nov  6 10:09:23.548 UTC
111.0.0.1/32, version 221, attached, internal 0x1000041 0x0 (ptr 0x8b00ea80) [1], 0x0
(0x8afd9768), 0x0 (0x0)
Updated Nov  6 10:08:07.424
Prefix Len 32, traffic index 0, precedence n/a, priority 2
  gateway array (0x8ae4baf0) reference count 1, flags 0x0, source rib (7), 0 backups
                [2 type 3 flags 0x40008441 (0x8af020c0) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8afd9768, sh-ldi=0x8af020c0]
  gateway array update type-time 1 Nov  6 10:08:07.423
LDI Update time Nov  6 10:08:07.423
LW-LDI-TS Nov  6 10:08:07.424
   via tunnel-ip1, 0 dependencies, recursive [flags 0x8]
    path-idx 0 NHID 0x0 [0x8ae0d728 0x0]
    local adjacency

 LEAF - HAL pd context :
 sub-type : IPV4, ecd_marked:0, has_collapsed_ldi:0
collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
    PI:0x308b00ea80 PD:0x308b00eb20 rev:293 type: IPV4 (0)
    LEAF location: LEM
    FEC key: 0x1640000dc6

    LWLDI:
        PI:0x308afd9768 PD:0x308afd97a8 rev:292 p-rev:291  ldi type:IP
        FEC key: 0x1640000dc6 fec index: 0x0(0) num paths:1, bkup paths: 0

REC-SHLDI HAL PD context :
ecd_marked:0, collapse_bwalk_required:0, load_shared_lb:0

    RSHLDI:
        PI:0x308af020c0 PD:0x308af02190 rev:291 dpa-rev:2448 flag:0x1
        FEC key: 0x1640000dc6 fec index: 0x2001ffd8(131032) num paths: 1
        p-rev:
        Path:0 fec index: 0x2001ffd8(131032) DSP fec index: 0x20000001(1),
               TEP Encap Id: 0x40013801

LEAF - HAL pd context :
 sub-type : IPV4, ecd_marked:0, has_collapsed_ldi:0
collapse_bwalk_required:0, ecdv2_marked:0,
HW Walk:
LEAF:
    PI:0x308b00e558 PD:0x308b00e5f8 rev:270 type: IPV4 (0)
    LEAF location: LEM
    FEC key: 0x1240000dc6

    LWLDI:
        PI:0x308afd9128 PD:0x308afd9168 rev:269 p-rev:268 268  ldi type:IP
        FEC key: 0x1240000dc6 fec index: 0x0(0) num paths:2, bkup paths: 0

        SHLDI:
          PI:0x308af00d88 PD:0x308af00e58 rev:268 dpa-rev:2433 cbf_enabled:0 pbts_enabled:0
 flag:0x0
```

```
            FEC key: 0x1240000dc6 fec index: 0x20000001(1) num paths: 2 bkup paths: 0
            p-rev:265 262
            Path:0 fec index: 0x2001ffdc(131036) DSP:0x15 Dest fec index: 0x0(0)
            Path:1 fec index: 0x2001ffdd(131037) DSP:0x16 Dest fec index: 0x0(0)

            TX-NHINFO:
              PI: 0x308c8d8298 PD: 0x308c8d8318 rev:265 dpa-rev:2431 Encap hdl: 0x308c84c350

                Encap id: 0x40010001 Remote: 0 L3 int: 1552 flags: 0x3
                npu_mask: 0x1 DMAC: f0:78:16:62:f6:a7

            TX-NHINFO:
              PI: 0x308c8d80b0 PD: 0x308c8d8130 rev:262 dpa-rev:2429 Encap hdl: 0x308c84c968

                Encap id: 0x40010000 Remote: 0 L3 int: 1551 flags: 0x3
                npu_mask: 0x1 DMAC: f0:78:16:62:f6:a6


    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   tunnel-ip1                10.10.10.1
```

Associated Commands

- router static
- show cef

BGP Attributes Download

The BGP Attributes Download feature enables you to display the installed BGP attributes in CEF.

- The **show cef bgp-attribute** command displays the installed BGP attributes in CEF.
- The **show cef bgp-attribute** *attribute-id* command and the **show cef bgp-attribute** *local-attribute-id* command are used to view the specific BGP attributes by attribute ID and local attribute ID.

Verification

```
Router# show cef bgp-attribute
Router ID is 216.1.1.1

IP CEF with switching (Table Version 0) for node0_RP0_CPU0

  Load balancing: L4
  Tableid 0xe0000000 (0x8cf5b368), Vrfid 0x60000000, Vrid 0x20000000, Flags 0x1019
  Vrfname default, Refcount 4129
  56 routes, 0 protected, 0 reresolve, 0 unresolved (0 old, 0 new), 7616 bytes
  13 rib, 0 lsd, 0:27 aib, 1 internal, 10 interface, 4 special, 1 default routes
  56 load sharing elements, 24304 bytes, 1 references
  1 shared load sharing elements, 432 bytes
  55 exclusive load sharing elements, 23872 bytes
  0 route delete cache elements
  13 local route bufs received, 1 remote route bufs received,  0 mix bufs received
  13 local routes, 0 remote routes
  13 total local route updates processed
  0 total remote route updates processed
  0 pkts pre-routed to cust card
  0 pkts pre-routed to rp card
```

```
  0 pkts received from core card
  0 CEF route update drops, 0 revisions of existing leaves
  0 CEF route update drops due to version mis-match
  Resolution Timer: 15s
  0 prefixes modified in place
  0 deleted stale prefixes
  0 prefixes with label imposition, 0 prefixes with label information
  0 LISP EID prefixes, 0 merged, via 0 rlocs
 28 next hops
  1 incomplete next hop

 0 PD backwalks on LDIs with backup path


VRF: default

_____________
Table ID: 0xe0000000. Total number of entries: 0
OOR state: GREEN. Number of OOR attributes: 0
```

Associated Commands

- show cef bgp-attribute

Proactive Address Resolution Protocol and Neighbor Discovery

When CEF installs a route for which there is no layer 2 adjacency information, CEF creates an incomplete layer 3 next-hop and programs it on the hardware. Because of this incomplete programming, the first packet will be forwarded to the software forwarding path. The software forwarding in turn strips off the layer 2 header from the packet and forwards it to ARP (Address Resolution Protocol) or ND (Neighbor Discovery) in order to resolve the layer 2 adjacency information. In such a packet, if there is feature specific information present in the layer 2 header, the software forwarding path fails to strip off the layer 2 header completely and thus ARP or ND is unable to resolve the missing layer 2 adjacency information and thereby this results in traffic being dropped.

Proactive ARP and ND feature solves the above problem by ensuring that CEF proactively triggers ARP or ND in order to resolve the missing layer 2 adjacency information, retrying every 15 seconds until the next-hop information is resolved. Thus, when you configure a static route which has an incomplete next-hop information, this feature automatically triggers ARP or ND resolution.

Configuration

```
/* Enter the configuration mode and configure Proactive ARP/ND */
Router# configure
Router(config)# cef proactive-arp-nd enable
Router(config)# commit
```

Running Config

```
Show running-config
cef proactive-arp-nd enable
end
```

Enhance FIB debuggability

To enhance the debuggability of the FIB objects, information like allocated object-id of table can be displayed.

FIB debuggability is a new enhancement that displays

- all object-ids allocated within the default table
- all object-ids allocated within the encap-id table
- specific object-id within the default or encap-id table.

Table 9: Feature History Table

Feature name	Release Information	Feature Description
Enhance FIB debuggability	Release 25.3.1	You can now enhance the debuggability of FIB objects by requesting information like allocated object-ids within the required table. This feature introduces these changes: CLI: • **show cef global object-id summary** • **show cef global object-id table**

With the introduction of two new CLI commands, user can display the object-ids of the required table and also any specific object-ids allocated within the table.