Interface and Hardware Component Configuration Guide, Cisco IOS XE Release 2

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Configuring Physical Interfaces

The Cisco ASR 1000 Series Aggregation Services Routers support many different types of physical (hardware) interfaces such as Gigabit Ethernet, Packet over SONET (POS), and serial shared port adapter (SPA) interfaces. For hardware technical descriptions and information about installing interfaces, refer to the hardware installation and configuration publication for your product.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Configuration Information

- For information about using the Gigabit Ethernet Management Ethernet interface, see the "Using the Management Ethernet Interface" chapter of the *Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide* at:

http://www.cisco.com/en/US/docs/routers/asr1000/configuration/guide/chassis/asrswcfg.html

- For information about configuring and troubleshooting SPA interface processors (SIPs) and SPAs that are supported on a Cisco ASR 1000 Series Aggregation Services Router, see the *Cisco ASR 1000 Series Aggregation Services Routers SIP and SPA Software Configuration Guide* at:

http://cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRspasw.html

Command Reference Information

- Complete descriptions of the commands used to configure interfaces are included in the *Cisco IOS Interface and Hardware Component Command Reference* at:

http://www.cisco.com/en/US/docs/ios/interface/command/reference/ir_book.html

- For information about other Cisco IOS XE commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases* , at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html .

Configuring Virtual Interfaces

Virtual interfaces are software-based interfaces that you create in the memory of the networking device using Cisco IOS XE commands. Virtual interfaces do not have a hardware component such as the RJ-45 female port on a 100BASE-T Fast Ethernet network interface card. This module describes the four common types of virtual, or logical, interfaces that can be configured using Cisco IOS XE software:

- Loopback interfaces
- Null interfaces
- Subinterfaces
- Tunnel interfaces

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring Virtual Interfaces

Before virtual interfaces can be used in your network, you must have some physical (hardware) interfaces configured and you must be able to communicate between the networking devices on which you wish to use virtual interfaces.

Information About Configuring Virtual Interfaces

Virtual Interfaces

Virtual interfaces are network interfaces that are not associated with a physical interface. Physical interfaces have some form of physical element--for example, an RJ-45 male connector on an Ethernet cable. Virtual interfaces exist only in software; there are no physical elements. You identify an individual virtual interface using a numerical ID after the virtual interface name. For example: loopback 0, tunnel 1, and fastethernet 0/0/0.1. The ID is unique per virtual interface type to make the entire name string unique; for example both a loopback 0 interface and a null 0 interface can exist, but two loopback 0 interfaces cannot exist in a single networking device.

Cisco IOS XE software supports four types of virtual interfaces:

- Loopback
- Null
- Subinterface
- Tunnel

Benefits of Virtual Interfaces

A loopback interface can provide a stable interface on which you can assign a Layer 3 address such as an IP or IPX address. This address can be configured as the source address when the networking device needs to send data for protocols such as NetFlow or Cisco Discovery Protocol (CDP) to another device in your network and you want the receiving device to always see the same source IP address from the networking device. This is an issue in networks with multiple equal-cost paths because under normal circumstances the packets that are generated by a networking device use the IP address from the outbound interface as the source address for the packets and because in a network with two or more equal-cost paths from the networking device to the receiving host each packet might use a different outbound interface.

A null interface provides an alternative method of filtering without the overhead involved with using access lists. For example, instead of creating an outbound access list that prevents traffic to a destination network from being transmitted out an interface, you can configure a static route for the destination network that points to the null interface.

Subinterfaces were invented as a method of virtually subdividing a physical interface into two or more interfaces so that the IP routing protocols would see the network connection to each remote networking device as a separate physical interface even though the subinterfaces share a common physical interface. One of the first uses of subinterfaces was to resolve the problem with split horizon on Frame Relay WANs.

The following are several situations in which tunneling (encapsulating traffic in another protocol) is useful:

- To enable multiprotocol local networks over a single-protocol backbone

- To provide workarounds for networks that use protocols that have limited hop counts; for example, RIP version 1, AppleTalk
- To connect discontiguous subnetworks
- To allow virtual private networks across WANs

Loopback Interfaces

You can specify a software-only interface called a loopback interface to emulate a physical interface. Loopback interfaces are supported on all platforms. A loopback interface is a virtual interface on a Cisco router that remains up (active) after you issue the **no shutdown** command until you disable it with the **shutdown** command. Unlike subinterfaces, loopback interfaces are independent of the state of any physical interface.

The loopback interface can be considered stable because once you enable it, it will remain up until you shut it down. This makes loopback interfaces ideal for assigning Layer 3 addresses such as IP addresses when you want a single address as a reference that is independent of the status of any physical interfaces in the networking device. A good example of this is using the IP address of a loopback interface as the IP address for the domain name system (DNS) host address for the networking device. Before loopback interfaces were available, network administrators had to configure a DNS host entry for every interface on a router that had an IP address assigned to it because they could never be certain which interface IP address might be available at any given time for managing the router. In the sample interface configuration and DNS entries for Router A shown below, you can see that there is a DNS entry for each interface.

Router A Interface Configuration Before Loopback

```
GigabitEthernet0 10.10.10.1 255.255.255.0
GigabitEthernet1 10.10.11.1 255.255.255.0
GigabitEthernet2 10.10.12.1 255.255.255.0
GigabitEthernet3 10.10.13.1 255.255.255.0
GigabitEthernet4 10.10.14.1 255.255.255.0
GigabitEthernet5 10.10.15.1 255.255.255.0
```

Router A DNS Entries Before Loopback

```
RouterA   IN  A  10.10.10.1
          IN  A  10.10.11.1
          IN  A  10.10.12.1
          IN  A  10.10.13.1
          IN  A  10.10.14.1
          IN  A  10.10.15.1
```

Interfaces on networking devices can fail, and they can also be taken out of service for maintenance. If any of the interfaces in Router A fails or is taken out of service, another networking device will not be able to access that interface. When you configure a networking device with a loopback interface and assign it an IP address that is advertised throughout the network, the networking device will be reachable by using this IP address as long as the networking device has at least one network interface capable of sending and receiving IP traffic. In the sample interface configuration and DNS entries for Router A after a loopback interface is configured, you can see that there is now only one DNS entry that can be used to reach the router over any of its physical interfaces.

Router A Interface Configuration After Loopback

```
Loopback 172.16.78.1 255.255.255.0
GigabitEthernet0 10.10.10.1 255.255.255.0
GigabitEthernet1 10.10.11.1 255.255.255.0
GigabitEthernet2 10.10.12.1 255.255.255.0
```

```
GigabitEthernet3 10.10.13.1 255.255.255.0
GigabitEthernet4 10.10.14.1 255.255.255.0
GigabitEthernet5 10.10.15.1 255.255.255.0
```

Router A DNS Entries After Loopback

```
RouterA   IN  A  172.16.78.1
```

The configured IP address of the loopback interface--172.16.78.1--can be used as the source address for packets generated by the router and forwarded to networking management applications and routing protocols. Unless this loopback interface is explicitly shut down, it is always reachable.

You can use the loopback interface as the termination address for open shortest path first (OSPF) or border gateway protocol (BGP) sessions. A loopback interface can also be used to establish a Telnet session from the console port of the device to its auxiliary port when all other interfaces are down. In applications where other routers or access servers attempt to reach this loopback interface, you should configure a routing protocol to distribute the subnet assigned to the loopback address.

IP Packets routed to the loopback interface are rerouted back to the router or access server and processed locally. IP packets routed out the loopback interface but not destined to the loopback interface are dropped. Under these two conditions, the loopback interface can behave like a null interface.

Loopback Interfaces Versus Loopback Mode

Loopback interfaces provide a stable source interface to ensure that the IP address assigned to the interface is always reachable as long as the IP routing protocols continue to advertise the subnet assigned to the loopback interface. Loopback mode, however, is used to test and diagnose issues with WAN (serial) links such as bit loss or data corruption. The idea is to configure a loop to return the data packets that were received by the interface back out the same interface to the device that originated the traffic. Loopback mode is used to troubleshoot problems by checking that the data packets are returned in the same condition in which they were sent. Errors in the data packets indicate a problem with the WAN infrastructure. Many types of serial interfaces have their own form of loopback command syntax that is entered under interface or controller configuration mode.

Null Interfaces

The null interface is a virtual network interface that is similar to the loopback interface. Whereas traffic to the loopback interface is directed to the router itself, traffic sent to the null interface is discarded. This interface is always up and can never forward or receive traffic; encapsulation always fails. The null interface functions similarly to the null devices available on most operating systems.

Null interfaces are used as a low-overhead method of discarding unnecessary network traffic. For example, if you do not want your network users to be able to reach certain IP subnets, you can create static IP routes for the subnets that point to the null interface of a networking device. Using the static IP routes takes less CPU time for the networking device than using IP access lists. The static-route configuration is also easier to configure than IP access lists because it is done in global configuration mode instead of in interface configuration mode.

The null interface may not be configured with an address. Traffic can be sent to this interface only by configuring a static route where the next hop is the null interface--represented by Null 0. One example of configuring the next hop to be the null interface is to create a route to an aggregate network that can then be announced through the BGP, or to ensure that traffic to a particular range of addresses is not propagated through the router, perhaps for security purposes.

The router always has a single null interface. By default, a packet sent to the null interface causes the router to respond by sending an Internet Control Message Protocol (ICMP) unreachable message to the source IP

address of the packet. You can configure the router either to send these responses or to drop the packets silently.

Subinterfaces

Subinterfaces are associated with physical interfaces. Subinterfaces are enabled when the physical interface with which they are associated is enabled, and subinterfaces are disabled when the physical interface is shut down.

Note Subinterfaces can be enabled and shut down independently of the physical port with which they are associated. However, you cannot enable a subinterface of a physical interface that has been shut down.

Subinterfaces are created by subdividing the physical interface into two or more virtual interfaces on which you can assign unique Layer 3 network addresses such as IP subnets. One of the first uses of subinterfaces was to resolve the problem with split horizon on Frame Relay WANs. Split horizon is a behavior associated with IP routing protocols such as RIP in which IP subnets are not advertised back out the same physical interface that they were learned over. Split horizon was implemented to prevent routing loops in IP networks. A routing loop can be created when the networking devices at both ends of a network connection advertise the same IP routes to each other. Split horizon was an issue for Frame Relay multipoint network interfaces--interfaces that connect to two or more remote networking devices over a single physical interface--because the default behavior of many networking devices was to implement split horizon, which means that the networking device did not advertise the IP routes that were learned over an interface back out the interface to other devices that were also reachable via the same physical interface. Subinterfaces were invented as a method of virtually subdividing a physical interface into two or more interfaces so that the IP routing protocols would see the network connection to each remote networking device as a separate physical interface even though the subinterfaces share a common physical interface. Although TCP/IP now disables split horizon limitations by default, protocols such as AppleTalk and IPX are still constrained by split horizon.

Subinterfaces are identified by a prefix that consists of the hardware interface descriptor (IDB) followed by a period and then by a number that is unique for that prefix. The full subinterface number must be unique to the networking device. For example, the first subinterface for GigabitEthernet interface 0/0/0 might be named GigabitEthernet 0/0/0.1 where .1 indicates the subinterface.

Tunnel Interfaces

Tunneling provides a way to encapsulate arbitrary packets inside a transport protocol. Tunnels are implemented as a virtual interface to provide a simple interface for configuration. The tunnel interface is not tied to specific "passenger" or "transport" protocols, but, rather, it is an architecture that is designed to provide the services necessary to implement any standard point-to-point encapsulation scheme.

There are several ways to implement tunnel interfaces depending on the connectivity that you need to provide. One common use for tunnels is to carry data traffic for a network protocol such as IPX over devices in your network that do not support IPX. For instance, if your network uses IPX in sites at the edge of your network but not in the core of your network, you can connect the IPX sites at the network edges by tunneling IPX in IP over the core of the network.

For more details about the various types of tunneling techniques available using Cisco IOS XE software, see the " Implementing Tunnels " module of the *Cisco IOS XE Interface and Hardware Component Configuration Guide.*

How to Configure Virtual Interfaces

Configuring a Loopback Interface

This task explains how to configure a loopback interface. A loopback interface can be considered stable because once you enable it, it will remain up until you shut it down. This makes loopback interfaces ideal for assigning Layer 3 addresses such as IP addresses when you want to have a single address to use as a reference that is independent of the status of any of the physical interfaces in the networking device.

The IP address for the loopback interface must be unique and not in use by another interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface loopback** *number*
4. **ip address** *ip-address mask* [**secondary**]
5. **end**
6. **show interfaces loopback** *number*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface loopback** *number* **Example:** `Router(config)# interface loopback 0`	Specifies a loopback interface and enters interface configuration mode. • Use the *number* argument to specify the number of the loopback interface that you want to create or configure. **Note** There is no limit on the number of loopback interfaces that you can create.

	Command or Action	Purpose
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.20.1.2 255.255.255.0`	Specifies an IP address for the loopback interface and enables IP processing on the interface. • Use the *ip-address* and *mask* arguments to specify the subnet for the loopback address.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 6	**show interfaces loopback** *number* **Example:** `Router# show interfaces loopback 0`	(Optional) Displays information about loopback interfaces. • Use the *number* argument to display information about one particular loopback interface. **Note** Only the syntax applicable to this task is used in this example. For more details, see the Cisco IOS Interface and Hardware Component Command Reference.
Step 7	**exit** **Example:** `Router# exit`	Exits privileged EXEC mode.

Examples

The following is sample output for the **show interfaces loopback** command.

```
Router# show interfaces loopback
Loopback0 is up, line protocol is up
  Hardware is Loopback
  Internet address is 10.20.1.2/24
  MTU 1514 bytes, BW 8000000 Kbit, DLY 5000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation LOOPBACK, loopback not set
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Configuring a Null Interface

This task explains how to configure a null interface. Null interfaces provide an alternative method to access control lists for filtering traffic. All unwanted traffic can be directed to the null interface; the null interface cannot receive or forward traffic, or allow its traffic to be encapsulated.

The only interface configuration command that you can specify for the null interface is the **no ip unreachables** command.

- ICMP Unreachable Messages from Null Interfaces, page 10

ICMP Unreachable Messages from Null Interfaces

By default, a packet sent to the null interface causes the router to respond by sending an Internet Control Message Protocol (ICMP) unreachable message to the source IP address of the packet. You can configure the router either to send these responses or to drop the packets silently.

To disable the sending of ICMP unreachable messages in response to packets sent to the null interface, use the **no ip unreachables** command in interface configuration mode. To reenable the sending of ICMP unreachable messages in response to packets sent to the null interface, use the **ip unreachables** command in interface configuration mode.

Note Only one null interface can be configured on each networking device.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface null** *number*
4. **no ip unreachables**
5. **end**
6. **show interfaces null** [*number*] [**accounting**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface null** *number* **Example:** `Router(config)# interface null 0`	Specifies a null interface and number, and enters interface configuration mode. • The number argument is always 0.
Step 4	**no ip unreachables** **Example:** `Router(config-if)# no ip unreachables`	Prevents the generation of ICMP unreachable messages on an interface. • This command affects all types of ICMP unreachable messages.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 6	**show interfaces null** [*number*] [**accounting**] **Example:** `Router# show interfaces null 0`	(Optional) Displays information about null interfaces. • For null interfaces, the *number* argument is always 0. **Note** Only the syntax applicable to this task is used in this example. For more details, see the Cisco IOS Interface and Hardware Component Command Reference.

Examples

The following is sample output for the **show interfaces null** command.

```
Router# show interfaces null
Null0 is up, line protocol is up
  Hardware is Unknown
  MTU 1500 bytes, BW 10000000 Kbit, DLY 0 usec,
     reliability 0/255, txload 0/255, rxload 0/255
  Encapsulation ARPA, loopback not set
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Configuring a Subinterface

This task explains how to configure a subinterface. Subinterfaces can be enabled and shut down independently of the physical port with which they are associated. However, you cannot enable a subinterface of a physical interface that has been shut down.

The IP address for the interface must be unique and not in use by another interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number.subinterface-number*
4. **ip address** *ip-address mask* [**secondary**]
5. **end**
6. **show interfaces** *type number.subinterface-number*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number.subinterface-number* **Example:** `Router(config)# interface GigabitEthernet 2/3.5`	Specifies the interface type, interface number, and subinterface number and enters interface configuration mode.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 209.165.200.225 255.255.255.224`	Specifies an IP address for the interface and enables IP processing on the interface.

	Command or Action	Purpose
Step 5	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 6	**show interfaces** *type number.subinterface-number* **Example:** `Router# show interfaces GigabitEthernet 2/3.5`	(Optional) Displays information about the interfaces.
Step 7	**exit** **Example:** `Router# exit`	Exits privileged EXEC mode.

Examples

The following is sample output from the **show interfaces**command:

```
Router# show interfaces GigabitEthernet 2/3.5
GigabitEthernet2/3.5432 is down, line protocol is down (notconnect)
  Hardware is c7600 1Gb 802.3, address is 001b.0de6.c100 (bia 001b.0de6.c100)
  Description: *sample*
  Internet address is 10.11.12.13/24
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation 802.1Q Virtual LAN, Vlan ID  2339.
  ARP type: ARPA, ARP Timeout 04:00:00
  Keepalive set (10 sec)
  Last clearing of "show interface" counters never
```

Configuration Examples for Virtual Interfaces

Example Configuring a Loopback Interface

The following example shows the configuration sequence of a loopback interface, loopback 0:

```
interface loopback 0
 ip address 209.165.200.225 255.255.255.0
 end
```

Example Configuring a Null Interface

The following example shows the configuration sequence of a null interface and how to drop the ICMP unreachable messages. All packets sent to the null interface are dropped and in this example, the ICMP messages usually sent in response to packets being sent to the null interface are dropped.

```
interface null 0
 no ip unreachables
 end
```

Example Configuring a Subinterface

The following example shows the configuration sequence of a subinterface:

```
interface GigabitEthernet 2/3.5
 description *sample*
 encapsulation dot1Q 2339
 ip address 209.165.200.225 255.255.255.224
end
```

Where to Go Next

- If you want to implement tunnels in your network, see the " Implementing Tunnels " module of the *Cisco IOS XE Interface and Hardware Component Configuration Guide* .
- If you want to implement physical (hardware) interfaces (such as Gigabit Ethernet or serial interfaces) in your network, see the " Configuring Physical Interfaces " module of the *Cisco IOS XE Interface and Hardware Component Configuration Guide* .

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Interface commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS Interface and Hardware Component Command Reference*
Cisco IOS XE Interface and Hardware Component configuration modules	*Cisco IOS XE Interface and Hardware Component Configuration Guide*
Configuration example showing how to use loopback interfaces with BGP	Sample Configuration for iBGP and eBGP With or Without a Loopback Address

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported, and support for existing RFCs has not been modified.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Implementing Tunnels

This module describes the various types of tunneling techniques. Configuration details and examples are provided for the tunnel types that use physical or virtual interfaces. Many tunneling techniques are implemented using technology-specific commands, and links are provided to the appropriate technology modules.

Tunneling provides a way to encapsulate arbitrary packets inside a transport protocol. Tunnels are implemented as virtual interfaces to provide a simple interface for configuration purposes. The tunnel interface is not tied to specific “passenger” or “transport” protocols, but rather is an architecture to provide the services necessary to implement any standard point-to-point encapsulation scheme.

Note Cisco ASR 1000 Series Aggregation Services Routers support VPN routing and forwarding (VRF)-aware generic routing encapsulation (GRE) tunnel keepalive features.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

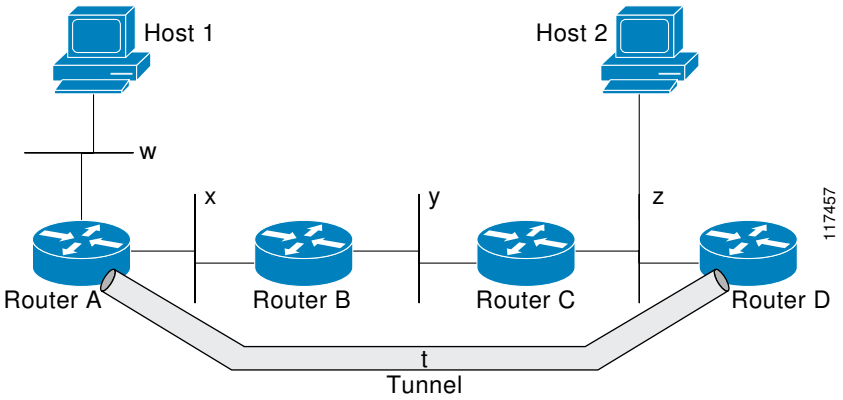
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Implementing Tunnels

- It is important to allow the tunnel protocol to pass through a firewall and access control list (ACL) check.

- Multiple point-to-point tunnels can saturate the physical link with routing information if the bandwidth is not configured correctly on a tunnel interface.
- A tunnel looks like a single hop link, and routing protocols may prefer a tunnel over a multihop physical path. The tunnel, despite looking like a single hop link, may traverse a slower path than a multihop link. A tunnel is as robust and fast, or as unreliable and slow, as the links that it actually traverses. Routing protocols that make their decisions based only on hop counts will often prefer a tunnel over a set of physical links. A tunnel might appear to be a one-hop, point-to-point link and have the lowest-cost path, but the tunnel may actually cost more in terms of latency when compared to an alternative physical topology. For example, in the topology shown in the figure below, packets from Host 1 will appear to travel across networks w, t, and z to get to Host 2 instead of taking the path w, x, y, and z because the tunnel hop count appears shorter. In fact, the packets going through the tunnel will still be traveling across Router A, B, and C, but they must also travel to Router D before coming back to Router C.

Figure 1 Tunnel Precautions: Hop Counts



- A tunnel may have a recursive routing problem if routing is not configured accurately. The best path to a tunnel destination is via the tunnel itself; therefore recursive routing causes the tunnel interface to flap. To avoid recursive routing problems, keep the control-plane routing separate from the tunnel routing by using the following methods:
  - Use a different autonomous system number or tag.
  - Use a different routing protocol.
  - Ensure that static routes are used to override the first hop (watch for routing loops).

  The following error is displayed when there is recursive routing to a tunnel destination:

```
%TUN-RECURDOWN Interface Tunnel 0
temporarily disabled due to recursive routing
```

Information About Implementing Tunnels

Tunneling Versus Encapsulation

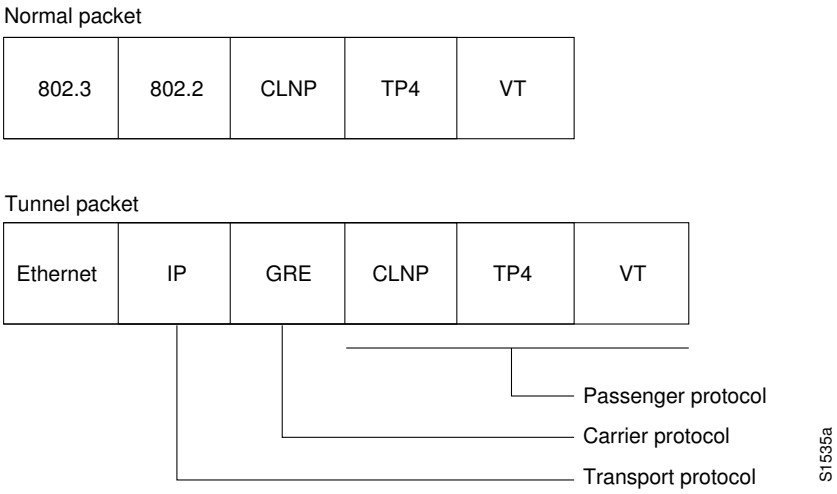
To understand how tunnels work, you must be able to distinguish between concepts of encapsulation and tunneling. Encapsulation is the process of adding headers to data at each layer of a particular protocol stack. The Open Systems Interconnection (OSI) reference model describes the functions of a network. To send a data packet from one host (for example, a PC) to another on a network, encapsulation is used to add a header in front of the data packet at each layer of the protocol stack in descending order. The header must contain a data field that indicates the type of data encapsulated at the layer immediately above the current layer. As the packet ascends the protocol stack on the receiving side of the network, each encapsulation header is removed in reverse order.

Tunneling encapsulates data packets from one protocol within a different protocol and transports the packets on a foreign network. Unlike encapsulation, tunneling allows a lower-layer protocol and a same-layer protocol to be carried through the tunnel. A tunnel interface is a virtual (or logical) interface. Tunneling consists of three main components:

- Passenger protocol—The protocol that you are encapsulating. For example, IPv4 and IPv6 protocols.
- Carrier protocol—The protocol that encapsulates. For example, generic routing encapsulation (GRE) and Multiprotocol Label Switching (MPLS).
- Transport protocol--The protocol that carries the encapsulated protocol. The main transport protocol is IP.

The figure below illustrates IP tunneling terminology and concepts:

Figure 2 IP Tunneling Terminology and Concepts



Tunnel ToS

Tunnel type of service (ToS) allows you to tunnel network traffic and group all packets in the same ToS byte value. The ToS byte values and Time-to-Live (TTL) hop-count value can be set in the encapsulating

IP header of tunnel packets for an IP tunnel interface on a router. Tunnel ToS feature is supported for Cisco Express Forwarding (formerly known as CEF), fast switching, and process switching.

The ToS and TTL byte values are defined in RFC 791. RFC 2474, and RFC 2780 obsolete the use of the ToS byte as defined in RFC 791. RFC 791 specifies that bits 6 and 7 of the ToS byte (the first two least significant bits) are reserved for future use and should be set to 0. For Cisco IOS XE Release 2.1, the Tunnel ToS feature does not conform to this standard and allows you to use the whole ToS byte value, including bits 6 and 7, and to decide to which RFC standard the ToS byte of your packets should conform.

Generic Routing Encapsulation

GRE is defined in RFC 2784. GRE is a carrier protocol that can be used with many different underlying transport protocols and can carry many passenger protocols. RFC 2784 also covers the use of GRE with IPv4 as the transport protocol and the passenger protocol. Cisco software supports GRE as the carrier protocol with many combinations of passenger and transport protocols.

GRE tunnels are described in the following sections:

- GRE Tunnel IP Source and Destination VRF Membership, page 20
- GRE IPv4 Tunnel Support for IPv6 Traffic, page 20

GRE Tunnel IP Source and Destination VRF Membership

The GRE Tunnel IP Source and Destination VRF Membership feature allows you to configure the source and destination of a tunnel to belong to any VPN routing and forwarding (VRFs) tables. A VRF table stores routing data for each VPN. The VRF table defines the VPN membership of a customer site that is attached to the network access server (NAS). Each VRF table comprises an IP routing table, a derived Cisco Express Forwarding table, and guidelines and routing protocol parameters that control the information that is included in the routing table.

Prior to Cisco IOS XE Release 2.2, GRE IP tunnels required the IP tunnel destination to be in the global routing table. The implementation of this feature allows you to configure a tunnel source and destination to belong to any VRF. As with existing GRE tunnels, the tunnel becomes disabled if no route to the tunnel destination is defined.

GRE IPv4 Tunnel Support for IPv6 Traffic

IPv6 traffic can be carried over IPv4 GRE tunnels by using the standard GRE tunneling technique to provide the services necessary to implement a standard point-to-point encapsulation scheme. GRE tunnels are links between either two points, with a separate tunnel for each point. The tunnels are not tied to a specific passenger or transport protocol, but in this case, IPv6 is the passenger protocol, GRE is the carrier protocol, and IPv4 is the transport protocol.

The primary use of GRE tunnels is to provide a stable connection and secure communication between two edge routers, or between an edge router and an end system. The edge router and the end system must have a dual-stack implementation.

GRE has a protocol field that identifies the passenger protocol. GRE tunnels allow intermediate system to intermediate system (IS-IS) or IPv6 to be specified as the passenger protocol; thus, allowing both IS-IS and IPv6 traffic to run over the same tunnel. If GRE does not have a protocol field, it becomes impossible to distinguish whether the tunnel is carrying IS-IS or IPv6 packets.

EoMPLS over GRE

Ethernet over MPLS (EoMPLS) is a tunneling mechanism that allows you to tunnel Layer 2 traffic through a Layer 3 MPLS network. EoMPLS is also known as Layer 2 tunneling.

EoMPLS effectively facilitates Layer 2 extension over long distances. EoMPLS over GRE helps you to create the GRE tunnel as hardware-based switched, and encapsulates EoMPLS frames within the GRE tunnel. The GRE connection is established between the two core routers, and then the MPLS label switched path (LSP) is tunneled over.

GRE encapsulation is used to define a packet that has header information added to it prior to being forwarded. De-encapsulation is the process of removing the additional header information when the packet reaches the destination tunnel endpoint.

When a packet is forwarded through a GRE tunnel, two new headers are added to the front of the packet and hence the context of the new payload changes. After encapsulation, what was originally the data payload and separate IP header are now known as the GRE payload. A GRE header is added to the packet to provide information on the protocol type and the recalculated checksum. A new IP header is also added to the front of the GRE header. This IP header contains the destination IP address of the tunnel.

The GRE header is added to packets such as IP, Layer 2 VPN, and Layer 3 VPN before the header enters into the tunnel. All routers along the path that receives the encapsulated packet use the new IP header to determine how the packet can reach the tunnel endpoint.

In IP forwarding, on reaching the tunnel destination endpoint, the new IP header and the GRE header are removed from the packet and the original IP header is used to forward the packet to the final destination.

The EoMPLS over GRE feature removes the new IP header and GRE header from the packet at the tunnel destination, and the MPLS label is used to forward the packet to the appropriate Layer 2 attachment circuit or Layer 3 VRF.

The scenarios in the following sections describe the L2VPN and L3VPN over GRE deployment on provider edge (PE) or provider (P) routers:

Provider Edge to Provider Edge Generic Routing EncapsulationTunnels

In the Provider Edge to Provider Edge (PE) GRE tunnels scenario, a customer does not transition any part of the core to MPLS but prefers to offer EoMPLS and basic MPLS VPN services. Therefore, GRE tunneling of MPLS traffic is done between PEs.

Provider to Provider Generic Routing Encapsulation Tunnels

In the Provider to Provider (P) GRE tunnels scenario, Multiprotocol Label Switching (MPLS) is enabled between Provider Edge (PE) and P routers but the network core can either have non-MPLS aware routers or IP encryption boxes. In this scenario, GRE tunneling of the MPLS labeled packets is done between P routers.

Provider Edge to Provider Generic Routing Encapsulation Tunnels

In a Provider Edge to Provider GRE tunnels scenario, a network has MPLS-aware P to P nodes. GRE tunneling is done between a PE to P non-MPLS network segment.

Features Specific to Generic Routing Encapsulation

You should understand the following configurations and information for a deployment scenario:

- Tunnel endpoints can be loopbacks or physical interfaces.
- Configurable tunnel keepalive timer parameters per endpoint and a syslog message must be generated when the keepalive timer expires.
- Bidirectional forwarding detection (BFD) is supported for tunnel failures and for the Interior Gateway Protocol (IGP) that use tunnels.
- IGP load sharing across a GRE tunnel is supported.
- IGP redundancy across a GRE tunnel is supported.
- Fragmentation across a GRE tunnel is supported.
- Ability to pass jumbo frames is supported.
- All IGP control plane traffic is supported.
- IP ToS preservation across tunnels is supported.
- A tunnel should be independent of the endpoint physical interface type; for example, ATM, Gigabit, Packet over SONET (POS), and TenGigabit.
- Up to 100 GRE tunnels are supported.

Features Specific to Ethernet over MPLS

- Any Transport over MPLS (AToM) sequencing.
- IGP load sharing and redundancy.
- Port mode Ethernet over MPLS (EoMPLS).
- Pseudowire redundancy.
- Support for up to to 200 EoMPLS virtual circuits (VCs).
- Tunnel selection and the ability to map a specific pseudowire to a GRE tunnel.
- VLAN mode EoMPLS.

Features Specific to Multiprotocol Label Switching Virtual Private Network

- Support for the PE role with IPv4 VRF.
- Support for all PE to customer edge (CE) protocols.
- Load sharing through multiple tunnels and also equal cost IGP paths with a single tunnel.
- Support for redundancy through unequal cost IGP paths with a single tunnel.
- Support for the IP precedence value being copied onto the expression (EXP) bits field of the Multiprotocol Label Switching (MPLS) label and then onto the precedence bits on the outer IPv4 ToS field of the generic routing encapsulation (GRE) packet.

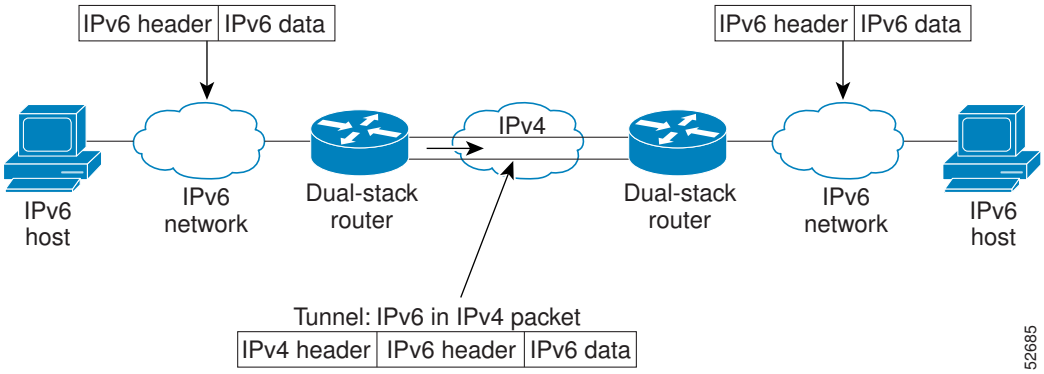
See the section, “Example: Configuring EoMPLS over GRE, page 45” for a sample configuration sequence of EoMPLS over GRE. For more details on EoMPLS over GRE, see the *Deploying and Configuring MPLS Virtual Private Networks In IP Tunnel Environments* document.

Overlay Tunnels for IPv6

The figure below illustrates how overlay tunneling encapsulates IPv6 packets in IPv4 packets for delivery across an IPv4 infrastructure (a core network or the Internet). By using overlay tunnels, you can communicate with isolated IPv6 networks without upgrading the IPv4 infrastructure between them. Overlay tunnels can be configured between border routers or between a border router and a host; however, both tunnel endpoints must support, IPv4 and IPv6 protocol stacks. IPv6 supports the following types of overlay tunneling mechanisms:

- 6to4
- GRE
- Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)
- IPv4-compatible
- Manual

Figure 3 Overlay Tunnels



Note If the basic IPv4 packet header does not have optional fields, overlay tunnels can reduce the maximum transmission unit (MTU) of an interface by 20 octets. A network that uses overlay tunnels is difficult to troubleshoot. Therefore, overlay tunnels that connect isolated IPv6 networks should not be considered as the final IPv6 network architecture. The use of overlay tunnels is considered as a transition technique for a network that supports either both IPv4 and IPv6 protocol stacks or just the IPv6 protocol stack.

Consult the table below to determine which type of tunnel you want to configure to carry IPv6 packets over an IPv4 network.

Table 1 Suggested Usage of Tunnel Types to Carry IPv6 Packets over an IPv4 Network

Tunneling Type	Suggested Usage	Usage Notes
6to4	Point-to-multipoint tunnels that can be used to connect isolated IPv6 sites.	Sites use addresses that begin with the 2002::/16 prefix.

Tunneling Type	Suggested Usage	Usage Notes
GRE/IPv4	Simple point-to-point tunnels that can be used within a site or between sites.	Tunnels can carry IPv6, Connectionless Network ServiceCLNS, and many other types of packets.
ISATAP	Point-to-multipoint tunnels that can be used to connect systems within a site.	Sites can use any IPv6 unicast addresses.
Manual	Simple point-to-point tunnels that can be used within a site or between sites.	Tunnels can carry IPv6 packets only.

Individual tunnel types are discussed in detail in the following concepts, and we recommend that you review and understand the information on the specific tunnel type that you want to implement. Consult the table below for a summary of the tunnel configuration parameters that you may find useful.

Table 2 Overlay Tunnel Configuration Parameters by Tunneling Type

Overlay Tunneling Type	Overlay Tunnel Configuration Parameter			
Tunnel Mode	Tunnel Source	Tunnel Destination	Interface Prefix/ Address	
6to4	ipv6ip 6to4	An IPv4 address or a reference to an interface on which IPv4 is configured.	Not required. These are all point-to-multipoint tunneling types. The IPv4 destination address is calculated, on a per-packet basis, from the IPv6 destination.	An IPv6 address. The prefix must embed the tunnel source IPv4 address.
GRE/IPv4	gre ip		An IPv4 address.	An IPv6 address.
ISATAP	ipv6ip isatap		Not required. These are all point-to-multipoint tunneling types. The IPv4 destination address is calculated on a per-packet basis from the IPv6 destination.	An IPv6 prefix in modified eui-64 format. The IPv6 address is generated from the prefix and the tunnel source IPv4 address.
Manual	ipv6ip		An IPv4 address.	An IPv6 address.

IPv6 Manually Configured Tunnels

A manually configured tunnel is equivalent to a permanent link between two IPv6 domains over an IPv4 backbone. The primary use of a manually configured tunnel is to stabilize connections that require secure communication between two edge routers, or between an end system and an edge router. The manual configuration tunnel also stabilizes connection between remote IPv6 networks.

An IPv6 address is manually configured on a tunnel interface. Manually configured IPv4 addresses are assigned to the tunnel source and destination. The host or router at each end of a configured tunnel must support both the IPv4 and IPv6 protocol stacks. Manually configured tunnels can be configured between border routers or between a border router and a host. Cisco Express Forwarding switching can be used for manually configured IPv6 tunnels. Switching can be disabled if process switching is required.

Automatic 6to4 Tunnels

An automatic 6to4 tunnel allows isolated IPv6 domains to be connected over an IPv4 network to remote IPv6 networks. The key difference between automatic 6to4 tunnels and manually configured tunnels is that the tunnel is not point-to-point; it is point-to-multipoint. In automatic 6to4 tunnels, routers are not configured in pairs because they treat the IPv4 infrastructure as a virtual nonbroadcast multiaccess (NBMA) links. The IPv4 address embedded in the IPv6 address is used to find the other end of the automatic tunnel.

An automatic 6to4 tunnel may be configured on a border router in an isolated IPv6 network, which creates a tunnel on a per-packet basis on a border router in another IPv6 network over an IPv4 infrastructure. The tunnel destination is determined by the IPv4 address of the border router extracted from the IPv6 address that starts with the prefix 2002::/16, where the format is 2002:*border-router-IPv4-address* ::/48.The embedded IPv4 addresses are 16 bits and can be used to number networks within the site. The border router at each end of a 6to4 tunnel must support both IPv4 and IPv6 protocol stacks. 6to4 tunnels are configured between border routers or between a border router and a host.

The simplest deployment scenario for 6to4 tunnels is to interconnect multiple IPv6 sites, each of which has at least one connection to a shared IPv4 network. This IPv4 network could either be the Internet or a corporate backbone. The key requirement is that each site have a globally unique IPv4 address; the Cisco software uses this address to construct a globally unique 6to4/48 IPv6 prefix. A tunnel with appropriate entries in a Domain Name System (DNS) that maps hostnames and IP addresses for both IPv4 and IPv6 domains, allows the applications to choose the required address.

ISATAP Tunnels

The Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) is an automatic overlay tunneling mechanism that uses the underlying IPv4 network as an NBMA link layer for IPv6. ISATAP is designed for transporting IPv6 packets within a site, where a native IPv6 infrastructure is not yet available; for example, when sparse IPv6 hosts are deployed for testing. ISATAP tunnels allow individual IPv4/IPv6 dual-stack hosts within a site to communicate with other such hosts on the same virtual link, basically creating an IPv6 network using the IPv4 infrastructure.

The ISATAP router provides standard router advertisement network configuration support for the ISATAP site. ISATAP allows clients to automatically configure themselves as they would do if they were connected to an Ethernet. ISATAP can also be configured to provide connectivity out of the site. It uses a well-defined IPv6 address format composed of any unicast IPv6 prefix (/64). The prefix can be link-local or global (including 6to4 prefixes), enabling IPv6 routing locally or on the Internet. The IPv4 address is encoded in the last 32 bits of the IPv6 address, enabling automatic IPv6-in-IPv4 tunneling.

Although the ISATAP tunneling mechanism is similar to other automatic tunneling mechanisms, such as IPv6 6to4 tunneling, ISATAP is designed for transporting IPv6 packets within a site, not between sites.

ISATAP uses unicast addresses that include a 64-bit IPv6 prefix and a 64-bit interface identifier. The interface identifier is created in modified Extended Unique Identifier (EUI)-64 format in which the first 32 bits contain the value 000:5EFE to indicate that the address is an IPv6 ISATAP address. The table below shows the layout of an ISATAP address.

Table 3 ISATAP Address Example

64 Bits	32 Bits	32 Bits
Link-local or global IPv6 unicast prefix	0000:5EFE	IPv4 address of the ISATAP link

As shown in the table above, an ISATAP address consists of an IPv6 prefix and the ISATAP interface identifier. This interface identifier includes the IPv4 address of the underlying IPv4 link. The following example shows how an actual ISATAP address looks if the prefix is 2001:0DB8:1234:5678::/64 and the embedded IPv4 address is 10.173.129.8. In the ISATAP address, the IPv4 address is expressed in hexadecimal as 0AAD:8108.

2001:0DB8:1234:5678:0000:5EFE:0AAD:8108

Path MTU Discovery

Path MTU Discovery (PMTUD) can be enabled on a GRE or IP-in-IP tunnel interface. When PMTUD (RFC 1191) is enabled on a tunnel interface, the router performs PMTUD processing for the GRE (or IP-in-IP) tunnel IP packets. The router always performs PMTUD processing on the original data IP packets that enter the tunnel. When PMTUD is enabled, packet fragmentation is not permitted for packets that traverse the tunnel because the Don’t Fragment (DF) bit is set on all the packets. If a packet that enters the tunnel encounters a link with a smaller MTU, the packet is dropped and an Internet Control Message Protocol (ICMP) message is sent 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 caused the packet to be dropped.

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

Use the **tunnel path-mtu-discovery** command to enable PMTUD for the tunnel packets and use the **show interfaces tunnel** command to verify the tunnel PMTUD parameters. PMTUD works only on GRE and IP-in-IP tunnel interfaces.

QoS Options for Tunnels

A tunnel interface supports various quality of service (QoS) features as a physical interface. QoS provides a way to ensure that mission-critical traffic has an acceptable level of performance. QoS options for tunnels include support for applying generic traffic shaping (GTS) directly on the tunnel interface and support for class-based shaping using the modular QoS CLI (MQC). Tunnel interfaces also support class-based policing, but they do not support committed access rate (CAR).

GRE tunnels allow the router to copy the IP precedence bit values of the ToS byte to the tunnel or the GRE IP header that encapsulates the inner packet. Intermediate routers between the tunnel endpoints can use the

IP precedence values to classify packets for QoS features such as policy routing, weighted fair queueing (WFQ), and weighted random early detection (WRED).

When packets are encapsulated by tunnel or encryption headers, QoS features are unable to examine the original packet headers and correctly classify the packets. Packets that travel across the same tunnel have the same tunnel headers, so the packets are treated identically if the physical interface is congested. Tunnel packets can, however, be classified before tunneling and encryption can occur when a user applies the QoS preclassify feature on the tunnel interface or on the crypto map.

Note Class-based WFQ (CBWFQ) inside class-based shaping is not supported on a multipoint interface.

For examples of how to implement some QoS features on a tunnel interface, see the section"Configuring QoS Options on Tunnel Interfaces Examples, page 48" on page 32.

How to Implement Tunnels

Determining the Tunnel Type

Before configuring a tunnel, you must determine the type of tunnel you want to create.

SUMMARY STEPS

1. Determine the passenger protocol. A passenger protocol is the protocol that you are encapsulating.
2. Determine the **tunnel mode** command keyword, if appropriate.

DETAILED STEPS

Step 1 Determine the passenger protocol. A passenger protocol is the protocol that you are encapsulating.

Step 2 Determine the **tunnel mode** command keyword, if appropriate.
The table below shows how to determine the appropriate keyword to be used with the **tunnel mode** command.

Table 4 Determining the tunnel mode Command Keyword

Keyword	Purpose
dvmrp	Use the **dvmrp** keyword to specify that the Distance Vector Multicast Routing Protocol encapsulation will be used.
gre ip	Use the **gre** and **ip** keywords to specify that GRE encapsulation over IP will be used.
gre ipv6	Use the **gre** and **ipv6** keywords to specify that GRE encapsulation over IPv6 will be used.
ipip [**decapsulate-any**]	Use the **ipip** keyword to specify that IP-in-IP encapsulation will be used. The optional **decapsulate-any** keyword terminates any number of IP-in-IP tunnels at one tunnel interface. Note that this tunnel will not carry any outbound traffic; however, any number of remote tunnel endpoints can use a tunnel configured as their destination.
ipv6	Use the **ipv6** keyword to specify that generic packet tunneling in IPv6 will be used.
ipv6ip	Use the **ipv6ip** keyword to specify that IPv6 will be used as the passenger protocol and IPv4 as both the carrier (encapsulation) and transport protocol. When additional keywords are not used, manual IPv6 tunnels are configured. Additional keywords can be used to specify IPv4-compatible, 6to4, or ISATAP tunnels.
mpls	Use the **mpls** keyword to specify that MPLS will be used for configuring traffic engineering (TE) tunnels.

Configuring an IPv4 GRE Tunnel

Perform this task to configure a GRE tunnel. A tunnel interface is used to pass protocol traffic across a network that does not normally support the protocol. To build a tunnel, you must define a tunnel interface on each of the two routers, and the tunnel interfaces must reference each other. At each router, the tunnel interface must be configured with a Layer 3 address. The tunnel endpoints, tunnel source, and tunnel destination must be defined, and the type of tunnel must be selected. Optional steps can be performed to customize the tunnel.

Remember to configure the router at each end of the tunnel. If only one side of a tunnel is configured, the tunnel interface may still come up and stay up (unless keepalive is configured), but packets going into the tunnel will be dropped.

- GRE Tunnel Keepalive, page 29
- What to Do Next, page 31

GRE Tunnel Keepalive

Keepalive packets can be configured to be sent over IP-encapsulated GRE tunnels. You can specify the rate at which keepalives are sent and the number of times that a device will continue to send keepalive packets without a response before the interface becomes inactive. GRE keepalive packets may be sent from both sides of a tunnel or from just one side.

Ensure that the physical interface to be used as the tunnel source in this task is up and configured with the appropriate IP address. For hardware technical descriptions and information about installing interfaces, see the hardware installation and configuration publication for your product.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **bandwidth** *kb/s*
5. **keepalive** [*period* [*retries*]]
6. **tunnel source** {*ip-address* | *interface-type interface-number*}
7. **tunnel destination** {*hostname* | *ip-address*}
8. **tunnel key** *key-number*
9. **tunnel mode gre** { **ip** | **multipoint**}
10. **ip mtu** *bytes*
11. **ip tcp mss** *mss-value*
12. **tunnel path-mtu-discovery** [**age-timer** {*aging-mins* | **infinite**}]
13. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface tunnel 0`	Specifies the interface type and number, and enters interface configuration mode. • To configure a tunnel, use **tunnel** for the *type* argument.

	Command or Action	Purpose
Step 4	**bandwidth** *kb/s* **Example:** `Router(config-if)# bandwidth 1000`	Sets the current bandwidth value for an interface and communicates it to higher-level protocols. • Specifies the tunnel bandwidth to be used to transmit packets. • Use the *kb/s* argument to set the bandwidth, in kilobits per second (kb/s). **Note** This is only a routing parameter; it does not affect the physical interface. The default bandwidth setting on a tunnel interface is 9.6 kb/s. You should set the bandwidth on a tunnel to an appropriate value.
Step 5	**keepalive** [*period* [*retries*]] **Example:** `Router(config-if)# keepalive 3 7`	(Optional) Specifies the number of times the device will continue to send keepalive packets without response before bringing the tunnel interface protocol down. • GRE keepalive packets may be configured either on only one side of the tunnel or on both. • If GRE keepalive is configured on both sides of the tunnel, the *period* and *retries* arguments can be different at each side of the link. **Note** This command is supported only on GRE point-to-point tunnels. **Note** The GRE tunnel keepalive feature should not be configured on a VRF tunnel. This combination of features is not supported.
Step 6	**tunnel source** {*ip-address* | *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source GigabitEthernet 0/0/0`	Configures the tunnel source. • Use the *ip-address* argument to specify the source IP address. • Use the *interface-type* and *interface-number* arguments to specify the interface to be used. **Note** The tunnel source IP address and destination IP addresses must be defined on two separate devices.
Step 7	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Router(config-if)# tunnel destination 10.0.2.1`	Configures the tunnel destination. • Use the *hostname* argument to specify the name of the host destination. • Use the *ip-address* argument to specify the IP address of the host destination. **Note** The tunnel source and destination IP addresses must be defined on two separate devices.
Step 8	**tunnel key** *key-number* **Example:** `Router(config-if)# tunnel key 1000`	(Optional) Enables an ID key for a tunnel interface. • Use the *key-number* argument to identify a tunnel key that is carried in each packet. • Tunnel ID keys can be used as a form of weak security to prevent improper configuration or injection of packets from a foreign source. **Note** This command is supported only on GRE tunnel interfaces. We do not recommend relying on this key for security purposes.

	Command or Action	Purpose
Step 9	**tunnel mode gre** { **ip** | **multipoint**} **Example:** `Device(config-if)# tunnel mode gre ip`	Specifies the encapsulation protocol to be used in the tunnel. • Use the **gre ip** keywords to specify that GRE over IP encapsulation will be used. • Use the **gre multipoint** keywords to specify that multipoint GRE (mGRE) will be used.
Step 10	**ip mtu** *bytes* **Example:** `Device(config-if)# ip mtu 1400`	(Optional) Sets the MTU size of IP packets sent on an interface. • If an IP packet exceeds the MTU set for the interface, the Cisco software will fragment it unless the DF bit is set. • All devices on a physical medium must have the same protocol MTU in order to operate. • For IPv6 packets, use the **ipv6 mtu** command. **Note** If the **tunnel path-mtu-discovery** command is enabled do not configure this command.
Step 11	**ip tcp mss** *mss-value* **Example:** `Device(config-if)# ip tcp mss 250`	(Optional) Specifies the maximum segment size (MSS) for TCP connections that originate or terminate on a router. • Use the *mss-value* argument to specify the maximum segment size for TCP connections, in bytes.
Step 12	**tunnel path-mtu-discovery** [**age-timer** {*aging-mins* | **infinite**}] **Example:** `Device(config-if)# tunnel path-mtu-discovery`	(Optional) Enables PMTUD on a GRE or IP-in-IP tunnel interface. • When PMTUD is enabled on a tunnel interface, PMTUD will operate for GRE IP tunnel packets to minimize fragmentation in the path between the tunnel endpoints.
Step 13	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

What to Do Next

Proceed to the “Verifying Tunnel Configuration and Operation” section.

Configuring GRE on IPv6 Tunnels

This task explains how to configure a GRE tunnel on an IPv6 network. GRE tunnels can be configured to run over an IPv6 network layer and to transport IPv4 packets in IPv6 tunnels.

When GRE/IPv6 tunnels are configured, IPv6 addresses are assigned to the tunnel source and the tunnel destination. The tunnel interface can be assigned either IPv4 or IPv6 addresses assigned (this is not shown

in the following task). The host or router at each end of a configured tunnel must support both the IPv4 and IPv6 protocol stacks.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **tunnel source** {*ipv6-address* | *interface-type interface-number*}
5. **tunnel destination** *ipv6-address*
6. **tunnel mode gre ipv6**
7. **ipv6 mtu** *bytes*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Router(config)# interface tunnel 0`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 4	**tunnel source** {*ipv6-address* | *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source GigabitEthernet 0/0/0`	Specifies the source IPv6 address or the source interface type and number for the tunnel interface. • If an interface type and number are specified, that interface must be configured with an IPv6 address. **Note** Only the syntax used in this context is displayed. For more details, see the *Cisco IPv6 Command Reference*.

	Command or Action	Purpose
Step 5	**tunnel destination** *ipv6-address* **Example:** `Router(config-if)# tunnel destination 2001:0DB8:0C18:2::300`	Specifies the destination IPv6 address for the tunnel interface. **Note** Only the syntax used in this context is displayed. For more details, see the *Cisco IOS IPv6 Command Reference*.
Step 6	**tunnel mode gre ipv6** **Example:** `Router(config-if)# tunnel mode gre ipv6`	Specifies a GRE IPv6 tunnel. **Note** The **tunnel mode gre ipv6** command specifies GRE as the encapsulation protocol for the tunnel.
Step 7	**ipv6 mtu** *bytes* **Example:** `Router(config-if)# ipv6 mtu 1400`	(Optional) Sets the MTU size of IPv6 packets sent on an interface.
Step 8	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

- What to Do Next, page 31

What to Do Next

Proceed to the “Verifying Tunnel Configuration and Operation” section.

Configuring GRE Tunnel IP Source and Destination VRF Membership

This task explains how to configure the source and destination of a tunnel to correspond to any VRF table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *slot*
4. **ip vrf forwarding** *vrf-name*
5. **ip address** *ip-address subnet-mask*
6. **tunnel source** {*ip-address* | *type number*}
7. **tunnel destination** {*hostname* | *ip-address*}
8. **tunnel vrf** *vrf-name*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *slot* **Example:** `Router(config)# interface tunnel 0`	Enters interface configuration mode for the specified interface.
Step 4	**ip vrf forwarding** *vrf-name* **Example:** `Router(config-if)# ip vrf forwarding vrf1`	Defines the VRF associated with the tunnel interface.
Step 5	**ip address** *ip-address subnet-mask* **Example:** `Router(config-if)# ip address 10.7.7.7 255.255.255.0`	Specifies the IP address and subnet mask.
Step 6	**tunnel source** {*ip-address* | *type number*} **Example:** `Router(config-if)# tunnel source loopback 0`	Specifies the tunnel source.

	Command or Action	Purpose
Step 7	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Router(config-if)# tunnel destination 10.5.5.5`	Defines the tunnel destination.
Step 8	**tunnel vrf** *vrf-name* **Example:** `Router(config-if)# tunnel vrf finance1`	Defines the VRF associated with the physical interface from which tunnel packets are sent.
Step 9	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

- What to Do Next, page 31

What to Do Next

Proceed to the “Verifying Tunnel Configuration and Operation” section.

Manually Configuring IPv6 Tunnels

For manually configured IPv6 tunnels, an IPv6 address is configured on a tunnel interface. Manually configured IPv4 addresses are assigned to the tunnel source and the tunnel destination. The host or the router at each end of a configured tunnel must support both IPv4 and IPv6 protocol stacks.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **ipv6 address** *ipv6-prefix/prefix-length* [**eui-64**]
5. **tunnel source** {*ip-address* | *interface-type interface-number*}
6. **tunnel destination** *ip-address*
7. **tunnel mode ipv6ip**
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Router(config)# interface tunnel 0`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	**ipv6 address** *ipv6-prefix/prefix-length* [**eui-64**] **Example:** `Router(config-if)# ipv6 address 2001:0DB8:1234:5678::3/126`	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. **Note** See the "Configuring Basic Connectivity for IPv6" module for more information on configuring IPv6 addresses.
Step 5	**tunnel source** {*ip-address* | *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source GigabitEthernet 0/0/0`	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. • If an interface is specified, the interface must be configured with an IPv4 address.
Step 6	**tunnel destination** *ip-address* **Example:** `Router(config-if)# tunnel destination 10.16.30.1`	Specifies the destination IPv4 address for the tunnel interface.
Step 7	**tunnel mode ipv6ip** **Example:** `Router(config-if)# tunnel mode ipv6ip`	Specifies a manually configured IPv6 tunnel. **Note** The **tunnel mode ipv6ip** command specifies IPv6 as the passenger protocol and IPv4 as both the carrier (encapsulation) and transport protocol for the manual IPv6 tunnel.

	Command or Action	Purpose
Step 8	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

- What to Do Next, page 31

What to Do Next

Proceed to the "Verifying Tunnel Configuration and Operation" section.

Configuring 6to4 Tunnels

With 6to4 tunnels, the tunnel destination is determined by the border-router IPv4 address, which is concatenated to the prefix 2002::/16 in the format 2002:*border-router-IPv4-address* ::/48. The border router at each end of a 6to4 tunnel must support both the IPv4 and IPv6 protocol stacks.

Note The configuration of only one IPv4-compatible tunnel and one 6to4 IPv6 tunnel is supported on a router. If you choose to configure both of these tunnel types on the same router, Cisco recommends that they not share the same tunnel source.

A 6to4 tunnel and an IPv4-compatible tunnel cannot share the same interface because both of them are NBMA "point-to-multipoint" access links, and only the tunnel source can be used to reorder the packets from a multiplexed packet stream into a single packet stream for an incoming interface. When a packet with an IPv4 protocol type of 41 arrives on an interface, the packet is mapped to an IPv6 tunnel interface on the basis of the IPv4 address. However, if both the 6to4 tunnel and the IPv4-compatible tunnel share the same source interface, the router cannot determine the IPv6 tunnel interface to which it should assign the incoming packet.

Manually configured IPv6 tunnels can share the same source interface because a manual tunnel is a "point-to-point" link, and both IPv4 source and the IPv4 destination of the tunnel are defined.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **ipv6 address** *ipv6-prefix/prefix-length* [**eui-64**]
5. **tunnel source** {*ip-address* | *interface-type interface-number*}
6. **tunnel mode ipv6ip 6to4**
7. **exit**
8. **ipv6 route** *ipv6-prefix* / *prefix-length* **tunnel** *tunnel-number*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Router(config)# interface tunnel 0`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 4	**ipv6 address** *ipv6-prefix/prefix-length* [**eui-64**] **Example:** `Router(config-if)# ipv6 address 2002:c0a8:6301:1::1/64`	Specifies the IPv6 address assigned to the interface and enables IPv6 processing on the interface. • The 32 bits following the initial 2002::/16 prefix correspond to an IPv4 address assigned to the tunnel source. **Note** See the "Configuring Basic Connectivity for IPv6" module for more information on configuring IPv6 addresses.
Step 5	**tunnel source** {*ip-address* | *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source GigabitEthernet 0/0/0`	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. **Note** The interface type and number specified in the **tunnel source** command must be configured with an IPv4 address.
Step 6	**tunnel mode ipv6ip 6to4** **Example:** `Router(config-if)# tunnel mode ipv6ip 6to4`	Specifies an IPv6 overlay tunnel using a 6to4 address.
Step 7	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 8	**ipv6 route** *ipv6-prefix / prefix-length* **tunnel** *tunnel-number* **Example:** `Router(config)# ipv6 route 2002::/16 tunnel 0`	Configures a static route to the specified tunnel interface. **Note** When configuring a 6to4 overlay tunnel, you must configure a static route for the IPv6 6to4 prefix 2002::/16 to the 6to4 tunnel interface. • The tunnel number specified in the **ipv6 route** command must be the same tunnel number specified in the **interface tunnel** command.
Step 9	**end** **Example:** `Router(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

• What to Do Next, page 31

What to Do Next

Proceed to the “Verifying Tunnel Configuration and Operation” section.

Configuring ISATAP Tunnels

This task describes how to configure an ISATAP overlay tunnel.

The **tunnel source** command used in the configuration of an ISATAP tunnel must point to an interface that is configured with an IPv4 address. The ISATAP IPv6 address and prefix (or prefixes) advertised are configured for a native IPv6 interface. The IPv6 tunnel interface must be configured with a modified EUI-64 address because the last 32 bits in the interface identifier are constructed using the IPv4 tunnel source address.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **ipv6 address** *ipv6-prefix prefix-length* [**eui-64**]
5. **no ipv6 nd suppress-ra**
6. **tunnel source** {*ip-address* | *interface-type interface-number*}
7. **tunnel mode ipv6ip isatap**
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)# interface tunnel 1`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 4	**ipv6 address** *ipv6-prefix prefix-length* [**eui-64**] **Example:** `Device(config-if)# ipv6 address 2001:0DB8:6301::/64 eui-64`	Specifies the IPv6 address assigned to the interface and enables IPv6 processing on the interface. **Note** For more information on configuring IPv6 addresses, see the "Configuring Basic Connectivity for IPv6" module.
Step 5	**no ipv6 nd suppress-ra** **Example:** `Device(config-if)# no ipv6 nd suppress-ra`	Enables sending of IPv6 device advertisements to allow client autoconfiguration. • Sending of IPv6 device advertisements is disabled by default on tunnel interfaces.
Step 6	**tunnel source** {*ip-address* | *interface-type interface-number*} **Example:** `Device(config-if)# tunnel source GigabitEthernet 1/0/1`	Specifies the source IPv4 address or the source interface type and number for the tunnel interface. **Note** The interface type and number specified in the **tunnel source** command must be configured with an IPv4 address.
Step 7	**tunnel mode ipv6ip isatap** **Example:** `Device(config-if)# tunnel mode ipv6ip isatap`	Specifies an IPv6 overlay tunnel using an ISATAP address.
Step 8	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Verifying Tunnel Configuration and Operation

The **show** and **ping** commands in the steps below can be used in any sequence. The following commands can be used for GRE tunnels, IPv6 manually configured tunnels, and IPv6 over IPv4 GRE tunnels.

SUMMARY STEPS

1. **enable**
2. **show interfaces tunnel** *number* [**accounting**]
3. **ping** [*protocol*] *destination*
4. **show ip route** [*address* [*mask*]]
5. **ping** [*protocol*] *destination*

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show interfaces tunnel** *number* [**accounting**]

Two routers are configured to be endpoints of a tunnel. Device A has Gigabit Ethernet interface 0/0/0 configured as the source for tunnel interface 0 with an IPv4 address of 10.0.0.1 and an IPv6 prefix of 2001:0DB8:1111:2222::1/64. Device B has Gigabit Ethernet interface 0/0/0 configured as the source for tunnel interface 1 with an IPv4 address of 10.0.0.2 and an IPv6 prefix of 2001:0DB8:1111:2222::2/64.

To verify that the tunnel source and destination addresses are configured, use the **show interfaces tunnel** command on Device A.

Example:

```
Device A# show interfaces tunnel 0

Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.0.0.1 (GigabitEthernet0/0/0), destination 10.0.0.2, fastswitch TTL 255
  Tunnel protocol/transport GRE/IP, key disabled, sequencing disabled
  Tunnel TTL 255
  Checksumming of packets disabled,  fast tunneling enabled
  Last input 00:00:14, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/0 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     4 packets input, 352 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     8 packets output, 704 bytes, 0 underruns
```

```
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Step 3 **ping** [*protocol*] *destination*

To check that the local endpoint is configured and working, use the **ping** command on Device A.

Example:

```
DeviceA# ping 2001:0DB8:1111:2222::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:0DB8:1111:2222::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/20 ms
```

Step 4 **show ip route** [*address* [*mask*]]

To check that a route exists to the remote endpoint address, use the **show ip route** command.

Example:

```
DeviceA# show ip route 10.0.0.2

Routing entry for 10.0.0.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
  * directly connected, via GigabitEthernet0/0/0
      Route metric is 0, traffic share count is 1
```

Step 5 **ping** [*protocol*] *destination*

To check that the remote endpoint address is reachable, use the **ping** command on Device A.

Note The remote endpoint address may not be reachable using the **ping** command because of filtering, but the tunnel traffic may still reach its destination.

Example:

```
DeviceA# ping 10.0.0.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/21/28 ms
```

To check that the remote IPv6 tunnel endpoint is reachable, use the **ping** command again on Device A. The note regarding filtering earlier in step also applies to this example.

Example:

```
DeviceA# ping 2001:0DB8:1111:2222::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/20/20 ms
```

These steps may be repeated at the other endpoint of the tunnel.

Configuration Examples for Implementing Tunnels

Example: Configuring a GRE IPv4 Tunnel

The following example shows a simple configuration of GRE tunneling. Note that Gigabit Ethernet interface 0/0/1 is the tunnel source for Router A and the tunnel destination for Router B. Fast Ethernet interface 0/0/1 is the tunnel source for Router B and the tunnel destination for Router A.

Router A

```
interface Tunnel 0
 ip address 10.1.1.2 255.255.255.0
 tunnel source GigabitEthernet 0/0/1
 tunnel destination 192.168.3.2
 tunnel mode gre ip
!
interface GigabitEthernet 0/0/1
 ip address 192.168.4.2 255.255.255.0
```

Router B

```
interface Tunnel 0
 ip address 10.1.1.1 255.255.255.0
 tunnel source FastEthernet 0/0/1
 tunnel destination 192.168.4.2
 tunnel mode gre ip
!
interface FastEthernet 0/0/1
 ip address 192.168.3.2 255.255.255.0
```

The following example configures a GRE tunnel running both IS-IS and IPv6 traffic between Router A and Router B:

Router A

```
ipv6 unicast-routing
clns routing
!
interface Tunnel 0
 no ip address
 ipv6 address 2001:0DB8:1111:2222::1/64
 ipv6 router isis
 tunnel source GigabitEthernet 0/0/0
 tunnel destination 10.0.0.2
 tunnel mode gre ip
!
```

```
interface GigabitEthernet 0/0/0
 ip address 10.0.0.1 255.255.255.0
!
router isis
 network 49.0000.0000.000a.00
```

Router B

```
ipv6 unicast-routing
clns routing
!
interface Tunnel 0
 no ip address
 ipv6 address 2001:0DB8:1111:2222::2/64
 ipv6 router isis
 tunnel source GigabitEthernet 0/0/0
 tunnel destination 10.0.0.1
 tunnel mode gre ip
!
interface GigabitEthernet 0/0/0
 ip address 10.0.0.2 255.255.255.0
!
router isis
 network 49.0000.0000.000b.00
 address-family ipv6
 redistribute static
 exit-address-family
```

Example: Configuring GRE on IPv6 Tunnels

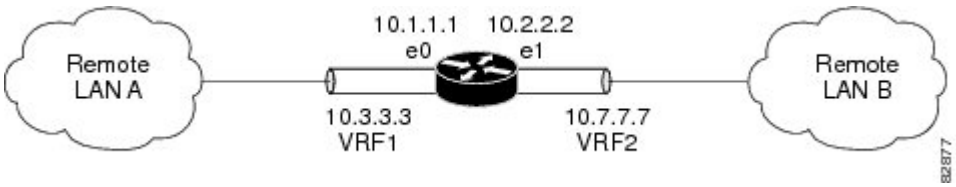
The following example shows how to configure a GRE tunnel over an IPv6 transport. Gigabit Ethernet interface 0/0/0 is configured with an IPv6 address, and this is the source address used by the tunnel interface. The destination IPv6 address of the tunnel is specified directly. In this example, the tunnel carries both IPv4 and IS-IS traffic.

```
interface tunnel 0
 ip address 10.1.1.1 255.255.255.0
 ip router isis
 tunnel source GigabitEthernet 0/0/0
 tunnel destination 2001:DB8:1111:2222::1
 tunnel mode gre ipv6
!
interface FastEthernet 0/0
 no ip address
 ipv6 address 2001:DB8:1111:1111::1/64
!
router isis
 net 49.0001.0000.0000.000a.00
```

Example: Configuring GRE Tunnel IP Source and Destination VRF Membership

In this example, packets received on Fast Ethernet interface 0/0/1 using VRF1, are forwarded out of the tunnel through Fast Ethernet interface 1/0/0 using VRF2. The figure below shows a simple tunnel scenario.

Figure 4 GRE Tunnel Diagram



The following example shows the configuration for the tunnel in the figure above:

```
ip vrf1
 rd 1:1
ip vrf2
 rd 1:2
interface loopback 0
 ip vrf forwarding vrf1
 ip address 10.7.7.7 255.255.255.255
interface tunnel 0
 ip vrf forwarding vrf2
 ip address 10.3.3.3 255.255.255.0
 tunnel source loopback 0
 tunnel destination 10.5.5.5
 tunnel vrf1
interface GigabitEthernet 0/0/0
 ip vrf forwarding vrf2
 ip address 10.1.1.1 255.255.255.0
interface GigabitEthernet 0/0/1
 ip vrf forwarding vrf1
 ip address 10.2.2.2 255.255.255.0
 ip route vrf1 10.5.5.5 255.255.255.0 GigabitEthernet 0/0/1
```

Example: Configuring EoMPLS over GRE

Router A Configuration

```
vrf definition VPN1
 rd 100:1
 address-family ipv4
 route-target both 100:1
 exit-address-family
 !
mpls label protocol ldp
mpls ldp neighbor 209.165.200.224 targeted
mpls ldp router-id Loopback0 force
!
interface tunnel 0
 ip address 209.165.200.225 255.255.255.224
 mpls label protocol ldp
 mpls ip
 keepalive 10 3
 tunnel source TenGigabitEthernet 2/1/0
 tunnel destination 209.165.200.226
!
interface Loopback 0
 ip address 209.165.200.230 255.255.255.224
 !
interface TenGigabitEthernet 2/1/0
 mtu 9216
 ip address 209.165.200.235 255.255.255.224
!
interface TenGigabitEthernet 9/1
 no ip address
!
interface TenGigabitEthernet 9/1.11
 vrf forwarding VPN1
 encapsulation dot1Q 300
 ip address 209.165.200.237 255.255.255.224
!
interface TenGigabitEthernet 9/2
 mtu 9216
 no ip address
xconnect 209.165.200.239 200 encapsulation mpls
!
router bgp 65000
 bgp log-neighbor-changes
 neighbor 209.165.200.240 remote-as 65000
```

```
 neighbor 209.165.200.240 update-source Loopback0
neighbor 209.165.200.245 remote-as 100
 !
address-family vpnv4
  neighbor 209.165.200.240 activate
  neighbor 209.165.200.240 send-community extended
 !
 address-family ipv4 vrf VPN1
  no synchronization
  neighbor 209.165.200.247 remote-as 100
  neighbor 209.165.200.248 activate
  neighbor 209.165.200.249 send-community extended
!
ip route 209.165.200.251 255.255.255.224 tunnel 0
ip route 209.165.200.254 255.255.255.224 209.165.200.256
Router B Configuration
vrf definition VPN1
 rd 100:1
 address-family ipv4
 route-target both 100:1
exit-address-family
 !
mpls ldp neighbor 209.165.200.229 targeted
mpls label protocol ldp
mpls ldp router-id Loopback0 force
!
interface tunnel 0
 ip address 209.165.200.230 255.255.255.224
 mpls label protocol ldp
 mpls ip
 keepalive 10 3
 tunnel source TenGigabitEthernet 3/3/0
 tunnel destination 209.165.200.232
!
interface Loopback 0
 ip address 209.165.200.234 255.255.255.224
!
interface TenGigabitEthernet 2/1/1
 mtu 9216
 no ip address
 xconnect 209.165.200.237 200 encapsulation mpls
!
interface TenGigabitEthernet 2/3/1
 mtu 9216
 no ip address
!
interface TenGigabitEthernet 2/3.11/1
 vrf forwarding VPN1
 encapsulation dot1Q 300
 ip address 209.165.200.239 255.255.255.224
!
interface TenGigabitEthernet 3/3/0
 mtu 9216
 ip address 209.165.200.240 255.255.255.224
!
router bgp 65000
 bgp log-neighbor-changes
 neighbor 209.165.200.241 remote-as 65000
 neighbor 209.165.200.241 update-source Loopback0
 neighbor 209.165.200.244 remote-as 200
 !
 address-family vpnv4
  neighbor 209.165.200.241 activate
  neighbor 209.165.200.241 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN1
  no synchronization
  neighbor 209.165.200.246 remote-as 200
  neighbor 209.165.200.246 activate
  neighbor 209.165.200.246 send-community extended
 exit-address-family
¡
```

```
ip route 209.165.200.226 255.255.255.224 tunnel 0
ip route 209.165.200.229 255.255.255.224 209.165.200.235
```

Example: Manually Configuring IPv6 Tunnels

The following example shows how to manually configure an IPv6 tunnel between Router A and Router B. In the example, tunnel interface 0 for both Router A and Router B is manually configured with a global IPv6 address. The tunnel source and destination addresses are also manually configured.

Router A

```
interface GigabitEthernet 0/0/0
 ip address 192.168.99.1 255.255.255.0
interface tunnel 0
 ipv6 address 2001:0db8:c18:1::3/126
 tunnel source GigabitEthernet 0/0/0
 tunnel destination 192.168.30.1
 tunnel mode ipv6ip
```

Router B

```
interface GigabitEthernet 0/0/0
 ip address 192.168.30.1 255.255.255.0
interface tunnel 0
 ipv6 address 2001:0db8:c18:1::2/126
 tunnel source GigabitEthernet 0/0/0
 tunnel destination 192.168.99.1
 tunnel mode ipv6ip
```

Example: Configuring 6to4 Tunnels

The following example shows how to configure a 6to4 tunnel on a border router in an isolated IPv6 network. The IPv4 address is 192.168.99.1, which translates to the IPv6 prefix of 2002:c0a8:6301::/48. The IPv6 prefix is subnetted into 2002:c0a8:6301::/64 for the tunnel interface: 2002:c0a8:6301:1::/64 for the first IPv6 network and 2002:c0a8:6301:2::/64 for the second IPv6 network. The static route ensures that any other traffic for the IPv6 prefix 2002::/16 is directed to tunnel interface 0 for automatic tunneling.

```
interface GigabitEthernet 0/0/0
 description IPv4 uplink
 ip address 192.168.99.1 255.255.255.0
!
interface GigabitEthernet 0/0/1
 description IPv6 local network 1
 ipv6 address 2002:c0a8:6301:1::1/64
!
interface GigabitEthernet 0/0/2
 description IPv6 local network 2
 ipv6 address 2002:c0a8:6301:2::1/64
!
interface tunnel 0
 description IPv6 uplink
 no ip address
 ipv6 address 2002:c0a8:6301::1/64
 tunnel source GigabitEthernet 0/0/0
 tunnel mode ipv6ip 6to4
!
ipv6 route 2002::/16 tunnel0
```

Example: Configuring ISATAP Tunnels

The following example shows how the tunnel source defined on Gigabit Ethernet interface 0/0/0 and the **tunnel mode** command are used to configure the ISATAP tunnel. Router advertisements are enabled to allow client autoconfiguration.

```
interface tunnel 1
 tunnel source GigabitEthernet 0/0/0
 tunnel mode ipv6ip isatap
 ipv6 address 2001:0DB8::/64 eui-64
 no ipv6 nd suppress-ra
```

Configuring QoS Options on Tunnel Interfaces Examples

The following sample configuration applies GTS directly on the tunnel interface. In this example, the configuration shapes the tunnel interface to an overall output rate of 500 kb/s.

```
interface Tunnel 0
 ip address 10.1.2.1 255.255.255.0
 traffic-shape rate 500000 125000 125000 1000
 tunnel source 10.1.1.1
 tunnel destination 10.2.2.2
```

The following sample configuration shows how to apply the same shaping policy to the tunnel interface with the MQC commands:

```
policy-map tunnel
 class class-default
 shape average 500000 125000 125000
!
interface Tunnel 0
 ip address 10.1.2.1 255.255.255.0
 service-policy output tunnel
 tunnel source 10.1.35.1
 tunnel destination 10.1.35.2
```

- Policing Example, page 48

Policing Example

When an interface becomes congested and packets start to queue, you can apply a queueing method to packets that are waiting to be transmitted. Logical interfaces--tunnel interfaces in this example--do not inherently support a state of congestion and do not support the direct application of a service policy that applies a queueing method. Instead, you must apply a hierarchical policy. Create a "child" or lower-level policy that configures a queueing mechanism, such as low-latency queueing, with the **priority** command and CBWFQ with the **bandwidth** command.

```
policy-map child
 class voice
 priority 512
```

Create a "parent" or top-level policy that applies class-based shaping. Apply the child policy as a command under the parent policy because admission control for the child class is done according to the shaping rate for the parent class.

```
policy-map tunnel
 class class-default
 shape average 2000000
 service-policy child
```

Apply the parent policy to the tunnel interface.

```
interface tunnel 0
 service-policy tunnel
```

In the following example, a tunnel interface is configured with a service policy that applies queueing without shaping. A log message is displayed noting that this configuration is not supported.

```
Router(config)# interface tunnel1
Router(config-if)# service-policy output child
Class Based Weighted Fair Queueing not supported on this interface
```

Additional References

The following sections provide references related to implementing tunnels.

Related Documents

Related Topic	Document Title
All Cisco IOS XE commands	*Cisco IOS Master Command List, All Releases* .
Tunnel commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS Interface and Hardware Component Command Reference*
IPv6 commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS IPv6 Command Reference*
Cisco IOS XE Interface and Hardware Component configuration modules	*Cisco IOS XE Interface and Hardware Component Configuration Guide*, Release 2
Cisco IOS XE IPv6 configuration modules	*Cisco IOS XE IPv6 Configuration Guide*, Release 2
Cisco IOS XE Quality of Service Solutions configuration modules	*Cisco IOS XE Quality of Service Solutions Configuration Guide*
Cisco IOS XE Multiprotocol Label Switching configuration modules	*Cisco IOS XE Multiprotocol Label Switching Configuration Guide*
Configuration example for a VRF-aware dynamic multipoint VPN (DMVPN)	"Dynamic Multipoint VPN (DMVPN)" configuration module in the *Cisco IOS XE Security Configuration Guide: Secure Connectivity*

Standards/RFCs

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

Standard	Title
RFC 791	*Internet Protocol*
RFC 1191	*Path MTU Discovery*
RFC 1323	*TCP Extensions for High Performance*
RFC 1483	*Multiprotocol Encapsulation over ATM Adaptation Layer 5*
RFC 2003	*IP Encapsulation Within IP*
RFC 2018	*TCP Selective Acknowledgment Options*
RFC 2460	*Internet Protocol, Version 6 (IPv6)*
RFC 2473	*Generic Packet Tunneling in IPv6 Specification*
RFC 2474	*Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*
RFC 2516	*A Method for Transmitting PPP over Ethernet (PPPoE)*
RFC 2547	*BGP/MPLS VPNs*
RFC 2780	*IANA Allocation Guidelines for Values in the Internet Protocol and Related Headers*
RFC 2784	*Generic Routing Encapsulation (GRE)*
RFC 2890	*Key and Sequence Number Extensions to GRE*
RFC 2893	*Transition Mechanisms for IPv6 Hosts and Routers*
RFC 3056	*Connection of IPv6 Domains via IPv4 Clouds*
RFC 3147	*Generic Routing Encapsulation over CLNS Networks*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Implementing Tunnels

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5 Feature Information for Implementing Tunnels

Feature Name	Releases	Feature Information
EoMPLS over GRE	Cisco IOS XE Release 2.5	The EoMPLS over GRE feature allows you to tunnel Layer 2 traffic through a Layer 3 MPLS network. This feature also helps to create the GRE tunnel as hardware-based switched, and with high performance that encapsulates EoMPLS frames within the GRE tunnel. No new commands were introduced or modified by this feature.
GRE Tunnel IP Source and Destination VRF Membership	Cisco IOS XE Release 2.2	The GRE Tunnel IP Source and Destination VRF Membership feature allows you to configure the source and destination of a tunnel to belong to any VPN VRF table. The following command was introduced or modified: **tunnel vrf**.

Feature Name	Releases	Feature Information
GRE Tunnel Keepalive	Cisco IOS XE Release 2.1	The GRE Tunnel Keepalive feature provides the capability of configuring keepalive packets to be sent over IP-encapsulated GRE tunnels. You can specify the rate at which keepalives will be sent and the number of times that a device will continue to send keepalive packets without a response before the interface becomes inactive. GRE keepalive packets may be sent from both sides of a tunnel or from just one side. The following command was introduced by this feature: **keepalive (tunnel interfaces)** .
IP over IPv6 Tunnels	Cisco IOS XE Release 2.4	The following commands were modified by this feature: **tunnel destination**, **tunnel mode**, and **tunnel source**.
IP Precedence for GRE Tunnels	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Aggregation Services Routers.
IP Tunnel— SSO	Cisco IOS XE Release 3.6	High availability support was added to IP Tunnels. No new commands were introduced or modified by this feature.
Tunnel ToS	Cisco IOS XE Release 2.1	The Tunnel ToS feature allows you to configure the ToS and Time-to-Live (TTL) byte values in the encapsulating IP header of tunnel packets for an IP tunnel interface on a router. The Tunnel ToS feature is supported in Cisco Express Forwarding, fast switching, and process switching forwarding modes. The following commands were introduced or modified by this feature: **show interfaces tunnel**, **tunnel tos**, **tunnel,** and **ttl**.