# Generic Routing Encapsulation

This document describes the Generic Routing Encapsulation (GRE) feature. This feature is a tunneling protocol that enables the encapsulation of a wide variety of protocol packet types inside IP tunnels, creating a virtual point-to-point link to Cisco routers at remote points over an IP internetwork.


- Finding Feature Information, on page 1
- Hardware Compatibility Matrix for the Cisco cBR Series Routers, on page 2
- Restrictions for Implementing Tunnels, on page 2
- Restrictions for GRE IPv6 Tunnels, on page 3
- Information About Implementing Tunnels, on page 4
- Information About IPv6 over IPv4 GRE Tunnels, on page 5
- Information About GRE IPv6 Tunnels, on page 8
- How to Implement Tunnels, on page 8
- Configuration Examples for Implementing Tunnels, on page 15
- How to Configure IPv6 over IPv4 GRE Tunnels, on page 18
- Configuration Examples for IPv6 over IPv4 GRE Tunnels, on page 19
- How to Configure GRE IPv6 Tunnels, on page 20
- Configuration Examples for GRE IPv6 Tunnels, on page 22
- Additional References, on page 23
- Feature Information for Generic Routing Encapsulation , on page 24


# Finding Feature Information

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

# Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 1: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

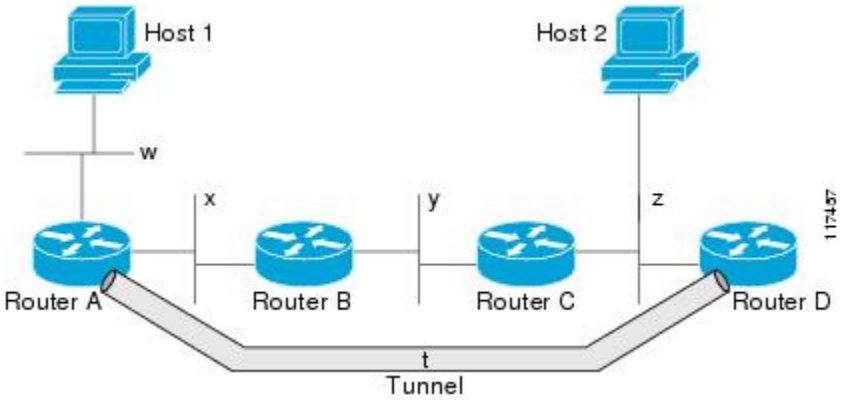
| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G<br>• PID—CBR-CCAP-SUP-60G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# 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
```

# Restrictions for GRE IPv6 Tunnels

- GRE tunnel keepalive packets are not supported.
- Multipoint GRE (mGRE) IPv6 tunneling is not supported.

# 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.

## 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.

## 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, on page 17" on page 32.

# Information About IPv6 over IPv4 GRE Tunnels

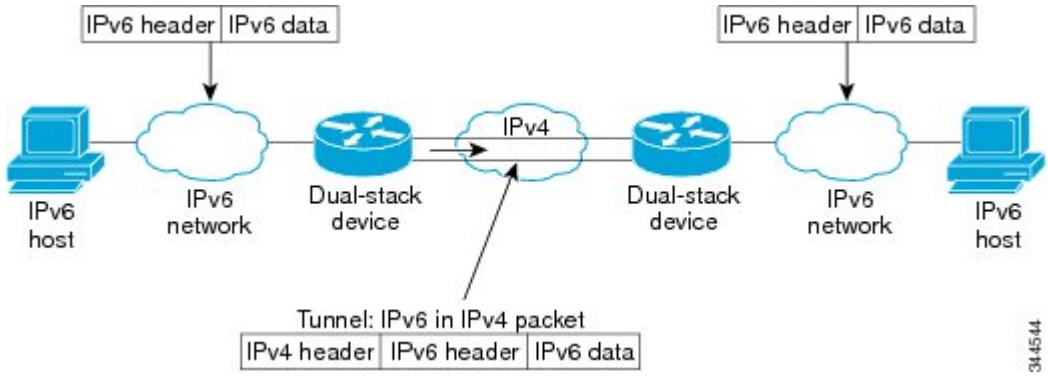
## Overlay Tunnels for IPv6

Overlay tunneling encapsulates IPv6 packets in IPv4 packets for delivery across an IPv4 infrastructure (a core network or the figure below). 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 devices or between a border device and a host; however, both tunnel endpoints must support both the IPv4 and IPv6 protocol stacks. IPv6 supports the following types of overlay tunneling mechanisms:

- Manual
- Generic routing encapsulation (GRE)

- IPv4-compatible
- 6to4
- Intrasite Automatic Tunnel Addressing Protocol (ISATAP)

***Figure 2: Overlay Tunnels***




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

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

***Table 2: Suggested Usage of Tunnel Types to Carry IPv6 Packets over an IPv4 Network***

| Tunneling Type | Suggested Usage | Usage Notes |
|---|---|---|
| Manual | Simple point-to-point tunnels that can be used within a site or between sites. | Can carry IPv6 packets only. |
| GRE- and IPv4-compatible | Simple point-to-point tunnels that can be used within a site or between sites. | Can carry IPv6, Connectionless Network Service (CLNS), and many other types of packets. |
| IPv4- compatible | Point-to-multipoint tunnels. | Uses the ::/96 prefix. We do not recommend using this tunnel type. |
| 6to4 | Point-to-multipoint tunnels that can be used to connect isolated IPv6 sites. | Sites use addresses from the 2002::/16 prefix. |
| 6RD | IPv6 service is provided to customers over an IPv4 network by using encapsulation of IPv6 in IPv4. | Prefixes can be from the SP's own address block. |

| Tunneling Type | Suggested Usage | Usage Notes |
|---|---|---|
| ISATAP | Point-to-multipoint tunnels that can be used to connect systems within a site. | Sites can use any IPv6 unicast addresses. |

Individual tunnel types are discussed in detail in this document. We recommend that you review and understand the information about the specific tunnel type that you want to implement. When you are familiar with the type of tunnel you need, see the table below for a summary of the tunnel configuration parameters that you may find useful.

***Table 3: Tunnel Configuration Parameters by Tunneling Type***

| Tunneling Type | Tunnel Configuration Parameter | | | |
|---|---|---|---|---|
| **Tunnel Mode** | **Tunnel Source** | **Tunnel Destination** | **Interface Prefix or Address** | |
| Manual | ipv6ip | An IPv4 address, or a reference to an interface on which IPv4 is configured. | An IPv4 address. | An IPv6 address. |
| GRE/IPv4 | gre ip | | An IPv4 address. | An IPv6 address. |
| IPv4- compatible | ipv6ip auto-tunnel | | 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. | Not required. The interface address is generated as ::*tunnel-source*/96. |
| 6to4 | ipv6ip 6to4 | | | An IPv6 address. The prefix must embed the tunnel source IPv4 address. |
| 6RD | ipv6ip 6rd | | | An IPv6 address. |
| ISATAP | ipv6ip isatap | | | An IPv6 prefix in modified eui-64 format. The IPv6 address is generated from the prefix and the tunnel source IPv4 address. |

# GRE IPv4 Tunnel Support for IPv6 Traffic

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

The primary use of GRE tunnels is for stable connections that require regular secure communication between two edge devices or between an edge device and an end system. The edge devices and the end systems must be dual-stack implementations.

# Information About GRE IPv6 Tunnels

## Overview of GRE IPv6 Tunnels

The GRE IPv6 Tunnels feature enables the delivery of packets from other protocols through an IPv6 network and allows the routing of IPv6 packets between private networks across public networks with globally routed IPv6 addresses.

For point-to-point GRE tunnels, each tunnel interface requires a tunnel source IPv6 address and a tunnel destination IPv6 address when being configured. All packets are encapsulated with an outer IPv6 header and a GRE header.

# 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

### Procedure

**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. |

| Keyword | Purpose |
|---|---|
| **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

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.

### Before you begin

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.

### Procedure

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Router(config)# interface tunnel 0` | Specifies the interface type and number, and enters interface configuration mode.<br>• To configure a tunnel, use **tunnel** for the *type* argument. |
| **Step 4** | **bandwidth** *kb/s*<br>**Example:**<br>`Router(config-if)# bandwidth 1000` | Sets the current bandwidth value for an interface and communicates it to higher-level protocols.<br>• Specifies the tunnel bandwidth to be used to transmit packets.<br>• Use the *kb/s* argument to set the bandwidth, in kilobits per second (kb/s).<br>**Note**<br>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*]]<br>**Example:**<br>`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.<br>• GRE keepalive packets may be configured either on only one side of the tunnel or on both.<br>• 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.<br>**Note**<br>This command is supported only on GRE point-to-point tunnels.<br>**Note**<br>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*}<br>**Example:**<br>`Router(config-if)# tunnel source TenGigabitEthernet 4/1/0` | Configures the tunnel source.<br>**Note**<br>The tunnel source IP address and destination IP addresses must be defined on two separate devices. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 7** | **tunnel destination** {*hostname* \| *ip-address*}<br>**Example:**<br>`Router(config-if)# tunnel destination 10.0.2.1` | Configures the tunnel destination.<br>**Note** The tunnel source and destination IP addresses must be defined on two separate devices. |
| **Step 8** | **tunnel key** *key-number*<br>**Example:**<br>`Router(config-if)# tunnel key 1000` | (Optional) Enables an ID key for a tunnel interface.<br>**Note** This command is supported only on GRE tunnel interfaces. We do not recommend relying on this key for security purposes. |
| **Step 9** | **tunnel mode gre** { **ip** \| **multipoint**}<br>**Example:**<br>`Device(config-if)# tunnel mode gre ip` | Specifies the encapsulation protocol to be used in the tunnel. |
| **Step 10** | **ip mtu** *bytes*<br>**Example:**<br>`Device(config-if)# ip mtu 1400` | (Optional) Sets the MTU size of IP packets sent on an interface.<br>• If an IP packet exceeds the MTU set for the interface, the Cisco software will fragment it unless the DF bit is set.<br>• All devices on a physical medium must have the same protocol MTU in order to operate.<br>• For IPv6 packets, use the **ipv6 mtu** command.<br>**Note** If the **tunnel path-mtu-discovery** command is enabled do not configure this command. |
| **Step 11** | **ip tcp mss** *mss-value*<br>**Example:**<br>`Device(config-if)# ip tcp mss 250` | (Optional) Specifies the maximum segment size (MSS) for TCP connections that originate or terminate on a router. |
| **Step 12** | **tunnel path-mtu-discovery** [**age-timer** {*aging-mins* \| **infinite**}]<br>**Example:**<br>`Device(config-if)# tunnel path-mtu-discovery` | (Optional) Enables PMTUD on a GRE or IP-in-IP tunnel interface.<br>• 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**<br>**Example:**<br>`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 6to4 Tunnels

### Before you begin

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.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface tunnel** *tunnel-number*<br>**Example:**<br>`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**]<br>**Example:** | Specifies the IPv6 address assigned to the interface and enables IPv6 processing on the interface. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config-if)# ipv6 address 2002:c0a8:6301:1::1/64` | • The 32 bits following the initial 2002::/16 prefix correspond to an IPv4 address assigned to the tunnel source.<br>**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*}<br>**Example:**<br>`Router(config-if)# tunnel source TenGigabitEthernet 4/1/0` | Specifies the source IPv4 address or the source interface type and number for the tunnel interface.<br>**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**<br>**Example:**<br>`Router(config-if)# tunnel mode ipv6ip 6to4` | Specifies an IPv6 overlay tunnel using a 6to4 address. |
| **Step 7** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Exits interface configuration mode and returns to global configuration mode. |
| **Step 8** | **ipv6 route** *ipv6-prefix* / *prefix-length* **tunnel** *tunnel-number*<br>**Example:**<br>`Router(config)# ipv6 route 2002::/16 tunnel 0` | Configures a static route to the specified tunnel interface.<br>**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.<br>• The tunnel number specified in the **ipv6 route** command must be the same tunnel number specified in the **interface tunnel** command. |
| **Step 9** | **end**<br>**Example:**<br>`Router(config)# end` | Exits global configuration mode and returns to privileged EXEC mode. |

## What to Do Next

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

# 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.

## Procedure

**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 TenGigabit Ethernet interface 4/1/0 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 TenGigabit Ethernet interface 4/1/0 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 (TenGigabitEthernet4/1/0), 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 TenGigabitEthernet4/1/0
      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 TenGigabit Ethernet interface 4/1/0 is the tunnel source for Router A and the tunnel destination for Router B. TenGigabit Ethernet interface 4/1/1 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 TenGigabitEthernet 4/1/0
 tunnel destination 192.168.3.2
 tunnel mode gre ip
!
interface TenGigabitEthernet 4/1/0
 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 TenGigabitEthernet 4/1/1
 tunnel destination 192.168.4.2
 tunnel mode gre ip
!
interface TenGigabitEthernet 4/1/1
 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 TenGigabitEthernet 4/1/0
 tunnel destination 10.0.0.2
 tunnel mode gre ip
!
interface TenGigabitEthernet 4/1/0
 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 TenGigabitEthernet 4/1/0
 tunnel destination 10.0.0.1
 tunnel mode gre ip
!
interface TenGigabitEthernet 4/1/0
 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
```

# 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

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
```

# How to Configure IPv6 over IPv4 GRE Tunnels

## Configuring GRE on IPv6 Tunnels

GRE tunnels can be configured to run over an IPv6 network layer and to transport IPv4 and IPv6 packets in IPv6 tunnels.

### Before you begin

When GRE IPv6 tunnels are configured, IPv6 addresses are assigned to the tunnel source and the tunnel destination. The tunnel interface can have either IPv4 addresses or IPv6 addresses assigned (this is not shown in the task). The host or device at each end of a configured tunnel must support both the IPv4 and IPv6 protocol stacks.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface tunnel** *tunnel-number*<br>**Example:**<br>`Device(config)# interface tunnel 0` | Specifies a tunnel interface and number, and enters interface configuration mode. |
| **Step 4** | Enter one of the following commands:<br>• **ipv6 address** {*ipv6-address/prefix-length* \| *prefix-name sub-bits/prefix-length*}<br>• **ipv6 address** *ipv6-prefix/prefix-length* [**eui-64**]<br>**Example:** | Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface.<br>• If you specify the **eui-64** keyword, the software configures an IPv6 address for an interface and enables IPv6 processing on the interface using an EUI-64 interface ID in the low-order 64 bits of the address. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config-if)# ipv6 address 3ffe:b00:c18:1::3/127` | |
| **Step 5** | **tunnel source** {*ip-address* \| *ipv6-address* \| *interface-type interface-number*}<br>**Example:**<br>`Device(config-if)# tunnel source Tengigabitethernet 4/1/0` | Specifies the source IPv4 address, IPv6 address, or the source interface type and number for the tunnel interface.<br>• If an interface is specified, the interface must be configured with an IPv4 address. |
| **Step 6** | **tunnel destination** {*hostname* \| *ip-address* \| *ipv6-address*}<br>**Example:**<br>`Device(config-if)# tunnel destination 2001:DB8:1111:2222::1/64` | Specifies the destination IPv4 address, IPv6 address, or hostname for the tunnel interface. |
| **Step 7** | **tunnel mode** {**aurp** \| **cayman** \| **dvmrp** \| **eon** \| **gre** \| **gre multipoint** \| **gre ipv6** \| **ipip** [**decapsulate-any**] \| **iptalk** \| **ipv6** \| **mpls** \| **nos**}<br>**Example:**<br>`Device(config-if)# tunnel mode gre ipv6` | Specifies a GRE IPv6 tunnel.<br>**Note**<br>The **tunnel mode gre ipv6** command specifies GRE as the encapsulation protocol for the tunnel. |
| **Step 8** | **end**<br>**Example:**<br>`Device(config-if)# end` | Returns to privileged EXEC mode. |

# Configuration Examples for IPv6 over IPv4 GRE Tunnels

## Example: GRE Tunnel Running IS-IS and IPv6 Traffic

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

**Router A Configuration**

```
ipv6 unicast-routing
clns routing
!
interface tunnel 0
 no ip address
 ipv6 address 3ffe:b00:c18:1::3/127
 ipv6 router isis
 tunnel source TenGigabitEthernet 4/1/0
 tunnel destination 2001:DB8:1111:2222::1/64
 tunnel mode gre ipv6
```

```
!
interface TenGigabitEthernet4/1/0
 ip address 10.0.0.1 255.255.255.0
!
router isis
 net 49.0000.0000.000a.00
```

**Router B Configuration**

```
ipv6 unicast-routing
clns routing
!
interface tunnel 0
 no ip address
 ipv6 address 3ffe:b00:c18:1::2/127
 ipv6 router isis
 tunnel source TenGigabitEthernet 4/1/0
 tunnel destination 2001:DB8:1111:2222::2/64
 tunnel mode gre ipv6
!
interface TenGigabitEthernet4/1/0
 ip address 10.0.0.2 255.255.255.0
!
router isis
 net 49.0000.0000.000b.00
 address-family ipv6
 redistribute static
 exit-address-family
```

## Example: Tunnel Destination Address for IPv6 Tunnel

```
Router(config)#interface Tunnel0
Router(config-if)#ipv6 address 2001:1:1::1/48
Router(config-if)#tunnel source TenGigabitEthernet 4/1/0
Router(config-if)#tunnel destination 10.0.0.2
Router(config-if)#tunnel mode gre ipv6
Router(config-if)#exit
!
Router(config)#interface TenGigabitEthernet4/1/0
Router(config-if)#ip address 10.0.0.1 255.255.255.0
Router(config-if)#exit
!
Router(config)#ipv6 unicast-routing
Router(config)#router isis
Router(config)#net 49.0000.0000.000a.00
```

# How to Configure GRE IPv6 Tunnels

## Configure CDP Over GRE IPv6 Tunnels

Perform this task to configure a GRE tunnel on an IPv6 network. GRE tunnels can be configured to run over an IPv6 network layer and transport IPv6 and IPv4 packets through IPv6 tunnels.

### Before you begin

When GRE IPv6 tunnels are configured, IPv6 addresses are assigned to the tunnel source and the tunnel destination. The tunnel interface can have either IPv4 or IPv6 addresses. The host or device at each end of the configured tunnel must support both IPv4 and IPv6 protocol stacks.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface tunnel** *tunnel-number*<br>**Example:**<br>`Device(config)# interface tunnel 0` | Specifies a tunnel interface and number and enters interface configuration mode. |
| **Step 4** | **CDP enable**<br>**Example:**<br>`Device(config)# CDP enable` | Enables Cisco Discovery Protocol on the interface. |
| **Step 5** | **tunnel source** {*ipv6-address* \| *interface-type* \|*interface-number* }<br>**Example:**<br>`Device(config-if)# tunnel source ethernet 0` | Specifies the source IPv6 address or the source interface type and number for the tunnel interface.<br>• If an interface type and number are specified, the interface must be configured with an IPv6 address.<br>**Note**<br>For more information on the tunnel source command, refer to the IPv6 command reference guide. |
| **Step 6** | **tunnel destination** *ipv6-address*<br>**Example:**<br>`Device(config-if)# tunnel destination 2001:0DB8:0C18:2::300` | Specifies the destination IPv6 address for the tunnel interface.<br>**Note**<br>For more information on the tunnel destination command, refer to the IPv6 command reference guide. |
| **Step 7** | **tunnel mode gre ipv6**<br>**Example:**<br>`Device(config-if)# tunnel mode gre ipv6` | Specifies a GRE IPv6 tunnel.<br>**Note**<br>The **tunnel mode gre ipv6** command specifies GRE as the encapsulation protocol for the tunnel interface. Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference. |

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

# Configuration Examples for GRE IPv6 Tunnels

## Example: Configuring CDP Over GRE IPv6 Tunnels

The following example shows how to configure a GRE tunnel over an IPv6 transport. In this example, Ethernet0/0 has 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 Tunnel0
 ip address 10.1.1.1 255.255.255.0
 ip router isis
 tunnel source Ethernet0/0
 tunnel destination 2001:DB8:1111:2222::1
 tunnel mode gre ipv6
!
interface Ethernet0/0
 no ip address
 ipv6 address 2001:DB8:1111:1111::1/64
!
router isis
 net 49.0001.0000.0000.000a.00
```

The following example shows how to configure CDP on GRE IPv6 P2P Tunnel Interface.

```
interface Tunnel1
  cdp enable
  ipv6 address 20::1/64
  tunnel source Ethernet0/0
  tunnel mode gre ipv6
  tunnel destination 10::2
end
```

The following example shows how to configure CDP on GRE IPv6 Multipoint Tunnel Interface.

```
interface Tunnel1
   ipv6 address 172::2/64
   ipv6 nhrp map 172::1/64 192::1
   ipv6 nhrp map multicast 192::1
    ipv6 nhrp network-id 1
   ipv6 nhrp nhs 172::1
   llp nhrp map multicast 192::1
   tunnel source 2000::1
   tunnel mode gre multipoint ipv6
end
```

The following show example displays the CDP neighbor tunnels that are configured in a device.

```
Router#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay
```

```
Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Router           Tunnel1           179               R    Linux Uni Tunnel1
```

# Additional References

The following sections provide references related to the GRE feature.

### Related Documents

| Related Topic | Document Title |
|---|---|
| CMTS Command Reference | Cisco CMTS Cable Command Reference, at the following URL: http://www.cisco.com/c/en/us/td/docs/cable/cmts/cmd_ref/b_cmts_cable_cmd_ref.html |
| Configuring GRE Tunnel over Cable | Configuring GRE Tunnel over Cable, at the following URL: http://www.cisco.com/en/US/tech/tk86/tk89/technologies_configuration_example09186a008011520d.sht |

### Standards

| Standard | Title |
|---|---|
| SP-RFIv1.1-I09-020830 | Data-over-Cable Service Interface Specifications Radio Frequency Interface Specification, version 1.1 ( http://www.cablemodem.com ) |

### MIBs

| MIB | MIBs Link |
|---|---|
| No new or modified MIBs are supported by this feature. | To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://tools.cisco.com/ITDIT/MIBS/servlet/index |

### RFCs

| RFC | Title |
|---|---|
| RFC 1701 | Generic Routing Encapsulation (GRE) |
| RFC 1702 | Generic Routing Encapsulation over IPv4 networks |
| RFC 1853 | IP in IP Tunneling |
| RFC 2003 | IP Encapsulation within IP |
| RFC 2784 | Generic Routing Ecapsulation (GRE) |

| RFC | Title |
|---|---|
| RFC 2890 | Key and Sequence Number Extensions to GRE |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Technical Support & Documentation website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Generic Routing Encapsulation

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 5: Feature Information for Generic Routing Encapsulation*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Generic Routing Encapsulation | Cisco IOS XE Fuji 16.7.1 | This feature was integrated on the Cisco cBR Series Converged Broadband Routers. |