Configuring IP-in-IP Tunnels

This chapter provides conceptual and configuration information for IP-in-IP tunnels.

IP-in-IP Tunnels

Table 1: Feature History Table

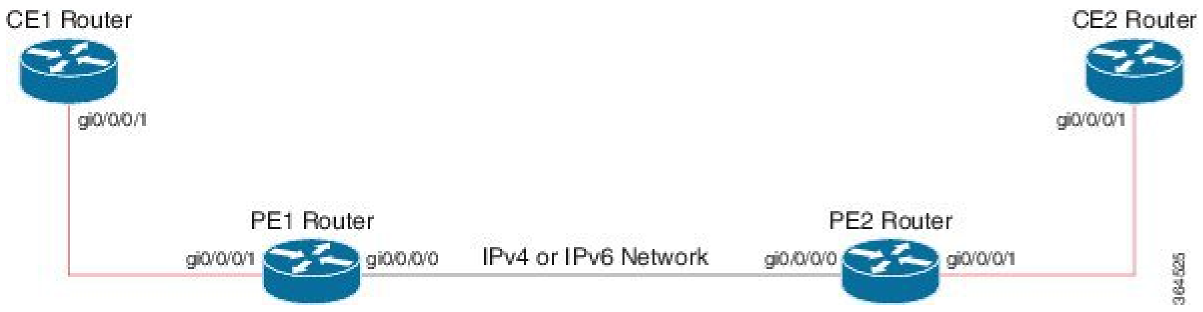
Feature Name	Release Information	Feature Description
Descapsulating IPv4 packets with IPv6 Outer Header	Release 7.5.4	With this release, decapsulation of IPv4 and IPv6 packets with IPv6 outer headers are supported. This decapsulation is supported only with **tunnel source direct** option and not with **tunnel source** with IPv6 address. This feature helps the administrators to take advantage of the benefits of IPv6, such as improved routing and security, without having to upgrade their entire network to IPv6.

Tunneling provides a mechanism to transport packets of one protocol within another protocol. IP-in-IP tunneling refers to the encapsulation and decapsulation of an IP packet as a payload in another IP packet. Cisco NCS 5500 Routers support IP-in-IP decapsulation with all possible combinations of IPv4 and IPv6; that is, IPv4 over IPv4, IPv6 over IPv4, IPv4 over IPv6, and IPv6 over IPv6. For example, an IPv4 over IPv6 refers to an IPv4 packet as a payload encapsulated within an IPv6 packet and routed across an IPv6 network to reach the destination IPv4 network, where it is decapsulated.

IP-in-IP tunneling can be used to connect remote networks securely or provide virtual private network (VPN) services.

The following example provides configurations for an IPv4 or IPv6 tunnel, with the transport VRF as the default VRF for the following simplified network topology.

Figure 1: IP-in-IP Tunnel Network Topology



Configuration Example for IPv4 Tunnel

PE1 Router Configuration

```
interface GigabitEthernet0/0/0/0
 !! Link between PE1-PE2
 ipv4 address 100.1.1.1/64
!
interface GigabitEthernet0/0/0/1
 !! Link between CE1-PE1
ipv4 address 20.1.1.1/24
 ipv6 address 20::1/64
!
interface tunnel-ip 1
ipv4 address 10.1.1.1/24
 ipv6 address 10::1/64
 tunnel mode ipv4
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100.1.1.2
 !

router static
address-family ipv4 unicast
   30.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   30::0/64 tunnel-ip1
  !
 !
!
```

PE2 Router Configuration

```
interface GigabitEthernet0/0/0/0
!! Link between PE1-PE2
 ipv4 address 100.1.1.2/64
!
interface GigabitEthernet0/0/0/1
 !! Link between PE2-CE2
ipv4 address 30.1.1.1/24
 ipv6 address 30::1/64
!
interface tunnel-ip 1
ipv4 address 10.1.1.2/24
 ipv6 address 10::2/64
 tunnel mode ipv4
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100.1.1.1
 !

router static
address-family ipv4 unicast
   20.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   20::0/64 tunnel-ip1
  !
 !
!
```

CE1 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE1-PE1
 ipv4 address 20.1.1.2 255.255.255.0
 ipv6 address 20::2/64
!
router static
 address-family ipv4 unicast
  30.1.1.0/24 20.1.1.1
 address-family ipv6 unicast
  30::0/64 20::1
 !
!
```

CE2 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE2-PE2
 ipv4 address 30.1.1.2 255.255.255.0
 ipv6 address 30::2/64
!
router static
 address-family ipv4 unicast
  20.1.1.0/24 30.1.1.1
address-family ipv6 unicast
  20::0/64 30::1
 !
!
```

Configuration Example for IPv6 Tunnel

PE1 Router Configuration

```
interface GigabitEthernet0/0/0/0
 !! Link between PE1-PE2
 ipv6 address 100::1/64
!
interface GigabitEthernet0/0/0/1
 !! Link between CE1-PE1
 vrf RED
 ipv4 address 20.1.1.1/24
 ipv6 address 20::1/64
!
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.1/24
 ipv6 address 10::1/64
 tunnel mode ipv6
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100::2
 !
vrf RED
 address-family ipv6 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
router static
vrf RED
  address-family ipv4 unicast
   30.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   30::0/64 tunnel-ip1
  !
 !
!
```

PE2 Router Configuration

```
interface GigabitEthernet0/0/0/0
!! Link between PE1-PE2
 ipv6 address 100::2/64
!
interface GigabitEthernet0/0/0/1
 !! Link between PE2-CE2
 vrf RED
 ipv4 address 30.1.1.1/24
 ipv6 address 30::1/64
!
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.2/24
 ipv6 address 10::2/64
 tunnel mode ipv6
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100::1
 !
vrf RED
 address-family ipv6 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
router static
vrf RED
  address-family ipv4 unicast
   20.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   20::0/64 tunnel-ip1
  !
 !
!
```

CE1 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE1-PE1
 ipv4 address 20.1.1.2 255.255.255.0
 ipv6 address 20::2/64
!
router static
 address-family ipv4 unicast
  30.1.1.0/24 20.1.1.1
 address-family ipv6 unicast
  30::0/64 20::1
 !
!
```

CE2 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE2-PE2
 ipv4 address 30.1.1.2 255.255.255.0
 ipv6 address 30::2/64
!
router static
 address-family ipv4 unicast
  20.1.1.0/24 30.1.1.1
address-family ipv6 unicast
  20::0/64 30::1
 !
!
```

- IP-in-IP Decapsulation, on page 4

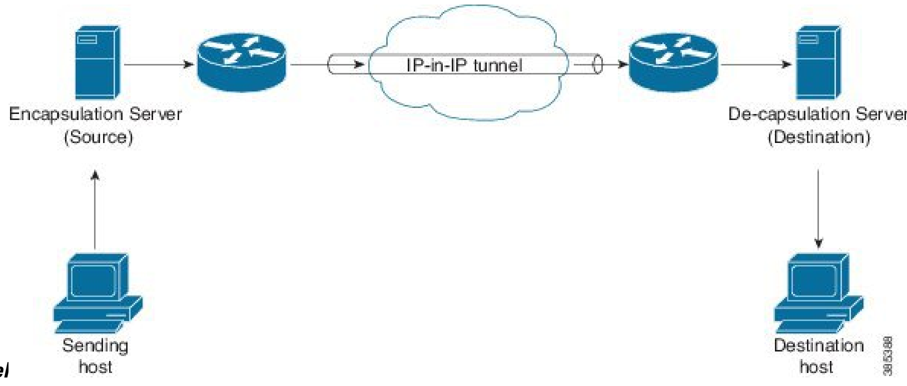
IP-in-IP Decapsulation

Encapsulation of datagrams in a network is done for multiple reasons, such as when a source server wants to influence the route that a packet takes to reach the destination host. The source server is also known as the encapsulation server.

IP-in-IP encapsulation involves the insertion of an outer IP header over the existing IP header. The source and destination address in the outer IP header point to the endpoints of the IP-in-IP tunnel. The stack of IP headers is used to direct the packet over a predetermined path to the destination, provided the network administrator knows the loopback addresses of the routers transporting the packet. This tunneling mechanism can be used for determining availability and latency for most network architectures. It is to be noted that the entire path from source to the destination does not have to be included in the headers, but a segment of the network can be chosen for directing the packets.

The following illustration describes the basic IP-in-IP encapsulation and decapsulation model.

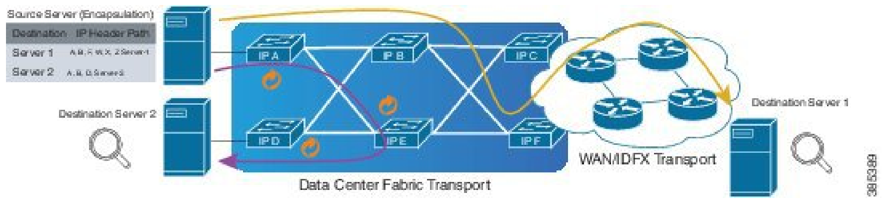
Figure 2: Basic Encapsulation and Decapsulation with an IP-in-IP Tunnel


IP-in-IP tunnel
Encapsulation Server (Source)
De-capsulation Server (Destination)
Sending host
Destination host
385388

Use Case: Configure IP-in-IP Decapsulation

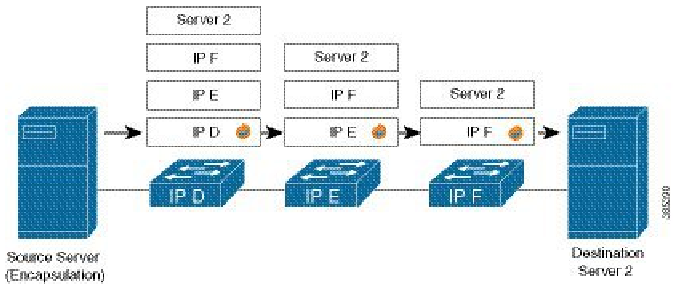
The following topology describes a use case where IP-in-IP encapsulation and decapsulation are used for different segments of the network from source to destination. The IP-in-IP tunnel consists of multiple routers that are used to decapsulate and direct the packet through the data center fabric network.

Figure 3: IP-in-IP Decapsulation Through a Data Center Network


Source Server (Encapsulation)
Destination IP Header Path
Server 1 A, B, F, W, X, Z Server1
Server 2 A, B, D Server2
Destination Server 2
IP A
IP B
IP C
IP D
IP E
IP F
Data Center Fabric Transport
WAN/IDFX Transport
Destination Server 1
385389

The following illustration shows how the stacked IPv4 headers are de-capsulated as they traverse through the de-capsulating routers.

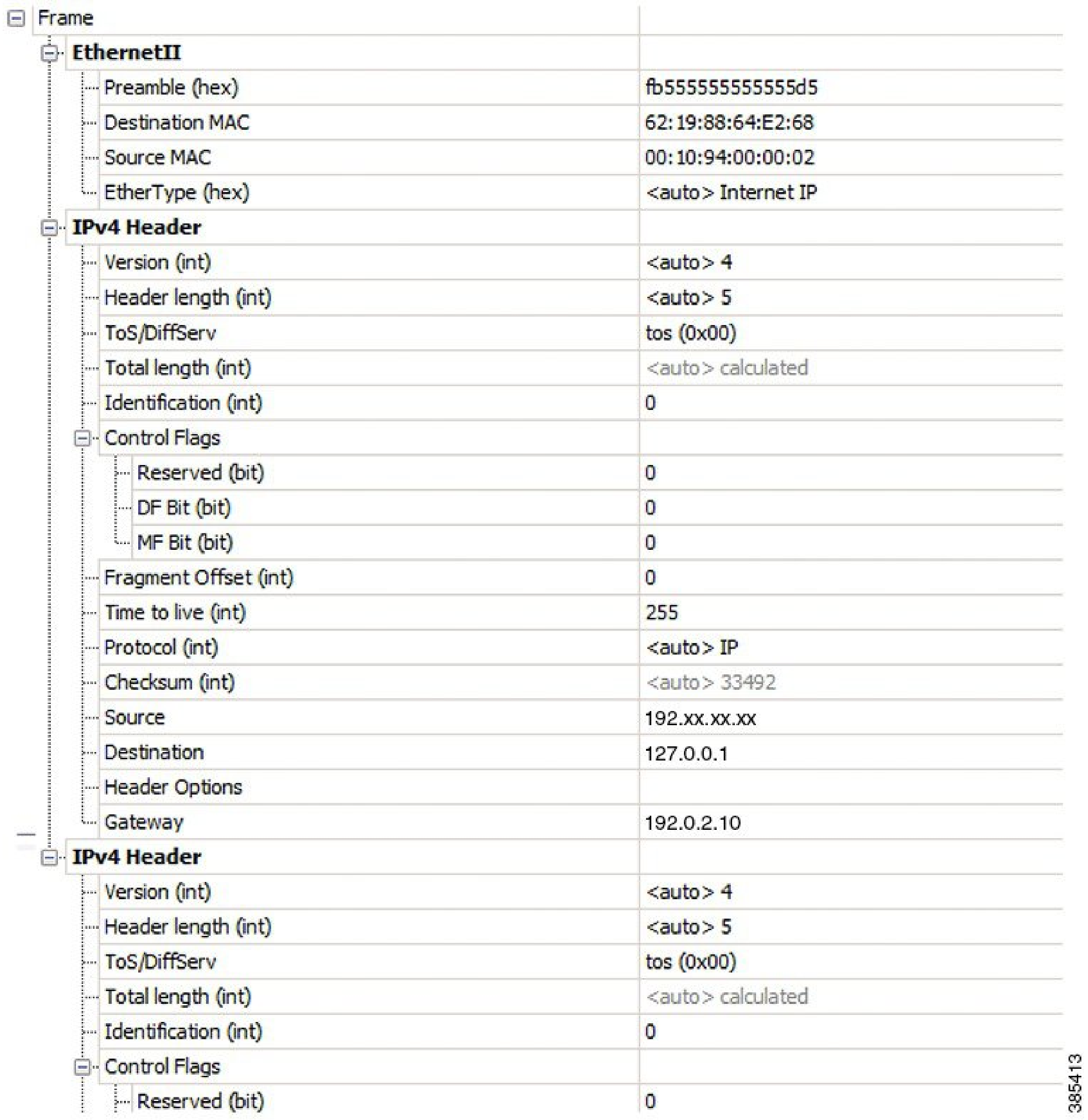
Figure 4: IP Header Decapsulation



Stacked IP Header in an Encapsulated Packet

The encapsulated packet has an outer IPv4 header that is stacked over the original IPv4 header, as shown in the following illustration.

Encapsulated Packet



Frame	
EthernetII	
Preamble (hex)	fb555555555555d5
Destination MAC	62:19:88:64:E2:68
Source MAC	00:10:94:00:00:02
EtherType (hex)	<auto> Internet IP
IPv4 Header	
Version (int)	<auto> 4
Header length (int)	<auto> 5
ToS/DiffServ	tos (0x00)
Total length (int)	<auto> calculated
Identification (int)	0
Control Flags	
Reserved (bit)	0
DF Bit (bit)	0
MF Bit (bit)	0
Fragment Offset (int)	0
Time to live (int)	255
Protocol (int)	<auto> IP
Checksum (int)	<auto> 33492
Source	192.xx.xx.xx
Destination	127.0.0.1
Header Options	
Gateway	192.0.2.10
IPv4 Header	
Version (int)	<auto> 4
Header length (int)	<auto> 5
ToS/DiffServ	tos (0x00)
Total length (int)	<auto> calculated
Identification (int)	0
Control Flags	
Reserved (bit)	0

Configuration

You can use the following sample configuration on the routers to decapsulate the packet as it traverses the IP-in-IP tunnel:

```
RP/0/RP0/CPU0:router(config)# interface tunnel-ip 10
RP/0/RP0/CPU0:router(config-if)# tunnel mode ipv4 decap
RP/0/RP0/CPU0:router(config-if)# tunnel source loopback 0
RP/0/RP0/CPU0:router(config-if)# tunnel destination 10.10.1.2/32
```

- **tunnel-ip**: configures an IP-in-IP tunnel interface.

- **ipv4 unnumbered loopback address**: enables ipv4 packet processing without an explicit address, except for loopback address.
- **tunnel mode ipv4 decap**: enables IP-in-IP decapsulation.
- **tunnel source**: indicates the source address for the IP-in-IP decap tunnel w.r.t the router interface.
- **tunnel destination**: indicates the destination address for the IP-in-IP decap tunnel w.r.t the router interface.

Running Configuration

```
RP/0/RP0/CPU0:router# show running-config interface tunnel-ip 10
...
interface tunnel-ip 10
tunnel mode ipv4 decap
tunnel source Loopback 0
tunnel destination 10.10.1.2/32
```

This completes the configuration of IP-in-IP decapsulation.

Decapsulation Using Tunnel Source Direct

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Decapsulating Using Tunnel Source Direct	Release 7.5.3	Tunnel source direct allows you to decapsulate the tunnels on any L3 interface on the router. You can use the **tunnel source direct** configuration command to choose the specific IP Equal-Cost Multipath (ECMP) links for troubleshooting, when there are multiple IP links between two devices.

To debug faults in various large networks, you may have to capture and analyze the network traffic at a packet level. In datacenter networks, administrators face problems with the volume of traffic and diversity of faults. To troubleshoot faults in a timely manner, DCN administrators must identify affected packets inside large volumes of traffic. They must track them across multiple network components, analyze traffic traces for fault patterns, and test or confirm potential causes.

In some networks, IP-in-IP decapsulation is currently used in network management, to verify ECMP availability and to measure the latency of each path within a datacenter.

The Network Management System (NMS) sends IP-in-IP (IPv4 or IPv6) packets with a stack (multiple) of predefined IPv4 or IPv6 headers (device IP addresses). The destination device at each hop removes the outside header, performs a lookup on the next header, and forwards the packets if a route exists.

Using the **tunnel source direct** command, you can choose the specific IP Equal-Cost Multipath (ECMP) links for troubleshooting, when there are multiple IP links between two devices.

Tip You can programmatically configure and manage the Ethernet interfaces using `openconfig-ethernet-if.yang` and `openconfig-interfaces.yang` OpenConfig data models. To get started with using data models, see the *Programmability Configuration Guide for Cisco NCS 5500 Series Routers*.

Guidelines and Limitations

The following guidelines are applicable to this feature.

- The **tunnel source direct** command is only compatible with 'tunnel mode decap' for IP-in-IP decapsulation.
- The source-direct tunnel is always operationally `up` unless it is administratively shut down. The directly connected interfaces are identified using the **show ip route direct** command.
- All Layer 3 interfaces that are configured on the device are supported.
- Platform can accept and program only certain number of IP addresses. The number of IP addresses depends on the make of the platform linecard (LC). Each LC can have different number of Network Processor (NP) slices and interfaces.
- Only one source-direct tunnel per address-family is supported for configuration.
- Source-direct and regular decap tunnels can't co-exist for a specific address-family. Any configuration that attempts to enable both is automatically rejected, and an error message is displayed to indicate the conflict.
- Inline modification of an existing regular decap tunnel (**tunnel source** *interface* | *IP address*) to a source-direct tunnel (**tunnel source direct**), or changing a source-direct tunnel to a regular decap tunnel, is not supported. Commit-replace may fail if the same tunnel-id is used as part of the commit-replace operation. You must delete the tunnel and recreate it.

The following functionalities are not supported for the **tunnel source direct** option.

- GRE tunneling mode.
- VRF (only default VRF is supported).
- ACL and QoS on the tunnels.
- Tunnel encapsulation.
- Tunnel NetIO DLL: Decapsulation is not supported if the packet is punted to slow path.

Configure Decapsulation Using Tunnel Source Direct

Configuration

The **tunnel source direct** configures IP-in-IP tunnel decapsulation on any directly connected IP addresses. This option is now supported only when the IP-in-IP decapsulation is used to source route the packets through the network.

This example shows how to configure IP-in-IP tunnel decapsulation on directly connected IP addresses:

```
Router# configure terminal
Router(config)#interface Tunnel4
```

```
  Router(config)#tunnel mode ipv4 decap
  Router(config)#tunnel source direct
  Router(config)#no shutdown
```

This example shows how to configure IP-in-IP tunnel decapsulation on IPv6 enabled networks:

```
Router# configure terminal
Router(config)#interface Tunnel6
  Router(config)#tunnel mode ipv6 decap
  Router(config)#tunnel source direct
  Router(config)#no shutdown
```

Verifying the Configuration

The following example shows how to verify IP-in-IP tunnel decapsulation with **tunnel source direct** option:

```
Router#show running-config interface tunnel 1
interface Tunnel1
  tunnel mode ipv6ipv6 decapsulate-any
  tunnel source direct
  no shutdown

Router#show interface tunnel 1
Tunnel1 is up    Admin State: up
MTU 1460 bytes, BW 9 Kbit
Tunnel protocol/transport IPv6/DECAPANY/IPv6
Tunnel source - direct
Tx    0 packets output, 0 bytes    Rx    0 packets input, 0 bytes
```