Configuring IP Tunnels

Information About IP Tunnels

IP tunnels can encapsulate a same-layer or higher layer protocol and transport the result over IP through a tunnel created between two devices.

IP Tunnel Overview

IP tunnels consists of the following three main components:

- Passenger protocol—The protocol that needs to be encapsulated. IPv4 is an example of a passenger protocol.
- Carrier protocol—The protocol that is used to encapsulate the passenger protocol. Cisco NX-OS supports GRE as a carrier protocol.
- Transport protocol—The protocol that is used to carry the encapsulated protocol. IPv4 is an example of a transport protocol. An IP tunnel takes a passenger protocol, such as IPv4, and encapsulates that protocol within a carrier protocol, such as GRE. The device then transmits this carrier protocol over a transport protocol, such as IPv4.

You configure a tunnel interface with matching characteristics on each end of the tunnel.

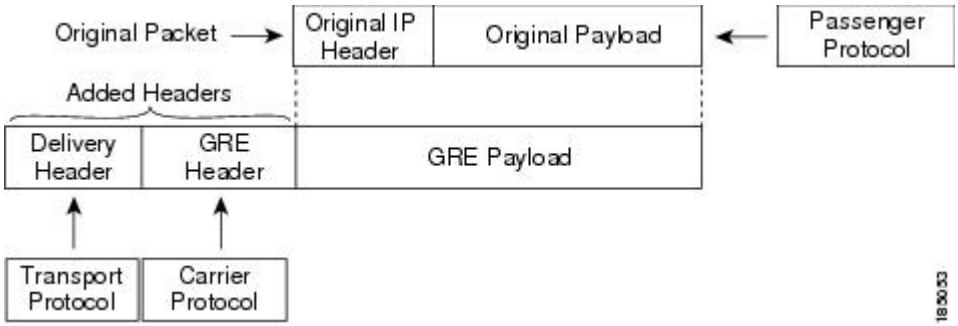
You must enable the tunnel feature before you can configure it. The system automatically takes a checkpoint prior to disabling the feature, and you can roll back to this checkpoint. See the Cisco Nexus 9000 Series NX-OS System Management Configuration Guide for information about rollbacks and checkpoints.

GRE Tunnels

You can use generic routing encapsulation (GRE) as the carrier protocol for a variety of passenger protocols.

The following figure shows the IP tunnel components for a GRE tunnel. The original passenger protocol packet becomes the GRE payload and the device adds a GRE header to the packet. The device then adds the transport protocol header to the packet and transmits it.

Figure 1: GRE PDU



Point-to-Point IP-in-IP Tunnel Encapsulation and Decapsulation

The point-to-point IP-in-IP encapsulation and decapsulation is a type of tunnel that you can create to send encapsulated packets from a source tunnel interface to a destination tunnel interface. This type of tunnel will carry both inbound and outbound traffic.

Note The selection of GRE or IP-in-IP tunnel destination based on the PBR policy is not supported.

Note IP-in-IP tunnel encapsulation and decapsulation is not supported on Cisco Nexus 9500 Series switches with N9K-X9636C-R, N9K-X9636Q-R, N9K-X9636C-RX line cards.

Note IP-in-IP tunnel encapsulation and decapsulation is not supported on a vPC setup on Cisco Nexus 9300-EX, 9300-FX, 9300-GX and Nexus 9500 platform switches.

Multi-Point IP-in-IP Tunnel Decapsulation

The multi-point IP-in-IP decapsulate-any is a type of tunnel that you can create to decapsulate packets from any number of IP-in-IP tunnels to one tunnel interface. This tunnel will not carry any outbound traffic. However, any number of remote tunnel endpoints can use a tunnel configured this way as their destination.

Path MTU Discovery

Path maximum transmission unit (MTU) discovery (PMTUD) prevents fragmentation in the path between two endpoints by dynamically determining the lowest MTU along the path from the packet's source to its destination. PMTUD reduces the send MTU value for the connection if the interface receives information that the packet would require fragmentation.

When you enable PMTUD, the interface sets the Don't Fragment (DF) bit on all packets that traverse the tunnel. If a packet that enters the tunnel encounters a link with a smaller MTU than the MTU value for the packet, the remote link drops the packet and sends an ICMP message back to the sender of the packet. This message indicates that fragmentation was required (but not permitted) and provides the MTU of the link that dropped the packet.

Note PMTUD on a tunnel interface requires that the tunnel endpoint can receive ICMP messages generated by devices in the path of the tunnel. Check that ICMP messages can be received before using PMTUD over firewall connections.

High Availability

IP tunnels support stateful restarts. A stateful restart occurs on a supervisor switchover. After the switchover, Cisco NX-OS applies the runtime configuration after the switchover.

Prerequisites for IP Tunnels

IP tunnels have the following prerequisites:

- You must be familiar with TCP/IP fundamentals to configure IP tunnels.
- You are logged on to the switch.
- You must enable the tunneling feature in a device before you can configure and enable any IP tunnels.

Guidelines and Limitations

IP tunnels have the following configuration guidelines and limitations:

- Guidelines for **source-direct** and **ipv6ipv6-decapsulate-any** options for tunnels:
  - The **source-direct** command is supported on the Cisco Nexus 9500 platform switches with the Application Spine Engine (ASE) and the Leaf Spine Engine (LSE).

    Cisco Nexus 9500 platform switches with the Network Forwarding Engine (NFE) do not support the **tunnel source direct** command.

    The **tunnel source direct** command with the **tunnel mode ipv6ipv6 decapsulate-any** command on the Cisco Nexus 9500 platform switches is only supported in the MPLS heavy routing template.
  - The IP tunnel supports the **tunnel source** CLI command with interface, IPv4 address, IPv6 address, or IPv4 prefix. You can configure IP-in-IP tunnel decapsulation on directly connected IP addresses

(for example, physical interface, port-channel, loopback, and SVI) using the new **tunnel source direct** CLI command. You can select the IP ECMP links when there are multiple IP links between the two switches. A single tunnel interface can decapsulate the tunneled packets whose outer destination IP is any of the IPv4 or IPv6 address that is locally configured and it is operationally *Up* in the switch.

- Currently, **tunnel mode ipip decapsulate-any** is supported for decapsulating IPv4 payload over IPv4 transport (IPv4inIPv4 packets). **tunnel mode ipv6ipv6 decapsulate-any** command is introduced to support IPv6 payload over IPv6 transport (IPv6inIPv6 packets).
- The **tunnel source direct** and **tunnel mode ipv6ipv6 decapsulate-any** CLI commands are not supported on Cisco Nexus 9500 platform switches with the Network Formation Engine (NFE).
- The **tunnel source direct** CLI command is supported only when an administrator uses the IP-in-IP decapsulation to source route the packets through the network. 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** CLI command.
- The **tunnel source direct** CLI command is supported only on decapsulate-any tunnel modes, for example, **tunnel mode ipip decapsulate-any** and **tunnel mode ipv6ipv6 decapsulate-any** .
- Auto-recovery for source-direct is not supported.
- For ipv6ipv6 decapsulate-any, inter-VRF is not supported. The tunnel interface VRF (iVRF) and tunnel transport or forwarding VRF (fVRF) must be the same. Only one decapsulate-any tunnel (irrespective of VRF) can be present in Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches; Cisco Nexus 9500 platform modular switches with EX and FX line cards.
- To enable IPv6 on ipv6ipv6 decap-any tunnel interface, configure a valid IPv6 address or configure **ipv6 address use-link-local-only** under the tunnel interface.

- See the following hardware limitations on the maximum sources that can be accommodated on a source direct tunnel and the related behavior:
  - Source direct tunnel is now supported for Cisco Nexus 9000 Series switches with Network Forwarding Engine (NFE), Application Spine Engine (ASE), and Leaf Spine Engine (LSE). Most of the limitations are only in case of scaled SIP (number of total IP/IPv6 addresses on the interfaces (L3, sub-interface, PC, PC-sub interfaces, loopback, SVI, and any secondary IP/IPv6 addresses.)

    See the following sample use cases.

    - Use Case 1: Non-deterministic behavior of which SIP gets installed if the number of IP/IPv6 interface scale is more.

      Both the switches have 512 entries for tunnel SIP. With tunnel source, direct any IP or IPv6 address w.r.t **ipip or ipv6ipv6 decap any** with tunnel source gets installed in the above table.

      The insertion of these entries is on a first come first serve basis without any CLI command to control which interface IP addresses get installed. If the system has more number of IP/IPv6 interfaces to be installed, the behavior is non-deterministic (The behavior can change across reload with interface flaps.)

    - Use Case 2: The scale numbers are different in both switches and each has its own advantages and disadvantages.

IPv4 individual scale can be more (up to 512) in case of switches with NFE but it is shared with IPv6. In the switches with ASE and LSE, the IPv4 individual scale can be 256 but it is not shared with IPv6.

Whenever the tunnel decap table gets filled, the TABLE_FULL error is displayed. If some entry gets deleted after the table gets full, the table full error is cleared.

Table 1: Scale Numbers

Commands	Switches with NFE: Table size 512, v4 takes 1 entry, v6 takes 4 entries	Switches with ASE and LSE: Table size 512, v4 takes 1 entry, v6 takes 2 entries (paired index)
IPIP decap any with tunnel source direct	Shared between v4 and v6, v6 takes 4 entries v4 + 4 *v6 =512 Maximum entries can be 512 with no v6 entries	Dedicated 256
IPv6IPv6 decap any with tunnel source direct	Shared between v4 and v6, v6 takes 4 entries v4 + 4 *v6 =512 Maximum entries can be 128 with no v4 entries	Dedicated 128

- Use Case 3: Auto-recovery is not supported.

  If any entry does not get installed in the hardware due to exhaustion of above table, removal of an already installed IP/IPv6 from interfaces does not automatically trigger the addition of the failed SIP in the table though the table has space now. You need to flap the tunnel interface or IP interface to get them installed.

  However, if an entry does not get installed in the hardware due to a duplicate entry (if there was already a **decap-any** with one source present and now the **source direct tunnel** CLI command is configured, there is a duplicate entry for the prior source configured) that was taken care of by removing the entry only when both the tunnels get deleted.

- For Cisco Nexus 9000 Series switches with Network Forwarding Engine (NFE) and Application Spine Engine (ASE), the syslog is different as the dedicated IPv4 and IPv6 decap entries are carved in the syslog. If the **tunnel-decap-table** is full, the user gets a syslog as follows:

```
2017 Apr 6 12:18:04 switch %$ VDC-1 %$ %IPFIB-2-FIB_HW_IPV4_TUNNEL_DECAP_TABLE_FULL:
 IPv4 tunnel decap hardware table full.
IP tunnel decapsulation may not work for some GRE/IPinIP traffic

2017 Apr 6 12:18:11 switch %$ VDC-1 %$ %IPFIB-2-FIB_HW_IPV6_TUNNEL_DECAP_TABLE_FULL:
 IPv6 tunnel decap hardware table full.
IP tunnel decapsulation may not work for some GRE/IPinIP traffic
```

  If the table is full and once some entry gets deleted from the table (due to an interface being operationally down or removal of IP address), the clear syslog for the table is displayed. Note that the deletion of the tunnel removes all the entries that are added as part of that tunnel.

```
2017 Apr 5 13:29:25 switch %$ VDC-1 %$
%IPFIB-2-FIB_HW_IPV4_TUNNEL_DECAP_TABLE_FULL_CLRD: IPv4 tunnel decap hardware table
 full exception cleared

2017 Apr 4 19:41:22 switch %$ VDC-1 %$
%IPFIB-2-FIB_HW_IPV6_TUNNEL_DECAP_TABLE_FULL_CLRD: IPv6 tunnel decap hardware table
 full exception cleared
```

- IP-in-IP tunnel decapsulation is supported on IPv6 enabled networks.

```
!
interface tunnel 1
  ipv6 address use-link-local-only             <<< enable IPv6
  tunnel mode ipv6ipv6 decapsulate-any
  tunnel source direct
  description IPinIP Decapsulation Interface
  mtu 1476
  no shutdown
```

- The **show** commands with the **internal** keyword are not supported.
- Cisco NX-OS supports only the following protocols:
  - IPv4 passenger protocol.
  - GRE carrier protocol.
- IP tunnels do not support access control lists (ACLs) or QoS policies.
- Cisco NX-OS supports the GRE header defined in IETF RFC 2784. Cisco NX-OS does not support tunnel keys and other options from IETF RFC 1701.
- Cisco NX-OS does not support GRE tunnel keepalives.
- All unicast routing protocols are supported by IP tunnels.
- The IP tunnel interface cannot be configured to be a span source or destination.
- IP tunnels do not support PIM or other Multicast features and protocols.
- The selection of GRE or IP-in-IP tunnel destination based on the PBR policy is not supported.
- IP tunnels are supported only in the default **system routing** mode and not in other modes.
- When configuring a tunnel interface to **ipip mode** , the maximum mtu value is9196.

  When downgrading from NX-OS 9.2(1)to an earlier release, with a tunnel interface in **ipip mode** that has an mtu value of 9196, the mtu configuration is lost as a result of the downgrade operation. As a best practice, adjust the mtu value to 9192 before commencing the downgrade to avoid losing the mtu configuration.
- When configuring a tunnel interface to **ipip mode**, the default mtu value is1480.

  When downgrading from NX-OS 9.2(1) or later release to an earlier release, with a tunnel interface in **ipip mode** with no explicit mtu configuration, the mtu value changes as a result of the downgrade operation from 1480 to 1476. As a best practice, adjust the mtu value to 1476 before commencing the downgrade to avoid any changes to the mtu value.

When upgrading to NX-OS 9.2(1) or later release,, with a tunnel interface in **ipip mode** with no explicit mtu configuration, the mtu value changes as a result of the upgrade operation from 1476 to 1480. As a best practice, adjust the mtu value to 1480 before commencing the upgrade to avoid any changes to the mtu value.

- On Cisco Nexus 9200 Series switches, GRE packets that are received on an IP-in-IP tunnel are not dropped as expected and are instead forwarded to the packet destination.
- Tx packets originating from the switch, such as control pkts, are not included in Tx statistics.
- Tunnel destinations that are reachable over another tunnel are not supported.
- The consistency checker is not supported for routes over a tunnel.
- Non-IP routing protocols, such as isis, are not supported over IP-in-IP tunnels.
- RFC5549 is not be supported over tunnels.
- BGP adjacency over tunnel is not supported in a scenario where the tunnel interface and tunnel source are in same VRF (example: VRF-A) and tunnel destination is reachable with route-leak from opposite end (example: via VRF-B)
- The feature **feature tunnel** on the Cisco Nexus 9000 switches cannot co-exist with the VXLAN feature, **feature nv overlay**.
- Cisco Nexus 9200, 9300-EX, 9300-FX, 9300-FX2 series switches and Cisco Nexus 9500 platform switches with 9700-EX/FX line cards may not have multiple tunnel interfaces in a single VRF that are sourced from or destined to the same IP address. For example, a device may not have tunnel 0 and tunnel 1 interfaces in the default VRF that are sourced from the same IP address or interface.
- Cisco Nexus 9300-EX, 9300-FX, 9300-GX and Nexus 9500 platform switches in vPC can act as GRE Tunnel endpoints for their respective tunnels. However, the tunnel destination can not be through a vPC.

Default Settings

The following table lists the default settings for IP tunnel parameters.

Table 2: Default IP Tunnel Parameters

Parameters	Default
Path MTU discovery age timer	10 minutes
Path MTU discovery minimum MTU	64
Tunnel feature	Disabled

Configuring IP Tunnels

Note If you are familiar with the Cisco IOS CLI, be aware that the Cisco NX-OS commands for this feature might differ from the Cisco IOS commands that you would use.

Enabling Tunneling

You must enable the tunneling feature before you can configure any IP tunnels.

SUMMARY STEPS

1. **configure terminal**
2. **feature tunnel**
3. **exit**
4. **show feature**
5. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature tunnel**
	Example:
	`switch(config)# feature tunnel`
	`switch(config-if)#`
		Allows the creation of a new tunnel interface.
		To disable the tunnel interface feature, use the **no** form of this command.
		Note The **feature tunnel** command may break the multicast functionality if multicast heavy template is enforced.

Step 3	**exit**
	Example:
	`switch(config-if)# exit`
	`switch#`
		Exits the interface mode and returns to the configuration mode.

Step 4	**show feature** **Example:** `switch(config-if)# show feature`	(Optional) Displays information about the features enabled on the device.

	Command or Action	Purpose
Step 5	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Creating a Tunnel Interface

You can create a tunnel interface and then configure this logical interface for your IP tunnel.

Note Cisco NX-OS supports a maximum of 8 IP tunnels.

Note Use the **no interface tunnel** command to remove the tunnel interface and all associated configuration.

Command	Purpose
no interface tunnel *number* **Example:** `switch(config)# no interface tunnel 1`	Deletes the tunnel interface and the associated configuration.
description *string* **Example:** `switch(config-if)# description GRE tunnel`	Configures a description for the tunnel.
mtu *value* **Example:** `switch(config-if)# mtu 1400`	Sets the MTU of IP packets sent on an interface.
tunnel ttl *value* **Example:** `switch(config-if)# tunnel ttl 100`	Sets the tunnel time-to-live value. The range is from 1 to 255.

Before you begin

You can configure the tunnel source and the tunnel destination in different VRFs. Ensure that you have enabled the tunneling feature.

SUMMARY STEPS

1. **configure terminal**
2. **interface tunnel** *number*

3. **tunnel mode** {**gre ip** | **ipip** {**ip** | **decapsulate-any**}}
4. **tunnel source** {*ip-address* |*interface-name*}
5. **tunnel destination** {*ip-address* |*host-name*}
6. **tunnel use-vrf** *vrf-name*
7. **show interfaces tunnel** *number*
8. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface tunnel** *number*
	Example:
	`switch(config)# interface tunnel 1`
	`switch(config-if)#`
		Creates a new tunnel interface.

Step 3	**tunnel mode** {**gre ip**	**ipip** {**ip**	**decapsulate-any**}}	Sets this tunnel mode to GRE, ipip, or ipip decapsulate-only. The **gre** and **ip** keywords specify that GRE encapsulation over IP will be used. The **ipip** keyword specifies that IP-in-IP encapsulation will be used. The optional **decapsulate-any** keyword terminates IP-in-IP tunnels at one tunnel interface. This keyword creates a tunnel that will not carry any outbound traffic. However, remote tunnel endpoints can use a tunnel configured as their destination.
Step 4	**tunnel source** {*ip-address* |*interface-name*} **Example:** `switch(config-if)# tunnel source ethernet 1/2`	Configures the source address for this IP tunnel. The source can be specified by IP address or logical interface name.
Step 5	**tunnel destination** {*ip-address* |*host-name*} **Example:** `switch(config-if)# tunnel destination 192.0.2.1`	Configures the destination address for this IP tunnel. The destination can be specified by IP address or logical host name.
Step 6	**tunnel use-vrf** *vrf-name* **Example:** `switch(config-if)# tunnel use-vrf blue`	(Optional) Uses the configured VRF to look up the tunnel IP destination address.
Step 7	**show interfaces tunnel** *number* **Example:** `switch# show interfaces tunnel 1`	(Optional) Displays the tunnel interface statistics.

	Command or Action	Purpose
Step 8	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Example

This example shows how to create a tunnel interface

```
switch# configure terminal
switch(config)# interface tunnel 1
switch(config-if)# tunnel source ethenet 1/2
switch(config-if)# tunnel destination 192.0.2.1
switch(config-if)# copy running-config startup-config
```

Creating an IP-in-IP Tunnel with a Netmask

Creating an IP-in-IP tunnel with a netmask allows you to specify a tunnel source subnet and a tunnel destination subnet, and decap the packet if it matches.

- The IP-in-IP decap-any tunnel receives encapsulated packets from any number of IP-in-IP tunnels.
- With the netmask feature, the switch receives packets from IP addresses which comply with the netmasks.

Notes for the netmask feature:

- Routing protocols are not supported on an IP-in-IP tunnel created with a netmask.
- Encap is not supported with the netmask feature; only decap from a set of sources in the same subnet is supported.

SUMMARY STEPS

1. **configure terminal**
2. **interface tunnel** *number*
3. **tunnel mode ipip** [**ip**]
4. **tunnel source** *ip-address* / *mask_length*
5. **tunnel destination** *ip-address* / *mask_length*
6. (Optional) **no shut**
7. **ip address** *ip-prefix*/*length*

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**interface tunnel** *number*
	Example:
	`switch(config)# interface tunnel 5`
	`switch(config-if)#`
		Creates a new tunnel interface.

Step 3	**tunnel mode ipip** [**ip**]	Sets this tunnel mode to ipip. The **ipip** keyword specifies that IP-in-IP encapsulation will be used.
Step 4	**tunnel source** *ip-address* / *mask_length* **Example:** `switch(config-if)# tunnel source 33.1.1.1 255.255.255.0`	Configures the source address for this IP tunnel. The source is specified by IP address and length of mask.
Step 5	**tunnel destination** *ip-address* / *mask_length* **Example:** `switch(config-if)# tunnel destination 33.1.1.2 255.255.255.0`	Configures the destination address for this IP tunnel. The destination is specified by IP address and length of mask.
Step 6	(Optional) **no shut**	Clears the interface.
Step 7	**ip address** *ip-prefix*/*length* **Example:** `switch(config-if)# ip address 50.1.1.1/24`	Configures an IP address for this interface.

Example

The following example shows how to create an IP-in-IP tunnel with a netmask.

```
switch(config)# interface tunnel 10
switch(config-if)# tunnel mode ipip
switch(config-if)# tunnel source 33.1.1.2/24
switch(config-if)# tunnel destination 33.1.1.1/24
switch(config-if)# no shut
switch(config-if)# ip address 10.10.10.10/24
switch(config-if)# end
switch# show interface tunnel 10
Tunnel10 is up
    Admin State: up
    Internet address is 10.10.10.10/24
    MTU 1476 bytes, BW 9 Kbit
    Tunnel protocol/transport IPIP/IP
    Tunnel source 33.1.1.2, destination 33.1.1.1
    Transport protocol is in VRF "default"
    Last clearing of "show interface" counters never
    Tx
    0 packets output, 0 bytes
    Rx
    0 packets input, 0 bytes

switch# show run interface tunnel 10
```

```
!Command: show running-config interface Tunnel10
!Time: Wed Aug 26 13:50:01 2015

version 7.0(3)I2(1)

interface Tunnel10
  ip address 10.10.10.10/24
  tunnel mode ipip ip
  tunnel source 33.1.1.2 255.255.255.0
  tunnel destination 33.1.1.1 255.255.255.0
  no shutdown
```

Configuring a Tunnel Interface

You can set a tunnel interface to GRE tunnel mode, ipip mode, or ipip decapsulate-only mode. GRE mode is the default tunnel mode. .

Beginning with Cisco NX-OS Release 7.0(3)I6(1), the **tunnel source direct** and **tunnel mode ipv6ipv6 decapsulate-any** CLI commands are supported on Cisco Nexus 9000 Series switches.

The **tunnel source direct** and **tunnel mode ipv6ipv6 decapsulate-any** CLI commands are supported on Cisco Nexus 9000 Series switches.

Note The **tunnel source direct** and **tunnel mode ipv6ipv6 decapsulate-any** CLI commands are not supported on Cisco Nexus 9500 platform switches with the Network Forwarding Engine (NFE).

The new CLI **tunnel mode ipv6ipv6 decapsulate-any** command is introduced to support IPv6 payload over IPv6 transport (IPv6inIPv6 packets). You can configure IP-in-IP tunnel decapsulation on directly connected IP addresses (for example, physical interface, port-channel, loopback, and SVI) using the new **tunnel source direct** CLI command.

Before you begin

Ensure that you have enabled the tunneling feature.

SUMMARY STEPS

1. **configure terminal**
2. **interface tunnel** *number*
3. **tunnel mode** {**gre ip** | **ipip** | {**ip** | **decapsulate-any**}}
4. (Optional) **tunnel mode ipv6ipv6 decapsulate-any**
5. **tunnel source direct**
6. **show interfaces tunnel** *number*
7. **mtu** *value*
8. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface tunnel** *number*
	Example:
	`switch(config)# interface tunnel 1`
	`switch(config-if)#`
		Creates a new tunnel interface.

Step 3	**tunnel mode** {**gre ip** | **ipip** | {**ip** | **decapsulate-any**}}	Sets this tunnel mode to GRE, ipip, or ipip decapsulate-only. The **gre** and **ip** keywords specify that GRE encapsulation over IP will be used. The ipip keyword specifies that IP-in-IP encapsulation will be used. The optional **decapsulate-any** keyword terminates IP-in-IP tunnels at one tunnel interface. This keyword creates a tunnel that will not carry any outbound traffic. However, remote tunnel endpoints can use a tunnel configured as their destination.
Step 4	(Optional) **tunnel mode ipv6ipv6 decapsulate-any**	Supports IPv6 payload over IPv6 transport (IPv6inIPv6 packets) . This step is applicable for IPv6 networks only. **Note** This command is not supported on Cisco Nexus 9500 platform switches.
Step 5	**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. **Note** This command is not supported on Cisco Nexus 9500 platform switches with the Network Forwarding Engine (NFE).
Step 6	**show interfaces tunnel** *number* **Example:** `switch(config-if)# show interfaces tunnel 1`	(Optional) Displays the tunnel interface statistics.
Step 7	**mtu** *value*	Sets the maximum transmission unit (MTU) of IP packets sent on an interface. The range is from 64 to 9192 units.

	Command or Action	Purpose
		Note When configuring **tunnel mode ipip**, the range is dependent on NX-OS release: • 64 to 9192 units • 64 to 9196 units
Step 8	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Example

This example shows how to create the tunnel interface to GRE:

```
switch# configure terminal
switch(config)# interface tunnel 1
switch(config-if)# tunnel mode gre ip
switch(config-if)# copy running-config startup-config
```

This example shows how to create an ipip tunnel:

```
switch# configure terminal
switch(config)# interface tunnel 1
switch(config-if)# tunnel mode ipip
switch(config-if)# mtu 1400
switch(config-if)# copy running-config startup-config
switch(config-if)# no shut
```

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

```
switch# configure terminal
switch(config)# interface tunnel 0
switch(config-if)# tunnel mode ipip ip
switch(config-if)# tunnel source direct
switch(config-if)# description IPinIP Decapsulation Interface
switch(config-if)# no shut
```

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

```
!
interface tunnel 1
  ipv6 address use-link-local-only            <<< enable IPv6
  tunnel mode ipv6ipv6 decapsulate-any
  tunnel source direct
  description IPinIP Decapsulation Interface
  mtu 1476
  no shutdown

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

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
Transport protocol is in VRF "default"
Tunnel interface is in VRF "default"
Last clearing of "show interface" counters never
Tx    0 packets output, 0 bytes    Rx    0 packets input, 0 bytes
```

Configuring a GRE Tunnel

You can set a tunnel interface to GRE tunnel mode.

Note Cisco NX-OS supports only the GRE protocol for IPV4 over IPV4.

Before you begin

Ensure that you have enabled the tunneling feature.

SUMMARY STEPS

1. **configure terminal**
2. **interface tunnel** *number*
3. **tunnel mode gre ip**
4. **show interfaces tunnel** *number*
5. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface tunnel** *number*
	Example:
	`switch(config)# interface tunnel 1`
	`switch(config-if)#`
		Creates a new tunnel interface.

Step 3	**tunnel mode gre ip** **Example:** `switch(config-if)# tunnel mode gre ip`	Sets this tunnel mode to GRE.
Step 4	**show interfaces tunnel** *number* **Example:**	(Optional) Displays the tunnel interface statistics.

	Command or Action	Purpose
	`switch(config-if)# show interfaces tunnel 1`	
Step 5	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Enabling Path MTU Discovery

Use the **tunnel path-mtu discovery** command to enable path MTU discovery on a tunnel.

SUMMARY STEPS

1. **tunnel path-mtu-discovery age-timer** *min*
2. **tunnel path-mtu-discovery min-mtu** *bytes*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**tunnel path-mtu-discovery age-timer** *min* **Example:** `switch(config-if)# tunnel path-mtu-discovery age-timer 25`	Enables Path MTU Discovery (PMTUD) on a tunnel interface. • min—Number of minutes. The range is from 10 to 30. The default is 10.
Step 2	**tunnel path-mtu-discovery min-mtu** *bytes* **Example:** `switch(config-if)# tunnel path-mtu-discovery min-mtu 1500`	Enables Path MTU Discovery (PMTUD) on a tunnel interface. • bytes—Minimum MTU recognized. The range is from 64 to 9192. The default is 64.

Assigning VRF Membership to a Tunnel Interface

You can add a tunnel interface to a VRF.

Before you begin

Ensure that you have enabled the tunneling feature.

Assign the IP address for a tunnel interface after you have configured the interface for a VRF.

SUMMARY STEPS

1. **configure terminal**
2. **interface tunnel** *number*
3. **vrf member** *vrf-name*
4. **ip address** *ip-prefix/length*

5. **show vrf** [*vrf-name*] **interface** *interface-type number*
6. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface tunnel** *number*
	Example:
	`switch(config)# interface tunnel 0`
	`switch(config-if)#`
		Enters interface configuration mode.

Step 3	**vrf member** *vrf-name* **Example:** `switch(config-if)# vrf member RemoteOfficeVRF`	Adds this interface to a VRF.
Step 4	**ip address** *ip-prefix/length* **Example:** `switch(config-if)# ip address 192.0.2.1/16`	Configures an IP address for this interface. You must do this step after you assign this interface to a VRF.
Step 5	**show vrf** [*vrf-name*] **interface** *interface-type number* **Example:** `switch(config-vrf)# show vrf Enterprise interface tunnel 0`	(Optional) Displays VRF information.
Step 6	**copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	(Optional) Saves this configuration change.

Example

This example shows how to add a tunnel interface to the VRF:

```
switch# configure terminal
switch(config)# interface tunnel 0
switch(config-if)# vrf member RemoteOfficeVRF
switch(config-if)# ip address 209.0.2.1/16
switch(config-if)# copy running-config startup-config
```

Verifying the IP Tunnel Configuration

To verify the IP tunnel configuration information, perform one of the following tasks:

Command	Purpose
show interface tunnel *number*	Displays the configuration for the tunnel interface (MTU, protocol, transport, and VRF). Displays input and output packets, bytes, and packet rates.
show interface tunnel *number* **brief**	Displays the operational status, IP address, encapsulation type, and MTU of the tunnel interface.
show interface tunnel *number* **counters**	Displays interface counters of input/output packets. **Note** The byte count displayed with the interface counters include the internal header size.
show interface tunnel *number* **description**	Displays the configured description of the tunnel interface.
show interface tunnel *number* **status**	Displays the operational status of the tunnel interface.
show interface tunnel *number* **status err-disabled**	Displays the error disabled status of the tunnel interface.

Configuration Examples for IP Tunneling

The following example shows a simple GRE tunnel. Ethernet 1/2 is the tunnel source for router A and the tunnel destination for router B. Ethernet interface 2/1 is the tunnel source for router B and the tunnel destination for router A.

Router A:

```
feature tunnel
interface tunnel 0
ip address 209.165.20.2/8
tunnel source ethernet 1/2
tunnel destination 192.0.2.2
tunnel mode gre ip
tunnel path-mtu-discovery 25 1500

interface ethernet 1/2
ip address 192.0.2.55/8
```

Router B:

```
feature tunnel
interface tunnel 0
ip address 209.165.20.1/8
tunnel source ethernet 2/1
tunnel destination 192.0.2.55
tunnel mode gre ip

interface ethernet 2/1
ip address 192.0.2.2/8
```

Related Documents

Related Topic	Document Title
IP Tunnel commands	*Cisco Nexus 9000 Series NX-OS Interfaces Command Reference*