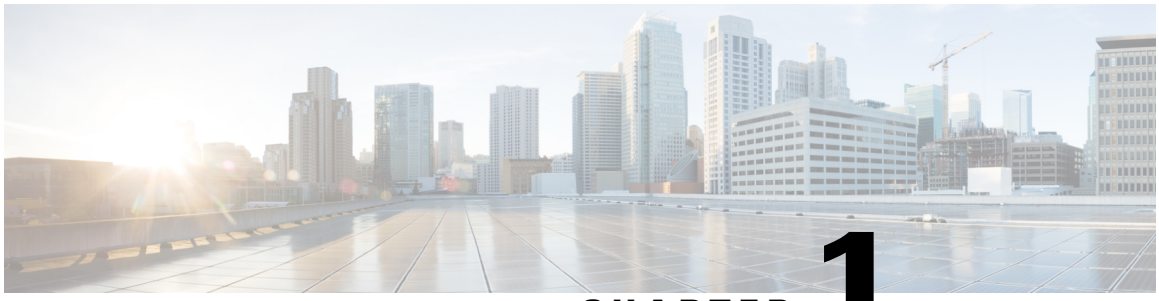
Interface and Hardware Component Configuration Guide, Cisco IOS XE Release 3S (ASR 1000)

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



CHAPTER 1

mGRE Tunnel Support over IPv6

mGRE tunnels are configured to enable service providers deploy IPv6 in their core infrastructure.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About mGRE Tunnel Support over IPv6

mGRE Support over IPv6

Multiple sites of a DMVPN are interconnected by IPv6. A single logical mGRE tunnel interface interconnects one VPN site to another. An IPv6 subnet connects a tunnel interface with other tunnel interfaces from various VPN sites. All tunnel interfaces connecting VPN sites act as hosts on the logical IPv6 subnet. This structure is referred to as the tunnel overlay network.

mGRE Tunnels Support over IPv6

To enable service providers deploy IPv6 in their core infrastructure, multipoint generic routing encapsulation (mGRE) tunnels over IPv6 are supported. Dynamic Multipoint Virtual Private Network (DMVPN) customers may run either IPv4 or IPv6 in their local networks, so the overlay endpoints can be either IPv4 or IPv6. For an IPv6 transport endpoint, the overlay endpoint can either be an IPv4 or IPv6 private network address.

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 a passenger protocol, which allows both IS-IS and IPv6 traffic to run over the same tunnel. If GRE did not have a protocol field, it would be impossible to distinguish whether the tunnel was carrying IS-IS or IPv6 packets. The GRE protocol field is why it is desirable that you tunnel IS-IS and IPv6 inside GRE.

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

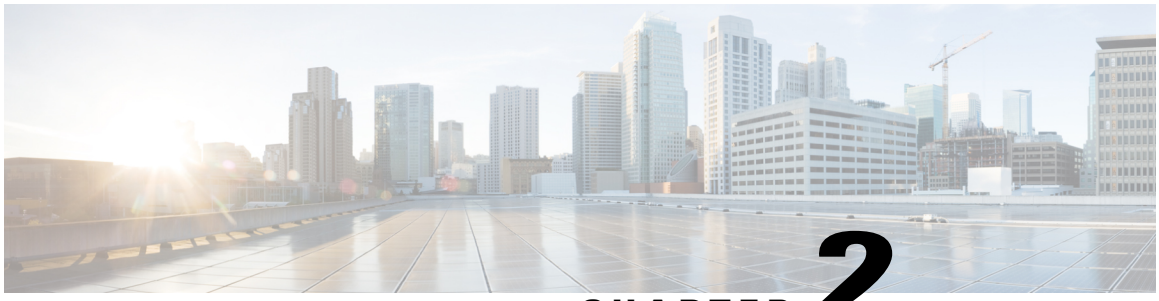
Feature Information for mGRE Tunnel Support over IPv6

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 1: Feature Information for mGRE Tunnel Support over IPv6

Feature Name	Releases	Feature Information
mGRE Tunnel Support over IPv6	15.2(1)T XE Release 3.8S	mGRE tunnels are configured to enable service providers deploy IPv6 in their core infrastructure.



CHAPTER 2

IP over IPv6 Tunnels

IPv6 supports IP over IPv6 tunnels, which includes the following:

- Generic routing encapsulation (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. 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.
- GRE support over IPv6 transport—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 a passenger protocol, which allows both IS-IS and IPv6 traffic to run over the same tunnel.
- VRF-aware IPv4/IPv6 over IPv6 tunnels - Virtual Routing and Forwarding (VRF)-aware tunnels are used to connect customer networks separated by untrusted core networks or core networks with different infrastructures (IPv4 or IPv6).

Information About IP over IPv6 Tunnels

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.

GRE Support over IPv6 Transport

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 a passenger protocol, which allows both IS-IS and IPv6 traffic to run over the same tunnel. If GRE did not have a protocol field, it would be impossible to distinguish whether the tunnel was carrying IS-IS or IPv6 packets. The GRE protocol field makes it desirable to tunnel IS-IS and IPv6 inside GRE.

How to Configure IP over IPv6 Tunnels

The following tasks describe how to configure an IPv6 tunnel. IPv6 or IPv4 packets can be forwarded on this tunnel.

Configuring 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 (this is not shown in the task below). The host or device at each end of the configured tunnel must support both 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. **end**

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(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:** `Device(config-if)# tunnel source ethernet 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, the interface must be configured with an IPv6 address. **Note** Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.
Step 5	**tunnel destination** *ipv6-address* **Example:** `Device(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 IPv6 Command Reference.
Step 6	**tunnel mode gre ipv6** **Example:** `Device(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 interface. Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for IP over IPv6 Tunnels

Example: IPv6 over IPv6 Tunnel

Example: Configuring CE1

```
!
ipv6 unicast-routing
ipv6 cef
```

```
!
interface Ethernet0/0
 no ipv6 address
 ipv6 address 2001:DB8:2:1::1/64
 no shutdown
 exit
!

ipv6 route 2001:DB8:2:5::/64 2001:DB8:2:1::2
ipv6 route 2001:DB8:2:9::/64 2001:DB8:2:1::2
!
```

Example: Configuring PE1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Tunnel0
 no ipv6 address
 ipv6 address 2001:DB8:2:9::1/64
 tunnel source 2001:DB8:2:2::1
 tunnel mode ipv6
 tunnel destination 2001:DB8:2:4::2
 exit
!
interface Ethernet0/0
 no ipv6 address
 ipv6 address 2001:DB8:2:1::2/64
 no shutdown
 exit
!
!
interface Ethernet1/1
 no ipv6 address
 ipv6 address 2001:DB8:2:2::1/64
 no shutdown
 exit
!

ipv6 route 2001:DB8:2:3::/64 2001:DB8:2:2::2
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:2::2
ipv6 route 2001:DB8:2:5::/64 Tunnel0 2001:DB8:2:9::2
```

Example: Configuring PE2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Tunnel0
 no ipv6 address
 ipv6 address 2001:DB8:2:9::2/64
 tunnel source 2001:DB8:2:4::2
 tunnel mode ipv6
 tunnel destination 2001:DB8:2:2::1
 exit
!
interface Ethernet0/0
 no ipv6 address
 ipv6 address 2001:DB8:2:5::1/64
 no shutdown
 exit
!
interface Ethernet0/1
 no ipv6 address
```

```
 ipv6 address 2001:DB8:2:4::2/64
 no shutdown
 exit
!
!
ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:4::1
ipv6 route 2001:DB8:2:3::/64 2001:DB8:2:4::1
ipv6 route 2001:DB8:2:1::/64 Tunnel0 2001:DB8:2:9::1
```

Example: Configuring CE2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/0
 no ipv6 address
 ipv6 address 2001:DB8:2:5::2/64
 no shutdown
 exit
!
ipv6 route 2001:DB8:2:1::/64 2001:DB8:2:5::1
ipv6 route 2001:DB8:2:9::/64 2001:DB8:2:5::1
!
```

Example: Configuring Core Device 1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet1/0
 no ipv6 address
 no shutdown
 ipv6 address 2001:DB8:2:3::1/64
 exit
!
interface Ethernet1/1
 no ipv6 address
 ipv6 address 2001:DB8:2:2::2/64
 no shutdown
 exit
!
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:3::2
```

Example: Configuring Core Device 2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/1
 no ip address
 ipv6 address 2001:DB8:2:4::1/64
 no shutdown
 exit
!
interface Ethernet1/0
 no ip address
 ipv6 address 2001:DB8:2:3::2/64
 no shutdown
 exit
!
ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:3::1
```

Example: IPv4 over IPv6 Tunnel

Example: Configuring CE1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/0
 no ip address
 ip address 192.168.1.1 255.255.255.0
 no shutdown
 exit
!
ip route 192.168.5.0 255.255.255.0 192.168.1.2
ip route 192.168.9.0 255.255.255.0 192.168.1.2
!
```

Example: Configuring PE1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Tunnel0
 no ip address
 ip address 192.168.9.1 255.255.255.0
 tunnel source 2001:DB8:2:2::1
 tunnel destination 2001:DB8:2:4::2
 tunnel mode ipv6
 exit
!
interface Ethernet0/0
 no ip address
 ip address 192.168.1.2 255.255.255.0
 no shutdown
 exit
!
!
interface Ethernet1/1
 no ipv6 address
 ipv6 address 2001:DB8:2:2::1/64
 no shutdown
 exit
!

ipv6 route 2001:DB8:2:3::/64 2001:DB8:2:2::2
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:2::2
ip route 192.168.5.0 255.255.255.0 Tunnel 0 192.168.9.2
```

Example: Configuring PE2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Tunnel0
 no ip address
```

```
 ip address 192.168.9.2 255.255.255.0
 tunnel source 2001:DB8:2:4::2
 tunnel destination 2001:DB8:2:2::1
 tunnel mode ipv6
 exit
!
interface Ethernet0/0
 no ip address
 ip address 192.168.5.1 255.255.255.0
 no shutdown
 exit
!
interface Ethernet0/1
 no ipv6 address
 ipv6 address 2001:DB8:2:4::2/64
 no shutdown
 exit
!
!
ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:4::1
ipv6 route 2001:DB8:2:3::/64 2001:DB8:2:4::1
ip route 192.168.1.0 255.255.255.0 Tunnel 0 192.168.9.1
```

Example: Configuring CE2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/0
 no ip address
 ip address 192.168.5.2 255.255.255.0
 no shutdown
 exit
!
ip route 192.168.1.0 255.255.255.0 192.168.1.2
ip route 192.168.9.0 255.255.255.0 192.168.1.2

!
```

Example: Configuring Core Device 1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet1/0
 no ipv6 address
 no shutdown
 ipv6 address 2001:DB8:2:3::1/64
 exit
!
interface Ethernet1/1
 no ipv6 address
 ipv6 address 2001:DB8:2:2::2/64
 no shutdown
 exit
!
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:3::2
```

Example: Configuring Core Device 2

```
!
```

```
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/1
 no ip address
 ipv6 address 2001:DB8:2:4::1/64
 no shutdown
 exit
!
interface Ethernet1/0
 no ip address
 ipv6 address 2001:DB8:2:3::2/64
 no shutdown
 exit
!
ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:3::1
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFC*

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

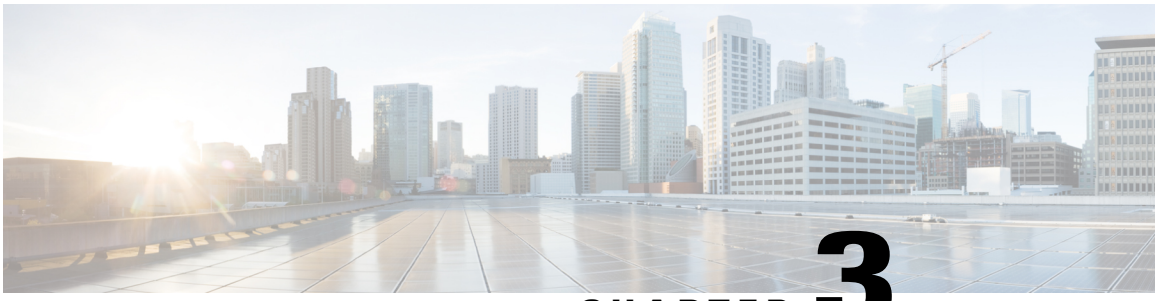
Feature Information for IP over IPv6 Tunnels

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 2: Feature Information for IP over IPv6 Tunnels

Feature Name	Releases	Feature Information
IP over IPv6 Tunnels	12.2(30)S 12.2(33)SRA 12.3(7)T 12.4 12.4(2)T 15.0(1)S Cisco IOS XE Release 2.1 15.1(1)SY Cisco IOS XE Release 3.2SE	IP over IPv6 Tunnels feature is supported. The following commands were introduced or modified: **tunnel destination**, **tunnel mode ipv6**, **tunnel mode gre ipv6**, **tunnel source**.



CHAPTER 3

Manually Configured IPv6 over IPv4 Tunnels

This feature provides support for manually configured IPv6 over IPv4 tunnels. A manually configured tunnel is equivalent to a permanent link between two IPv6 domains over an IPv4 backbone.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

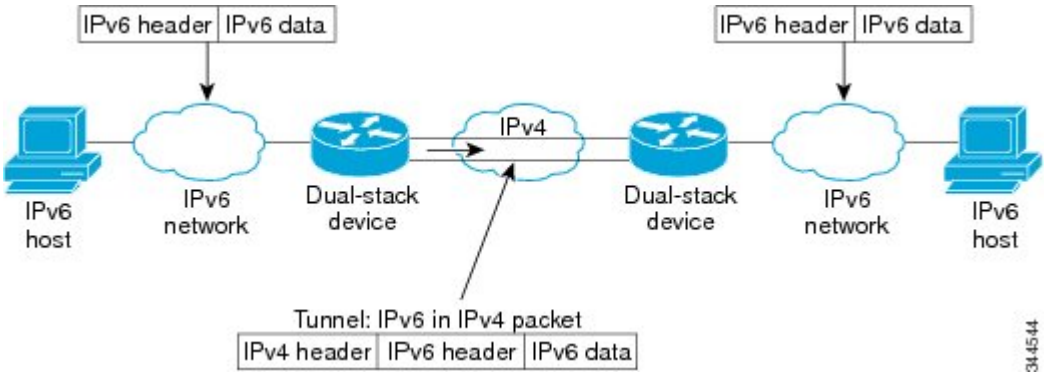
Information About Manually Configured IPv6 over IPv4 Tunnels

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 1: 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 3: 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.

Tunneling Type	Suggested Usage	Usage Notes
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.
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 4: Tunnel Configuration Parameters by Tunneling Type

Tunneling Type	Tunnel Configuration Parameter		
Tunnel Mode	**Tunnel Source**	**Tunnel Destination**	**Interface Prefix or Address**

Tunneling Type	Tunnel Configuration Parameter			
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.

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 is for stable connections that require regular secure communication between two edge devices or between an end system and an edge device, or for connection to remote IPv6 networks.

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

How to Enable Manually Configured IPv6 over IPv4 Tunnels

Configuring Manual IPv6 Tunnels

Before You Begin

With manually configured IPv6 tunnels, an IPv6 address is configured on a tunnel interface, and manually configured IPv4 addresses are assigned to the tunnel source and the tunnel destination. The host or device 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. Enter one of the following commands:
   - **ipv6 address** {*ipv6-address*/*prefix-length* | *prefix-name sub-bits*/*prefix-length*}
   - **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:** `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 0`	Specifies a tunnel interface and number, and enters interface configuration mode.

	Command or Action	Purpose
Step 4	Enter one of the following commands: • **ipv6 address** {*ipv6-address*/*prefix-length* | *prefix-name sub-bits*/*prefix-length*} • **ipv6 address** *ipv6-prefix*/*prefix-length* [**eui-64**] **Example:** `Device(config-if)# ipv6 address 3ffe:b00:c18:1::3/127`	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. • 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. **Note** See the "Implementing IPv6 Addressing and Basic Connectivity" module for more information on configuring IPv6 addresses.
Step 5	**tunnel source** {*ip-address* | *interface-type interface-number*} **Example:** `Device(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:** `Device(config-if)# tunnel destination 192.168.30.1`	Specifies the destination IPv4 address or hostname for the tunnel interface.
Step 7	**tunnel mode ipv6ip** **Example:** `Device(config-if)# tunnel mode ipv6ip`	Specifies a manual IPv6 tunnel. **Note** The **tunnel mode ipv6ip** command specifies IPv6 as the passenger protocol and IPv4 as both the encapsulation and transport protocol for the manual IPv6 tunnel.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuration Examples for Manually Configured IPv6 over IPv4 Tunnels

Example: Configuring Manual IPv6 Tunnels

The following example configures a manual 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 Configuration

```
interface gigabitethernet 0/0/0
 ip address 192.168.99.1 255.255.255.0
interface tunnel 0
 ipv6 address 3ffe:b00:c18:1::3/127
 tunnel source gigabitethernet 0/0/0
 tunnel destination 2001:DB8:1111:2222::1/64
 tunnel mode ipv6ip
```

Router B Configuration

```
interface gigabitethernet 0/0/0
 ip address 192.168.30.1 255.255.255.0
interface tunnel 0
 ipv6 address 3ffe:b00:c18:1::2/127
 tunnel source gigabitethernet 0/0/0
 tunnel destination 2001:DB8:1111:2222::2/64
 tunnel mode ipv6ip
```

Example: IPv6 over GRE IPv4 Tunnel

Example: Configuring CE1

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Ethernet0/0
 no ip address
 ipv6 address 2001:DB8:2:1::1/64
 no shutdown
 exit
!
!
ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:1::2
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:1::2
!
```

Example: Configuring PE1

```
ipv6 unicast-routing
ipv6 cef
!
```

```
interface Tunnel0
 no ip address
 ipv6 address 2001:DB8:2:4::1/64
 tunnel source 10.22.22.22
 tunnel destination 10.44.44.44
 exit
!
interface Ethernet0/0
 no ip address
 ipv6 address 2001:DB8:2:1::2/64
 no shutdown
 exit
!
interface Ethernet1/1
 no ip address
 ip address 10.22.22.22 255.255.255.0
 no shutdown
 exit
!
ip route 10.44.44.0 255.255.255.0 10.22.22.23
ipv6 route 2001:DB8:2:2::/64 Tunnel0 2001:DB8:2:4::2
```

Example: Configuring PE2

```
!
ipv6 unicast-routing
ipv6 cef
!
interface Tunnel0
 no ipv6 address
 ipv6 address 2001:DB8:2:4::2/64
 tunnel source 10.44.44.44
 tunnel destination 10.22.22.22
 exit
!
interface Ethernet0/0 no ipv6 address
 ipv6 address 2001:DB8:2:2::1/64
 no shutdown
 exit
!
interface Ethernet1/0
 no ip address
 ip address 10.44.44.44 255.255.255.0
 no shutdown
 exit
!
ip route 10.22.22.0 255.255.255.0 10.44.44.43
!
ipv6 route 2001:DB8:2:1::/64 Tunnel0 2001:DB8:2:4::1
!
```

Example: Configuring CE2

```
!
ipv6 unicast-routing
ipv6 cef
!
!
interface Ethernet0/0
 no ipv6 address
 ipv6 address 2001:DB8:2:2::2/64
 no shutdown
 exit
!
!
ipv6 route 2001:DB8:2:1::/64 2001:DB8:2:2::1
ipv6 route 2001:DB8:2:4::/64 2001:DB8:2:2::1
```

```
!
```

Example: Configuring Device X

```
!
interface Ethernet1/0
 no ip address
 ip address 10.44.44.43 255.255.255.0
 no shutdown
 exit
!
interface Ethernet1/1
 no ip address
 ip address 10.22.22.23 255.255.255.0
 no shutdown
 exit
!
```

Example: Verifying the Tunnel Configuration

From CE1

```
Device# ping ipv6 2001:db8:2:2::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/9/43 ms

Device# ping ipv6 2001:db8:2:2::2 source 2001:db8:2:1::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:2:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

From PE1

```
Device# show tunnel interface

Tunnel0
   Mode:GRE/IP, Destination 10.44.44.44, Source 10.22.22.22
   IP transport: output interface Ethernet1/1 next hop 10.22.22.23
   Application ID 1: unspecified
   Linestate - current up
   Internal linestate - current up, evaluated up
   Tunnel Source Flags: Local
   Transport IPv4 Header DF bit cleared
   OCE: IP tunnel decap
   Provider: interface Tu0, prot 47
     Performs protocol check [47]
     Protocol Handler: GRE: opt 0x0
       ptype: ipv4 [ipv4 dispatcher: punt]
       ptype: ipv6 [ipv6 dispatcher: from if Tu0]
       ptype: mpls [mpls dispatcher: drop]
       ptype: otv [mpls dispatcher: drop]
       ptype: generic [mpls dispatcher: drop]
There are 0 tunnels running over the EON IP protocol
There are 0 tunnels running over the IPinIP protocol
There are 0 tunnels running over the NOSIP protocol
There are 0 tunnels running over the IPv6inIP protocol
There are 0 tunnels running over the RBSCP/IP protocol

Device# show ip route 10.44.44.44
```

```
Routing entry for 10.44.44.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.22.22.23
      Route metric is 0, traffic share count is 1

Device# debug ipv6 icmp

  ICMP Packet debugging is on
*Jan  1 10:57:37.882: ICMPv6: Sent R-Advert, Src=FE80::A8BB:CCFF:FE00:5200, Dst=FF02::1
*Jan  1 11:00:18.634: ICMPv6: Received R-Advert, Src=FE80::A8BB:CCFF:FE00:5200,Dst=FF02::1
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

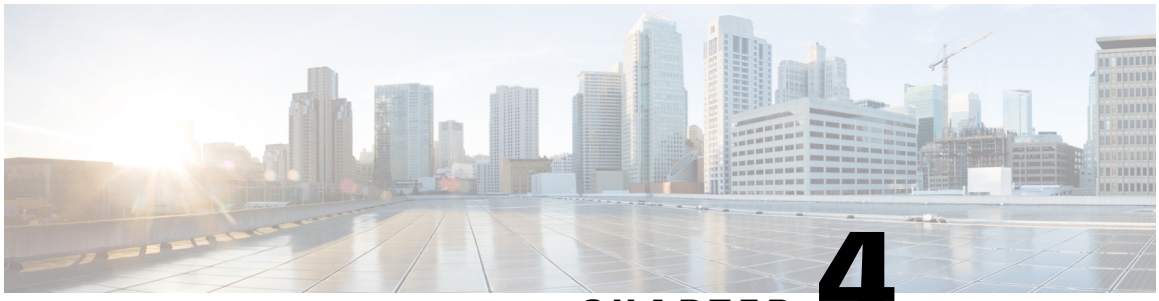
Feature Information for Manually Configured IPv6 over IPv4 Tunnels

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 Manually Configured IPv6 over IPv4 Tunnels

Feature Name	Releases	Feature Information
IPv6 Tunneling: Manually Configured IPv6 over IPv4 Tunnels	Cisco IOS XE Release 2.1	A manually configured tunnel is equivalent to a permanent link between two IPv6 domains over an IPv4 backbone. The following commands were introduced or modified: **tunnel destination**, **tunnel ipv6ip**, **tunnel source**.



CHAPTER 4

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 caveats and feature information, see Bug Search Tool and 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 module.

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:

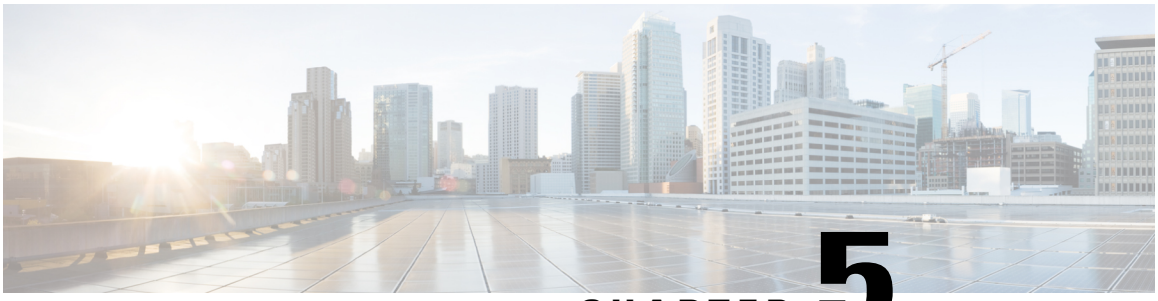
http://cisco.com/en/US/docs/interfaces_modules/shared_port_adapters/configuration/ASR1000/ASRspasw.html

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 .



CHAPTER 5

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 caveats and feature information, see Bug Search Tool and 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 module.

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.

Before You Begin

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.

	Command or Action	Purpose
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.
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

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.

You can configure only one null interface on a 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:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface null** *number* **Example:** `Device(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:** `Device(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:** `Device(config-if)# end`	Exits to privileged EXEC mode.
Step 6	**show interfaces null** [*number*] [**accounting**] **Example:** `Device# 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.

```
Device# 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.

Before You Begin

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

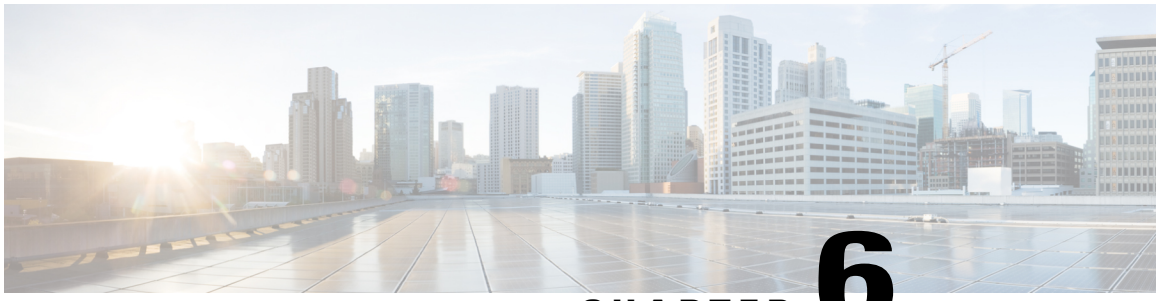
MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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



CHAPTER 6

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

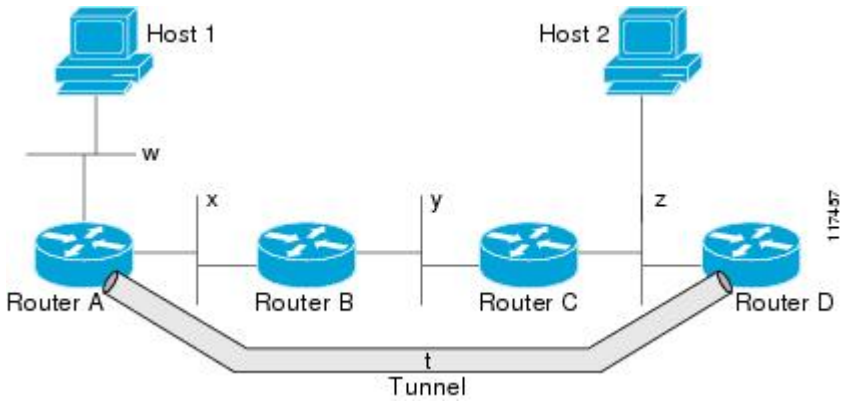
Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the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

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 2: 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.

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.

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" 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.

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 60" 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 6: 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

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.

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**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• 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.
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*}	Configures the tunnel destination. • Use the *hostname* argument to specify the name of the host destination.

	Command or Action	Purpose
	Example: `Router(config-if)# tunnel destination 10.0.2.1`	• 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.
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.

	Command or Action	Purpose
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 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.

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.

	Command or Action	Purpose
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.
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

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.

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 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
```

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
```

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*

Related Topic	Document Title
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.	--
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*

Standard	Title
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

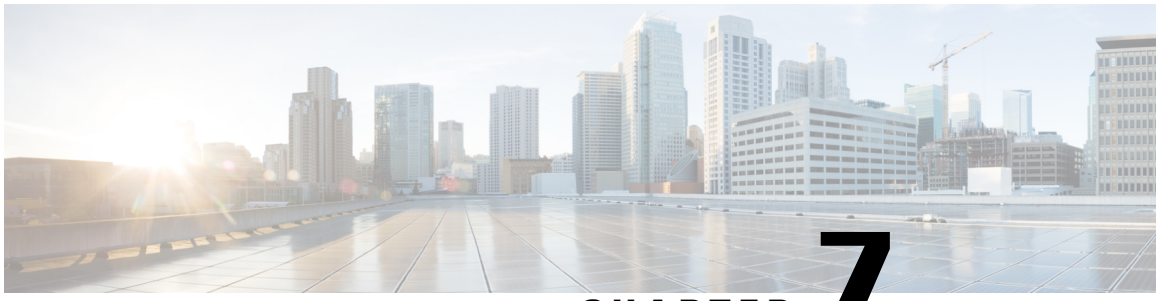
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train 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 7: 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**.
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**.

Feature Name	Releases	Feature Information
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**.



CHAPTER 7

Tunnel Route Selection

The Tunnel Route Selection feature allows the tunnel transport to be routed using a subset of the routing table. When there are equal-cost routes to a tunnel destination, normal tunnel transport behavior is to use one of the available routes chosen at random. The Tunnel Route Selection feature allows the explicit configuration of the outgoing interface for the tunnel transport.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 Tunnel Route Selection

Tunnel interfaces are configured.

Restrictions for Tunnel Route Selection

This feature is supported in the following tunnel modes only:

- Generic Routing Encapsulation (GRE) IP
- GRE Multipoint
- IP in IP
- Mobile User Datagram Protocol (UDP)

This feature is not supported on a tunnel when the tunnel transport is a GRE Multipoint tunnel.

Supported Configuration

```
interface tunnel 0
  tunnel mode gre multipoint
  tunnel route-via tunnel 1
interface tunnel 1
   tunnel mode gre ip
```

Unsupported Configuration

```
interface tunnel 0
  tunnel mode gre multipoint
  tunnel route-via tunnel 1
interface tunnel 1
  tunnel mode gre multipoint
```

Information About Tunnel Route Selection

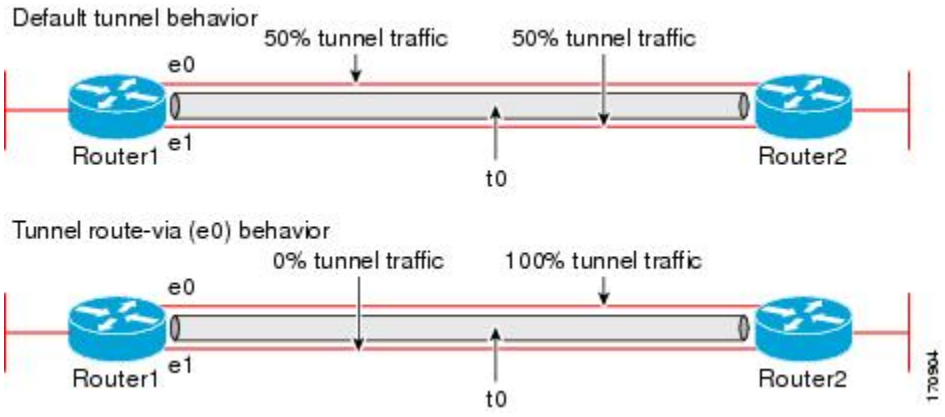
Tunnel Transport Behavior

The Tunnel Route Selection feature allows the tunnel transport to be routed using a subset of the routing table by specifying the outgoing interface of the tunnel transport.

The Tunnel Route Selection feature is not the same as an implementation of policy-based routing for the tunnel transport. The Tunnel Route Selection feature will forward traffic using only a subset of the route table, and it cannot introduce routing loops into the network.

The figure below compares default tunnel behavior with the Tunnel Route Selection behavior.

Figure 3: Tunnel Route Selection Traffic



How to Configure Tunnel Route Selection

Configuring Tunnel Route Selection

Perform the following steps to specify the outgoing interface of the tunnel transport to route the tunnel transport using a subset of the routing table.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *interface-number*
4. **tunnel route-via** *interface-type interface-number* {**mandatory** | **preferred**}
5. **end**

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 tunnel** *interface-number* **Example:** `Router(config)# interface tunnel 0`	Configures a tunnel interface and enters interface configuration mode.
Step 4	**tunnel route-via** *interface-type interface-number* {**mandatory** | **preferred**} **Example:** `Router(config-if)# tunnel route-via ethernet0 mandatory`	Specifies the outgoing interface to be used by the tunnel transport.
Step 5	**end** **Example:** `Router(config-if)# end`	Returns to privileged EXEC mode.

Troubleshooting Tips

To troubleshoot your configuration, use the **debug tunnel route-via** command in privileged EXEC mode. The following is sample output from the **debug tunnel route-via** command after the **tunnel route-via** command was used to route the tunnel transport explicitly using a subset of the routing table.

```
Router# debug tunnel route-via
Tunnel route-via debugging is on
Router#
*May 23 08:40:53.707: TUN-VIA: Tunnel0 candidate route-via Ethernet0/0, next hop 10.73.2.1
*May 23 08:40:53.707: TUN-VIA: Tunnel0 route-via action is forward
*May 23 08:41:03.719: TUN-VIA: Tunnel0 candidate route-via Ethernet0/0, next hop 10.73.2.1
*May 23 08:41:03.719: TUN-VIA: Tunnel0 route-via action is forward
Router# undebug tunnel route-via
Tunnel route-via debugging is off
```

What to Do Next

You can verify the tunnel route selection configuration. To verify your configuration, use the **show interfaces tunnel** command in privileged EXEC mode. The following example shows that the tunnel transport is routed using a subset of the routing table by specifying the outgoing interface of the tunnel transport.

```
Router# show running-config interface tunnel 0
Building configuration...
```

```
Current configuration : 147 bytes
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel source Loopback0
 tunnel destination 10.73.0.102
 tunnel route-via Ethernet0 preferred
end
Router# show interfaces tunnel 0 | include route-via
```
Tunnel route-via feature is on [Ethernet0, preferred]

Configuration Examples for Tunnel Route Selection

Example Configuring Tunnel Route Selection

The following example shows Tunnel 0 configured to use Ethernet interface 0 as its preferred outgoing transport interface. Traffic that exits the router using the tunnel 0 interface will be sent out of Ethernet interface 0 if there is a route to the tunnel destination out of Ethernet interface 0. If there is no route out of Ethernet interface 0, the traffic will be forwarded as if the Tunnel Route Selection feature were not configured.

If the **tunnel route-via** *interface-type interface-number* **mandatory** command is configured, and there is no route to the tunnel destination using that interface, a point-to-point tunnel interface will go into a down state.

```
Router> enable
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# interface tunnel 0
Router(config-if)# tunnel route-via ethernet0 preferred
Router(config-if)# end
Router# show running-config interface tunnel 0
Building configuration...
Current configuration : 147 bytes
!
interface Tunnel0
 ip unnumbered Loopback0
 tunnel source Loopback0
 tunnel destination 10.73.0.102
 tunnel route-via Ethernet0 preferred
end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Interface commands: **define interface-range**, **interface range**, and **interface vlan**.	Cisco IOS Interface and Hardware Component Command Reference
Configuration commands: **show running-config**.	Cisco IOS Configuration Fundamentals Command Reference

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

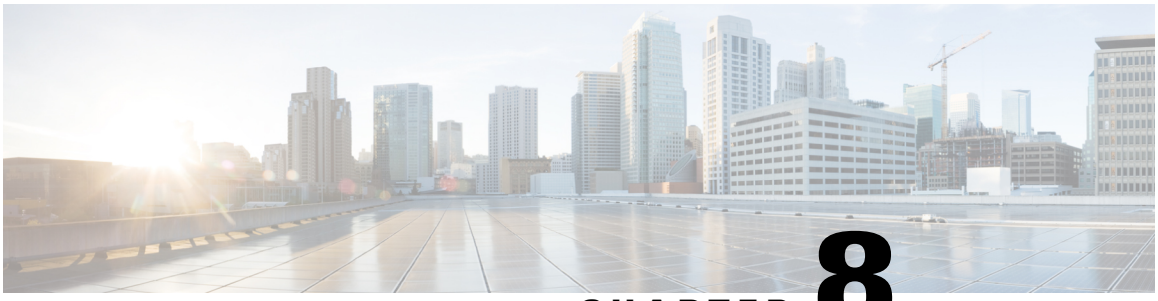
Feature Information for Tunnel Route Selection

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 8: Feature Information for Tunnel Route Selection

Feature Name	Releases	Feature Information
Tunnel Route Selection	12.4(11)T 15.0(1)M Cisco IOS Release 3.9S	The Tunnel Route Selection feature allows the tunnel transport to be routed using a subset of the routing table. When there are equal-cost routes to a tunnel destination, normal tunnel transport behavior is to use one of the available routes chosen at random. The Tunnel Route Selection feature allows the explicit configuration of the outgoing interface for the tunnel transport. The following commands were introduced or modified: **debug tunnel route-via, tunnel route-via, show interfaces tunnel**.



CHAPTER 8

MPLS VPN over mGRE

The MPLS VPN over mGRE feature overcomes the requirement that a carrier support multiprotocol label switching (MPLS) by allowing you to provide MPLS connectivity between networks that are connected by IP-only networks. This allows MPLS label switched paths (LSPs) to use generic routing encapsulation (GRE) tunnels to cross routing areas, autonomous systems, and internet service providers (ISPs). when MPLS VPNs are configured over multipoint GRE (mGRE) you can deploy layer-3 (L3) provider edge (PE) based virtual private network (VPN) services using a standards-based IP core. This allows you to provision the VPN services without using the overlay method.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 MPLS VPN over mGRE

Before you configure MPLS VPN with mGRE tunnels, ensure that the MPLS VPN is configured and working properly. See the "Configuring MPLS Layer 3 VPNs" module for information about setting up MPLS VPNs.

Restrictions for MPLS VPN over mGRE

- Tunnelled tag traffic must enter the router through a line card that supports MPLS VPN over mGRE.
- Each PE router supports one tunnel configuration only.
- MPLS VPN over mGRE does not support the transportation of multicast traffic between VPNs.
- When a GRE tunnel has the same destination address and source address as the mGRE, the tunnel gets route-cache switched.
- The packets that require fragmentation get route cache-switched.
- When an L3VPN profile is removed and added back, then you should clear the Border Gateway Protocol (BGP) using the **clear ip bgp soft**command.
- When an mGRE tunnel is created, a dummy tunnel is also created.
- The loopback or IP address used in the update source of the BGP configuration should be the same as that of the transport source of the L3VPN profile.
- mGRE is not stateful switchover (SSO) compliant. However, both mGRE and SSO coexist.
- mGRE and multicast distribution tree (MDT) tunnel should not be configured with the same loopback address.

The limitations for MPLS VPN over mGRE feature are as follows:

-
  - Not all GRE options are supported in the hardware (for example, GRE extended header and GRE key).
  - Checking identical VLANs (Internet Control Message Protocol [ICMP] redirect) is not supported on the tunnels.
  - Features such as unicast reverse path forwarding (uRPF) and BGP policy accounting are not supported on the tunnels.

Information About MPLS VPN over mGRE

You can configure mGRE tunnels to create a multipoint tunnel network that overlays an IP backbone. This overlay connects PE routers to transport VPN traffic.

In addition, when MPLS VPNs are configured over mGRE you can deploy L3 PE-based VPN services using a standards-based IP core. This allows you to provision the VPN services without using the overlay method. When MPLS VPN over mGRE is configured, the system uses IPv4-based mGRE tunnels to encapsulate VPN-labeled IPv4 and IPv6 packets between PEs. To deploy MPLS VPN over mGRE tunnels, you create a

VRF instance, enable and configure L3 VPN encapsulation, link the route map to the application template, and set up the BGP VPNv4 and VPNv6 exchange so that updates are filtered through the route map.

MPLS VPN over mGRE

GRE is a point-to-point tunneling protocol where two peers form the endpoints of the tunnel. It is designed to encapsulate network-layer packets inside IP tunneling packets. mGRE is a similar protocol with a single endpoint at one side of the tunnel connected to multiple endpoints at the other side of the tunnel. The mGRE tunnel provides a common link between branch offices that connect to the same VPN. Because mGRE is a point-to-multipoint model, fully meshed GRE tunnels are not required to interconnect MPLS VPN PE devices.

MPLS is a widely deployed VPN internet architecture. MPLS requires that all core routers in the network support MPLS. This feature is useful in networks where the service provider uses a backbone carrier to provide connectivity.

The MPLS VPN over mGRE feature overcomes the requirement of carrier support MPLS by allowing you to provide MPLS connectivity between networks that are connected by IP-only networks. This allows MPLS LSPs to use GRE tunnels to cross routing areas, autonomous systems, and ISPs.

When MPLS VPNs are configured over mGRE you can deploy L3 PE-based VPN services using a standards-based IP core. This allows you to provision the VPN services without using LSP or a Label Distribution Protocol (LDP). The system uses IPv4-based mGRE tunnels to encapsulate VPN-labeled IPv4 and IPv6 packets between PEs.

The MPLS VPN over mGRE feature also allows you to deploy existing MPLS VPN LSP-encapsulated technology concurrently with MPLS VPN over mGRE and enables the system to determine which encapsulation method is used to route specific traffic. The ingress PE router determines which encapsulation technology to use when a packet is sent to the remote PE router.

This section includes information on the following topics on MPLS VPN over mGRE feature:

Route Maps

By default, VPN traffic is sent using an LSP. The MPLS VPN over mGRE feature uses user-defined route maps to determine which VPN prefixes are reachable over an mGRE tunnel and which VPN prefixes are reachable using an LSP. The route map is applied to advertisements for VPNv4 and VPNv6 address families. The route map uses a next hop tunnel table to determine the encapsulation method for the VPN traffic.

To route traffic over the mGRE tunnel, the system creates an alternative address space that shows that all next hops are reached by encapsulating the traffic in an mGRE tunnel. To configure a specific route to use an mGRE tunnel, the user adds an entry for that route to the route map. The new entry remaps the Network Layer Reachability Information (NLRI) of the route to the alternative address space. If there is no remap entry in the route map for a route, then traffic on that route is forwarded over an LSP.

When the user configures MPLS VPN over mGRE, the system automatically provisions the alternative address space, normally held in the tunnel-encapsulated virtual routing and forwarding (VRF) instance. To ensure that all traffic reachable through the address space is encapsulated in an mGRE tunnel, the system installs a single default route out of a tunnel. The system also creates a default tunnel on the route map. The user can attach this default route map to the appropriate BGP updates.

Tunnel Endpoint Discovery and Forwarding

In order for the MPLS VPN over mGRE feature to function correctly, the system must be able to discover the remote PEs in the system and construct tunnel forwarding information for these remote PEs. In addition the system must be able to detect when a remote PE is no longer valid and remove the tunnel forwarding information for that PE.

If an ingress PE receives a VPN advertisement over BGP, it uses the route target attributes (which it inserts into the VRF) and the MPLS VPN label from the advertisement, to associate the prefixes with the appropriate customer. The next hop of the inserted route is set to the NLRI of the advertisement.

The advertised prefixes contain information about remote PEs in the system (in the form of NLRIs), and the PE uses this information to notify the system when an NLRI becomes active or inactive. The system uses this notification to update the PE forwarding information.

When the system receives notification of a new remote PE, it adds the information to the tunnel endpoint database, which causes the system to create an adjacency associated with the tunnel interface. The adjacency description includes information on the encapsulation and other processing that the system must perform to send encapsulated packets to the new remote PE.

The adjacency information is placed into the tunnel encapsulated VRF. When a user remaps a VPN NLRI to a route in the VRF (using the route map), the system links the NLRI to the adjacency; therefore the VPN is linked to a tunnel.

Tunnel Decapsulation

When the egress PE receives a packet from a tunnel interface that uses the MPLS VPN over mGRE feature, the PE decapsulates the packet to create a VPN label tagged packet, and sends the packet to the MPLS forwarding (MFI) code.

Tunnel Source

The MPLS VPN over mGRE feature uses a single tunnel configured as an mGRE tunnel to configure a system with a large number of endpoints (remote PEs). To identify the origin of tunnel-encapsulated packets, the system uses the tunnel source information.

At the transmitting (ingress) PE, when a VPN packet is sent to a tunnel, the tunnel destination is the NLRI. At a receiving (egress) PE, the tunnel source is the address that the packets encapsulated in the mGRE tunnel are received on. Therefore, at the egress PE the packet destination must match the NLRI from the local PE.

IPv6 VPN

If the advertising PE router has an IPv6 address then the NLRI must also be an IPv6 address (regardless of the network between the PEs). If the network between the PEs is IPv4 based, the system creates the IPv6 address of the advertising PE using an IPv4 mapped address in the following form: ::FFFF:IPv4-PE-address. The receiving PE sets the next hop for the VPN tag IPv6 prefixes to the IPv4 address embedded in the IPv6 NLRI. This enables the PE to link VPNv6 traffic to an LSP or an mGRE tunnel in the same way it maps VPNv4 traffic.

When a PE receives VPNv6 updates, it applies the IPv6 route map. The MPLS VPN over mGRE feature uses the IPv6 route map to set the next hop information in the Tunnel_Encap VRF.

How to Configure MPLS VPN over mGRE

Configuring an L3VPN Encapsulation Profile

This section describes how to configure an L3VPN encapsulation profile.

Note Transport protocols such as IPv6, MPLS, IP, and Layer 2 Tunneling Protocol version 3 (L2TPv3) can also be used in this configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **l3vpn encapsulation ip** *profile-name*
4. **transport ipv4 [source** *interface-type interface-number* **]**
5. **protocol gre [key** *gre-key* **]**
6. **end**
7. **show l3vpn encapsulation ip** *profile-name*

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	**l3vpn encapsulation ip** *profile-name* **Example:** `Router(config)# l3vpn encapsulation ip tunnel encap`	Enters L3 VPN encapsulation configuration mode to create the tunnel.
Step 4	**transport ipv4 [source** *interface-type interface-number* **]**	(Optional) Specifies IPv4 transport source mode and defines the transport source interface.

	Command or Action	Purpose
	Example: `Router(config-l3vpn-encap-ip)# transport ipv4 source loopback 0`	• If you use the **transport ipv4 source** *interface-type interface-number* command, make sure that the specified source address is used as the next hop in BGP updates advertised by the PE. • If you do not use this command, the **bgp update source**or **bgp next-hop** command is automatically used as the tunnel source.
Step 5	**protocol gre [key** *gre-key* **]** **Example:** `Router(config-l3vpn-encap-ip)# protocol gre key 1234`	Specifies GRE as the tunnel mode and sets the GRE key.
Step 6	**end** **Example:** `Router(config-l3vpn-encap-ip)# end`	Exits L3 VPN encapsulation configuration mode and returns to privileged EXEC mode.
Step 7	**show l3vpn encapsulation ip** *profile-name* **Example:** `Router# show l3vpn encapsulation ip tunnel encap`	(Optional) Displays the profile health and the underlying tunnel interface.

Configuring BGP and Route Maps

Perform this task to configure BGP and route maps. The following steps also enable you to link the route map to the application template and set up the BGP VPNv4 and VPNv6 exchange so that the updates are filtered through the route map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **bgp log-neighbor-changes**
5. **neighbor** *ip-address* **remote-as** *as-number*
6. **neighbor** *ip-address* **update-source** *interface name*
7. **address-family ipv4**
8. **no synchronization**
9. **redistribute connected**
10. **neighbor** *ip-address* **activate**
11. **no auto-summary**
12. **exit**
13. **address-family vpnv4**
14. **neighbor** *ip-address* **activate**
15. **neighbor** *ip-address* **send-community both**
16. **neighbor** *ip-address* **route-map** *map-name* **in**
17. **exit**
18. **address-family vpnv6**
19. **neighbor** *ip-address* **activate**
20. **neighbor** *ip-address* **send-community both**
21. **neighbor** *ip-address* **route-map** *map-name* **in**
22. **exit**
23. **route-map** *map-tag* **permit** *position*
24. **set ip next-hop encapsulate l3vpn** *profile-name*
25. **set ipv6 next-hop encapsulate l3vpn** *profile-name*
26. **exit**
27. **exit**

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	**router bgp** *as-number* **Example:** `Router(config)# router bgp 100`	Specifies the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along, and enters router configuration mode.
Step 4	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 5	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Router(config-router)# neighbor 209.165.200.225 remote-as 100`	Adds an entry to the BGP or multiprotocol BGP neighbor table.
Step 6	**neighbor** *ip-address* **update-source** *interface name* **Example:** `Router(config-router)# neighbor 209.165.200.225 update-source loopback 0`	Allows BGP sessions to use any operational interface for TCP connections.
Step 7	**address-family ipv4** **Example:** `Router(config-router)# address-family ipv4`	Enters address family configuration mode to configure routing sessions that use IPv4 address prefixes.
Step 8	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco software to advertise a network route without waiting for an IGP.
Step 9	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes routes from one routing domain into another routing domain and allows the target protocol to redistribute routes learned by the source protocol and connected prefixes on those interfaces over which the source protocol is running.

	Command or Action	Purpose
Step 10	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 activate`	Enables the exchange of information with a BGP neighbor.
Step 11	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 12	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.
Step 13	**address-family vpnv4** **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure routing sessions, such as BGP, that use standard VPNv4 address prefixes.
Step 14	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 activate`	Enables the exchange of information with a BGP neighbor.
Step 15	**neighbor** *ip-address* **send-community both** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 send-community both`	Specifies that a communities attribute, for both standard and extended communities, should be sent to a BGP neighbor.
Step 16	**neighbor** *ip-address* **route-map** *map-name* **in** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 route-map SELECT_UPDATE_FOR_L3VPN in`	Applies the named route map to the incoming route.
Step 17	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.

	Command or Action	Purpose
Step 18	**address-family vpnv6** **Example:** `Router(config-router)# address-family vpnv6`	Enters address family configuration mode to configure routing sessions, such as BGP, that use VPNv6 address prefixes.
Step 19	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 activate`	Enables the exchange of information with a BGP neighbor.
Step 20	**neighbor** *ip-address* **send-community both** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 send-community both`	Specifies that a communities attribute, for both standard and extended communities, should be sent to a BGP neighbor.
Step 21	**neighbor** *ip-address* **route-map** *map-name* **in** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 route-map SELECT_UPDATE_FOR_L3VPN in`	Applies the named route map to the incoming route.
Step 22	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.
Step 23	**route-map** *map-tag* **permit** *position* **Example:** `Router(config-router)# route-map SELECT_UPDATE_FOR_L3VPN permit 10`	Enters route-map configuration mode and defines the conditions for redistributing routes from one routing protocol into another. • The **redistribute** router configuration command uses the specified map tag to reference this route map. Multiple route maps may share the same map tag name. • If the match criteria are met for this route map, the route is redistributed as controlled by the set actions. • If the match criteria are not met, the next route map with the same map tag is tested. If a route passes none of the match criteria for the set of route maps sharing the same name, it is not redistributed by that set. • The *position* argument indicates the position a new route map will have in the list of route maps already configured with the same name.

	Command or Action	Purpose
Step 24	**set ip next-hop encapsulate l3vpn** *profile-name* **Example:** `Router(config-route-map)# set ip next-hop encapsulate l3vpn my profile`	Indicates that output IPv4 packets that pass a match clause of the route map are sent to the VRF for tunnel encapsulation.
Step 25	**set ipv6 next-hop encapsulate l3vpn** *profile-name* **Example:** `Router(config-route-map)# set ip next-hop encapsulate l3vpn tunnel encap`	Indicates that output IPv6 packets that pass a match clause of the route map are sent to the VRF for tunnel encapsulation.
Step 26	**exit** **Example:** `Router(config-route-map)# exit`	Exits route-map configuration mode and enters global configuration mode.
Step 27	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Configuration Examples for MPLS VPN over mGRE

Example Verifying the MPLS VPN over mGRE Configuration

Use the following examples to verify that the configuration is working properly:

Cisco Express Forwarding (CEF) Switching

You can verify that CEF switching is working as expected:

```
Router# show ip cef vrf Customer_A tunnel 0

209.165.200.250
/24
    nexthop 209.165.200.251 Tunnel0 label 16
```

Endpoint Creation

You can verify the tunnel endpoint that has been created:

```
Router# show tunnel endpoints tunnel 0
```

```
 Tunnel0 running in multi-GRE/IP mode
 Endpoint transport 209.165.200.251 Refcount 3 Base 0x2AE93F0 Create Time 00:00:42
   overlay 209.165.200.254 Refcount 2 Parent 0x2AE93F0 Create Time 00:00:42
```

Adjacency

You can verify that the corresponding adjacency has been created:

```
Router# show adjacency tunnel 0
   Protocol Interface                 Address
  IP       Tunnel0                   209.165.200.251(4)
  TAG      Tunnel0                   209.165.200.251(3)
```

Profile Health

You can use **show l3vpn encapsulation** *profile-name* command to get information on the basic state of the application. The output of this command provides you details on the references to the underlying tunnel.

```
Router# show l3vpn encapsulation ip tunnel encap
 Profile: tunnel encap
 transport ipv4 source Auto: Loopback0
 protocol gre
   Tunnel Tunnel0 Created [OK]
   Tunnel Linestate [OK]
   Tunnel Transport Source (Auto) Loopback0 [OK]
```

Example Configuration Sequence for MPLS VPN over mGRE

This example shows the configuration sequence for MPLS VPN over mGRE:

```
vrf definition Customer A
 rd 100:110
 route-target export 100:1000
 route-target import 100:1000
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
!
ip cef
!
ipv6 unicast-routing
ipv6 cef
!
!
l3vpn encapsulation ip sample profile name
 transport source loopback 0
 protocol gre key 1234
!
!
 interface Loopback0
  ip address 209.165.200.252 255.255.255.224
  ip router isis
!
interface Serial2/0
 vrf forwarding Customer A
 ip address 209.165.200.253 255.255.255.224
 ipv6 address 3FFE:1001::/64 eui-64
 no fair-queue
 serial restart-delay 0
!
router bgp 100
```

```
 bgp log-neighbor-changes
 neighbor 209.165.200.254 remote-as 100
 neighbor 209.165.200.254 update-source Loopback0
 !
 address-family ipv4
  no synchronization
  redistribute connected
  neighbor 209.165.200.254 activate
  no auto-summary
 exit-address-family
 !
 address-family vpnv4
  neighbor 209.165.200.254 activate
  neighbor 209.165.200.254 send-community both
  neighbor 209.165.200.254 route-map SELECT_UPDATE_FOR_L3VPN in
 exit-address-family
 !
 address-family vpnv6
  neighbor 209.165.200.254 activate
  neighbor 209.165.200.254 send-community both
  neighbor 209.165.200.254 route-map SELECT_UPDATE_FOR_L3VPN in
 exit-address-family
 !
 address-family ipv4 vrf Customer A
  no synchronization
  redistribute connected
 exit-address-family
 !
 address-family ipv6 vrf Customer A
  redistribute connected
  no synchronization
 exit-address-family
!
!
route-map SELECT_UPDATE_FOR_L3VPN permit 10
set ip next-hop encapsulate sample profile name
set ipv6 next-hop encapsulate sample profile name
```

Additional References

Related Documents

Related Topic	Document Title
Configuring MPLS Layer 3 VPNs	*Cisco IOS XE Multiprotocol Label Switching Configuration Guide*
Cisco Express Forwarding	*Cisco IOS XE IP Switching Configuration Guide*
Generic routing encapsulation	*Cisco IOS XE Interface and Hardware Component Configuration Guide*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
IETF-PPVPN-MPLS-VPN-MIB	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
RFC 2547	*BGP/MPLS VPNs*
RFC 2784	*Generic Routing Encapsulation (GRE)*
RFC 2890	*Key Sequence Number Extensions to GRE*
RFC 4023	Encapsulating MPLS in IP or Generic Routing Encapsulation
RFC 4364	*BGP/MPLS IP Virtual Private Networks (VPNs)*

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

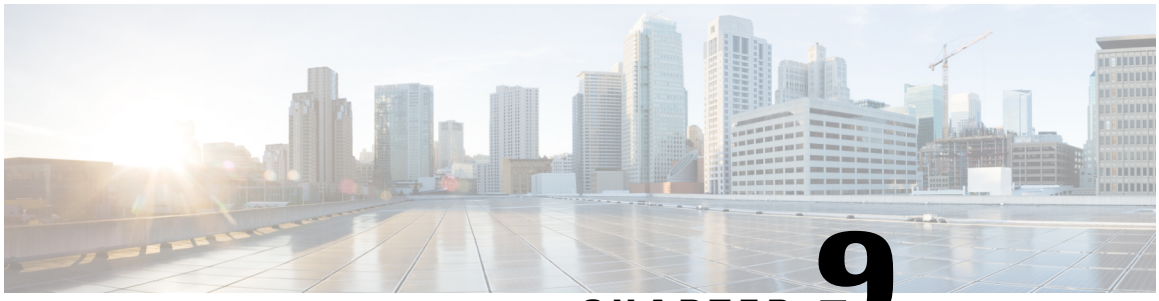
Feature Information for MPLS VPN over mGRE

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 9: Feature Information for MPLS VPN over mGRE

Feature Name	Releases	Feature Information
MPLS VPN over mGRE	Cisco IOS XE Release 3.1S	This feature provides support to carry MPLS Layer 3 VPN traffic over mGRE. The following commands were introduced or modified by this feature: l3vpn encapsulation ip, protocol gre, show l3vpn encapsulation ip, transport ipv4, set ip next-hop, set ipv6 next-hop.



CHAPTER 9

IP Tunnel MIBs

This module contains information about MIBs used with interfaces and hardware components. The IP Tunnel MIB feature provides a generic MIB for managing all IPv4- and IPv6-related tunnels, as outlined in RFC 4087, IP Tunnel MIB. Tunneling provides a way to encapsulate arbitrary packets inside a transport protocol. A number of tunneling mechanism s specified by Internet Engineering Task Force (IETF) are implemented by Cisco for both IPv4 and IPv6 environments. Various MIBs are available for managing tunnels.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 the IP Tunnel MIB

Configure Simple Network Management Protocol (SNMP) on the router on which the IP Tunnel MIB feature is to be used. See the Configuring the Router to Use SNMP, on page 92 for more information. For more information on configuring an SNMP server, see the "Configuring SNMP Support " chapter of the Cisco IOS Network Management Configuration Guide.

Restrictions for the IP Tunnel MIB

The IP Tunnel MIB feature supports only tunnels that can be created using the **interface tunnel** command. The IP Tunnel MIB feature does not support Layer 2 Tunnel Protocol (L2TP), Point-to-Point Tunneling Protocol (PPTP), or Multiprotocol Label Switching (MPLS) tunnels.

Information About the IP Tunnel MIB

Benefits of the IP Tunnel MIB

Improved Quality of Networks

Better IP tunnel instrumentation leads to an improvement in the quality of networks and better service delivery. A better quality network allows service providers to deliver a more reliable service.

Increased Reliability

The IP Tunnel MIB allows users of network management systems to set inventory and receive notification about their IP tunnel activity.

The IP Tunnel MIB supports both IPv4 and IPv6 network layers as defined in RFC 3291, and is used to manage IP tunnels implemented in the Cisco IOS software.

The IP Tunnel MIB supports all tunnel types, as well as tunnel creation and destruction capability.

Interoperability with Devices Other Than Cisco Devices

The IP Tunnel MIB works with key network management systems, including those of third-party vendors.

MIB Objects Supported by the IP Tunnel MIB

The following MIB objects are supported by the IP Tunnel MIB feature. For details regarding use of MIB objects, see RFC 4087, IP Tunnel MIB.

Table 10: Objects Supported by the IP Tunnel MIB

MIB Object	Description
tunnelIfEntry	Contains information on a particular configured tunnel. You can use the **interface tunnel** command to set a value for this object.
tunnelIfEncapsMethod	The encapsulation method used by the tunnel. You can use the **tunnel mode** command to set a value for this object.
tunnelIfHopLimit	Defines the IPv4 time to live (TTL) or IPv6 hop limit to use in the outer IP header. You can use the **tunnel ttl** command to set a value for this object.

MIB Object	Description
tunnelIfSecurity	Used by the tunnel to secure the outer IP header. The value ipsec indicates that IPsec is used between the tunnel endpoints for authentication or encryption, or both.
tunnelIfTOS	Used by the tunnel to set the high 6 bits (the differentiated services codepoint) of the IPv4 type of service (ToS) or IPv6 traffic class in the outer IP header. You can use the **tunnel tos** command to set a value for this object.
tunnelIfFlowLabel	Used to set the IPv6 Flow Label value. This object is supported for tunnels over IPv6. The default value for this object is 0.
tunnelIfAddressType	Shows the type of address in the corresponding tunelIfLocalInetAddress and tunnelIfRemoteInetAddress objects. This object cannot be configured individually through the command-line interface (CLI).
tunnelIfLocalInetAddress	The address of the local endpoint of the tunnel (that is, the source address used in the outer IP header). If the address is unknown, the value is 0.0.0.0 for IPv4 or :: for IPv6. The address type of this object is given by tunnelIfAddressType. You can use the **tunnel source** command to set a value for this object.
tunnelIfRemoteInetAddress	The address of the remote endpoint of the tunnel (that is, the destination address used in the outer IP header). If the address is unknown or the tunnel is not a point-to-point link (for example, a 6-to-4 tunnel), the value is 0.0.0.0 for tunnels over IPv4 or :: for tunnels over IPv6. The address type of this object is given by tunnelIfAddressType. You can use the **tunnel destination** command to set a value for this object.
tunnelIfEncapsLimit	Shows the maximum number of additional encapsulations permitted for packets undergoing encapsulation at this node. A value of -1 indicates that no limit is present (except as result of packet size).
tunnelInetConfigEntry	Contains information on a particular configured tunnel. There will be only one entry for multipoint tunnels and for tunnels that have the remote inet address 0.0.0.0 for IPv4 or :: for IPv6. Only generic routing encapsulation (GRE)/IP and GRE/IPv6 tunnels are created through the MIB.
tunnelInetConfigIfIndex	Shows the value of ifIndex corresponding to the tunnel interface. A value of 0 is not legal in the active state and means that the interface index has not yet been assigned.
tunnelInetConfigStatus	Used to create or delete table entries in the MIB table. You can use the **interface tunnel** to set a value for this object.

MIB Object	Description
tunnelInetConfigStorageType	Indicates the storage type. Only a nonvolatile storage value is supported.

How to Configure SNMP and Use the IP Tunnel MIB

Configuring the Router to Use SNMP

Note Some of the tasks in this section include examples of the SNMP CLI syntax used to set configuration parameters on the router and to read values from MIB objects on the router. These SNMP CLI syntax examples are taken from a Linux workstation using public domain SNMP tools. The SNMP CLI syntax for your workstation might be different. See the documentation that was provided with your SNMP tools for the correct syntax for your network management workstation.

Before you can use the IP Tunnel MIB feature, you must first configure the router to support SNMP. Perform this task to enable SNMP on the router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server community** *string1* **ro**
4. **snmp-server community** *string2* **rw**
5. **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	**snmp-server community** *string1* **ro**	Sets up the community access string to permit access to SNMP.

	Command or Action	Purpose
	Example: `Router(config)# snmp-server community public ro`	• The *string1* argument is a community string that consists of from 1 to 32 alphanumeric characters and functions much like a password, permitting access to the SNMP protocol. Blank spaces are not permitted in the community string. • The **ro** keyword specifies read-only access. SNMP management stations using this string can retrieve MIB objects. **Note** The SNMP community read-only (RO) string for the examples is public. You should use a more complex string for this value in your configuration.
Step 4	**snmp-server community** *string2* **rw** **Example:** `Router(config)# snmp-server community private rw`	Sets up the community access string to permit access to SNMP. • The *string2* argument is a community string that consists of from 1 to 32 alphanumeric characters and functions much like a password, permitting access to the SNMP protocol. Blank spaces are not permitted in the community string. • The **rw** keyword specifies read-write access. SNMP management stations using this string can retrieve and modify MIB objects. **Note** The SNMP community read-write (RW) string for the examples is private. You should use a more complex string for this value in your configuration.
Step 5	**end** **Example:** `Router(config)# end`	Exits the current configuration mode and returns to privileged EXEC mode.

What to Do Next

To implement the IP Tunnel MIB, you must configure a tunnel. For information on configuring tunnels, see the " Implementing Tunnels " chapter in the Cisco IOS Interface and Hardware Component Configuration Guide.

To debug or troubleshoot any issues related to configuring the IP Tunnel MIB through SNMP, use the debug snmp tunnel-mib command. For information on this command see Cisco IOS Interface and Hardware Component Command Reference.

Additional References

Related Documents

Related Topic	Document Title
SNMP commands, complete command syntax, command reference, command history, defaults, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Configuring SNMP Support	*Cisco IOS Network Management Configuration Guide*
Implementing tunnels	*Cisco IOS Interface and Hardware Component Configuration Guide*

Standards

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

MIBs

MIB	MIBs Link
IP Tunnel MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 4087	IP Tunnel MIB

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

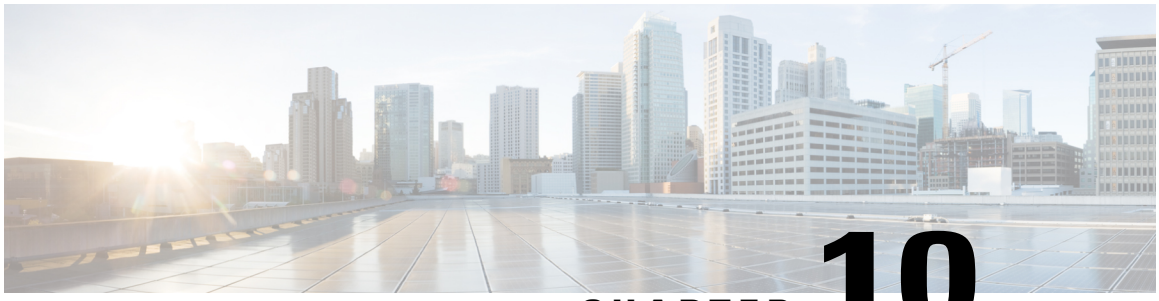
Feature Information for the Tunnel MIB

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 11: Feature Information for the IP Tunnel MIB

Feature Name	Releases	Feature Information
IP Tunnel MIB	12.2(33)SRB 12.2(1st)SY 12.2(44)SG 12.2(33)SRD 15.0(1)M Cisco IOS XE 3.1.0SG Cisco IOS XE Release 3.9S	The IP Tunnel MIB provides a generic MIB for managing all IPv4- and IPv6-related tunnels, as outlined in RFC 4087 IP Tunnel MIB.



CHAPTER 10

IF-MIBs

This module contains information about MIBs used with interfaces and hardware components. The IF-MIB supports all tables defined in RFC 2863, The Interfaces Group MIB, and the CISCO-IFEXTENSION-MIB. This MIB provides the ability to query the Interfaces MIB objects, and the information returned is restricted to the Virtual Private Network (VPN) routing/forwarding (VRF) instance to which the Simple Network Management Protocol (SNMP) context is mapped. Notification hosts may also be configured with contexts to restrict the notifications that need to be sent to the particular host.

The IF-MIB supports context-aware packet information in VRF environments. VRF environments require that contexts apply to VPNs so that clients can be given selective access to the information stored in the IF-MIB. Clients belonging to a particular VRF can access information about the interface from IF-MIB that belongs only to that VRF. When a client tries to get information from an interface that is associated with a particular context, the client can see the information that belongs to only that context and cannot see information to which it is not entitled.

This document describes the enhancement of the Interfaces Group MIB for subinterfaces and RFC 2233 compliance for Cisco's implementation of the IF-MIB in Cisco IOS software.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 Using the IF-MIB

To use the Interface Group MIB and Ethernet-like Interface MIB described in this document, you must configure SNMP on your system. It is assumed you will be using Cisco IOS or a network management system (NMS) such as CiscoWorks to monitor the performance of your network. For information on these topics, see the documents listed in the "Related Documents" section or the documentation that came with your network management application.

Information About the IF-MIB

The IF-MIB complies with RFC 2233 and provides SNMP support for subinterfaces. Additionally, you can configure SNMP to use either the existing Cisco implementation of linkUp or linkDown traps or the IF-MIB implementation consistent with IETF standards. Refer to RFC 2233 for information about linkUp and linkDown traps.

Starting with Cisco IOS Release 12.1(2)T/12.0(21)S3, you can configure your router to begin using the new RFC 2233 IETF standards-based implementation by using the **snmp-server trap link ietf** command. This command enables notification support for subinterfaces and is disabled by default to allow you to continue using the earlier Cisco implementation of linkUp/linkDown traps if you so choose.

However, please note that when using the earlier Cisco object definitions, an arbitrary value is used for the *locIfReason* object in linkUp/linkDown traps for subinterfaces, which may give you unintended results. This is because the *locIfReason* object is not defined for subinterfaces in the current Cisco implementation, which uses OLD-CISCO-INTERFACES-MIB.my.

If you do not enable this functionality, the link trap varbind list will consist of {ifIndex, ifDescr, ifType, locIfReason}. After you enable this functionality with the **snmp-server trap link ietf** command, the varbind list will consist of {inIndex, ifAdminStatus, ifOperStatus, if Descr, ifType}. The *locIfReason* object will also be conditionally included in this list depending on whether meaningful information can be retrieved for that object. A configured subinterface will generate retrievable information. On non-HWIDB interfaces, there will be no defined value for *locIfReason* , so it will be omitted from the trap message.

Other updates to the IF-MIB module have also been made to comply with RFC2233. These changes include the addition of the ifCounterDiscontinuityTime object, and the addition of basic support for ifTableLastChange. Updated Online Insertion and Removal (OIR) drivers are planned in a future release for full ifTableLastChange support.

Benefits of the IF-MIB

Compliance with RFC 2233

The enhancement to the IF-MIB allows Cisco IOS to support RFC 2233. Prior to this release, Cisco IOS supported only RFC 1573.

linkUp/linkDown Trap Generation for Subinterfaces

The enhancement to the IF-MIB allows linkUp and linkDown SNMP traps for subinterfaces to be generated correctly, while permitting unaffected users to continue using the earlier Cisco implementation.

The Context-Aware IF-MIB

The context-aware IF-MIB provides the ability to query the Interfaces MIB objects and the information returned be restricted to the VRF to which the SNMP context is mapped. Notification hosts may also be configured with contexts to restrict the notifications that need to be sent to the particular host.

In a VPN environment, different interfaces belong to different VRF instances. VRF instances can be uniquely associated with SNMP context. With the context-aware IF-MIB, when SNMP requests that include a specified context mapped to a VRF instance are received, only information related to those interfaces that belong to the VRF associated with the context is obtained.

Retrieve IP Helper Addresses

The IF-MIB enables you to retrieve all IP helper addresses configured on each interface.

How to Enable IETF-Compliant Link Traps for SNMP

Configuration of the IF-MIB is optional on your system and is disabled by default. To configure you need to enable IETF-Compliant Link Traps for SNMP. Perform this task to enable the use of the new object list for SNMP linkUp/linkDown traps, use the following commands, starting in privileged EXEC mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server trap link ietf**
4. **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.

	Command or Action	Purpose
Step 3	**snmp-server trap link ietf** **Example:** `Router(config)# snmp-server trap link ietf`	Enables SNMP traps that are compliant with RFC 2233.
Step 4	**end** **Example:** `Router(config)# end`	Ends the current configuration session and returns you to privileged EXEC mode.

What to Do Next

Verifying IETF-Compliant Link Traps for SNMP

Use the **more system:running-config** command in privileged EXEC mode to verify that the command is in your running configuration file.

Troubleshooting Tips

To monitor SNMP trap activity in real-time for the purposes of troubleshooting, use the SNMP debug commands, including the **debug snmp packet** command. For documentation of SNMP debug commands, see the Release 12.4 *Cisco IOS Debug Command Reference,* available on Cisco.com at http://www.cisco.com/en/US/docs/ios/debug/command/reference/db_book.html, or on the Cisco Documentation CD-ROM.

Example to Enable IETF-Compliant Link Traps for SNMP

The following example shows the SNMP related output before the IETF-compliant implementation is enabled, a configuration session in which it is enabled, and the changed output after the configuration:

```
Router#
more system:running config
. . .
snmp-server engineID local 00000009000000A1616C2056
snmp-server community public RO
snmp-server community private RW
. . .
Router#
conf term

Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
snmp-server trap link ietf

Router(config)#
end
```

```
Router#
 more system:running config
. . .
snmp-server engineID local 00000009000000A1616C2056
snmp-server community public RO
snmp-server community private RW
snmp-server trap link ietf
. . .
```

To enable/disable link traps for a particular interface:

```
7609_supBXL_45(config-if)#snmp trap link-status ?
  permit  Permit the following capability
  <cr>
7609_supBXL_45(config-if)#
```

To enable link up/down traps during switchover:

```
7609_supBXL_45(config)#snmp-server trap link ?
  ietf        Use IETF standard for SNMP traps
  switchover  Enable link up/down traps during switchover
```

How to Configure SNMP and Use the IF-MIB

Configuring the Router to Use SNMP

Before you query IF-MIB feature using SNMP, you must first configure the router to support SNMP.

Note Some of the tasks in this section include examples of the SNMP CLI syntax used to set configuration parameters on the router and to read values from MIB objects on the router. These SNMP CLI syntax examples are taken from a Linux workstation using public domain SNMP tools. The SNMP CLI syntax for your workstation might be different. See the documentation that was provided with your SNMP tools for the correct syntax for your network management workstation.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server community** *string1* **ro**
4. **snmp-server community** *string2* **rw**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server community** *string1* **ro** **Example:** `Router(config)# snmp-server community public ro`	Sets up the community access string to permit access to SNMP. • The *string1* argument is a community string that consists of from 1 to 32 alphanumeric characters and functions much like a password, permitting access to the SNMP protocol. Blank spaces are not permitted in the community string. • The **ro** keyword specifies read-only access. SNMP management stations using this string can retrieve MIB objects. **Note** The SNMP community read-only (RO) string for the examples is public. You should use a more complex string for this value in your configuration.
Step 4	**snmp-server community** *string2* **rw** **Example:** `Router(config)# snmp-server community private rw`	Sets up the community access string to permit access to SNMP. • The *string2* argument is a community string that consists of from 1 to 32 alphanumeric characters and functions much like a password, permitting access to the SNMP protocol. Blank spaces are not permitted in the community string. • The **rw** keyword specifies read-write access. SNMP management stations using this string can retrieve and modify MIB objects. **Note** The SNMP community read-write (RW) string for the examples is private. You should use a more complex string for this value in your configuration.
Step 5	**end** **Example:** `Router(config)# end`	Exits the current configuration mode and returns to privileged EXEC mode.

What to Do Next

To implement the IF-MIB, you must configure a tunnel. For information on configuring tunnels, see the "Implementing Tunnels" chapter in this guide.

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

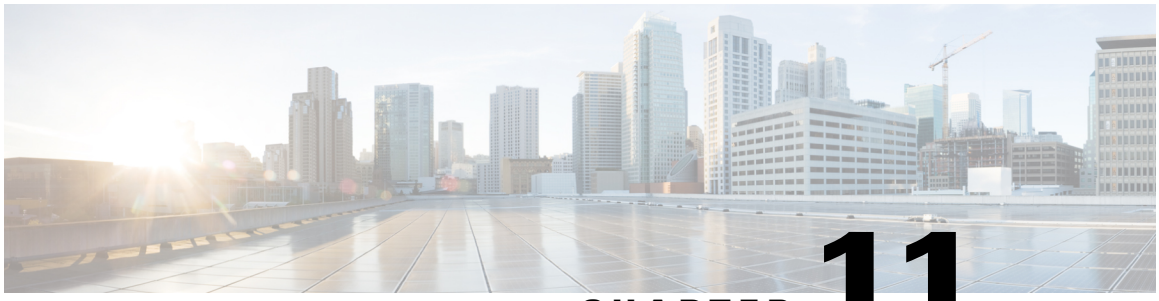
Feature Information for IF-MIBs

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 12: Feature Information for IF-MIB

Feature Name	Releases	Feature Information
IF-MIB	12.1(2)T 12.0(21)S3 12.3(2)T 12.0(24)S 12.2(2)SXI 12.2(33)SB Cisco IOS Release 3.9S	A router can be configured using the RFC 2233 IETF standards-based implementation. The IF-MIB enables notification support for subinterfaces. The LinkUp/Down traps are generated when a link goes up or down. This feature updates the LinkUp/Down trap information to include ifAdminStatus and ifOperStatus. The IF-MIB supports the IP Helper addresses and enable you to retrieve all IP helper addresses configured on each interface. You have the ability to query the Interfaces MIB objects and the information returned is restricted to the VRF to which the SNMP context is mapped to. Notification hosts may also be configured with contexts to restrict the notifications that need to be sent to a particular host.



CHAPTER 11

Synchronous Ethernet (SyncE) ESMC and SSM

This module describes Synchronization Status Message (SSM), Ethernet Synchronization Message Channel (ESMC), and generating the Simple Network Management Protocol (SNMP) traps on the SyncE feature.

With Ethernet equipment gradually replacing Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) equipment in service-provider networks, frequency synchronization is required to provide high-quality clock synchronization over Ethernet ports.

Synchronous Ethernet (SyncE) provides the required synchronization at the physical level. In SyncE, Ethernet links are synchronized by timing their bit clocks from high-quality, stratum-1-traceable clock signals in the same manner as SONET/SDH. Operation messages maintain SyncE links and ensure that a node always derives timing from the most reliable source.

SyncE synchronizes clock frequency over an Ethernet port. In SONET/SDH the communication channel for conveying clock information is SSM, and in SyncE it is the ESMC.

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 for Synchronous Ethernet (SyncE) ESMC and SSM, on page 117.

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

Prerequisites for Synchronous Ethernet (SyncE) ESMC and SSM

You need to first configure the network clock for SyncE configuration. Automatic synchronization of the network clock should be enabled. Ensure that the **network-clock-select** and **network-clock-participate** commands do not exist in the configuration in order to continue with the SyncE configuration.

Restrictions for Synchronous Ethernet (SyncE) ESMC and SSM

- To use the **network-clock synchronization ssm option** command, the following conditions are required:
  - No input source is in the configuration.
  - No network clock quality level is in the configuration.
  - No network clock source quality is set under any synchronous Ethernet interface.
- The **network-clock synchronization ssm option** command must be compatible with the **network-clock eec** command in the configuration.
- The **esmc process** and **synchronous mode** commands can be used only if the SyncE capable interface is installed on the router.

Information About Synchronous Ethernet (SyncE) ESMC and SSM

Synchronous Ethernet (SyncE) ESMC and SSM

Customers using a packet network find it difficult to provide timing to multiple remote network elements (NEs) through an external time division multiplexed (TDM) circuit. The SyncE feature helps to overcome this problem by providing effective timing to the remote NEs through a packet network. SyncE leverages the physical layer of the Ethernet to transmit frequency to the remote sites. SyncE's functionality and accuracy resemble the SONET/SDH network because of its physical layer characteristic. SyncE uses ESMC to allow the best clock source traceability to correctly define the timing source and help prevent a timing loop.

SONET/SDH use 4 bits from the two S bytes in the SONET/SDH overhead frame for message transmission. Ethernet relies on ESMC that is based on an IEEE 802.3 organization-specific slow protocol for message transmission. Each NE along the synchronization path supports SyncE, and SyncE effectively delivers frequency in the path. SyncE does not support relative time (for example, phase alignment) or absolute time (Time of Day).

SyncE provides the Ethernet physical layer network (ETY) level frequency distribution of known common precision frequency references. Clocks for use in SyncE are compatible with the clocks used in the SONET/SDH synchronization network. To achieve network synchronization, synchronization information is transmitted through the network via synchronous network connections with performance of egress clock. In SONET/SDH the communication channel for conveying clock information is Synchronization Status Message (SSM), and in SyncE it the Ethernet Synchronization Message Channel (ESMC).

ESMC carries a Quality Level (QL) identifier that identifies the timing quality of the synchronization trail. QL values in QL-TLV are the same as QL values defined for SONET and SDH SSM. Information provided by SSM QLs during the network transmission helps a node derive timing from the most reliable source and prevents timing loops. ESMC is used with the synchronization selection algorithms. Because Ethernet networks are not required to be synchronous on all links or in all locations, the ESMC channel provides this service. ESMC is composed of the standard Ethernet header for an organization-specific slow protocol; the ITU-T OUI, a specific ITU-T subtype; an ESMC-specific header; a flag field; and a type, length, value (TLV) structure. The use of flags and TLVs improves the management of SyncE links and the associated timing change. For details on Synchronous Ethernet support on Cisco 7600 series routers see Cisco 7600 Series Ethernet Services Plus (ES+) and Ethernet Services Plus T (ES+T) Line Card Configuration Guide .

How to Configure Synchronous Ethernet (SyncE) ESMC and SSM

Configuring SyncE

Perform this task to configure SyncE using ESMC and SSM.

SUMMARY STEPS

1. **enable**
2. **network-clock set l ockout** {**external** *slot* / *card* / *port*[**10m**| **2m**| **t1**{**sf** | **esf** | **d4**}] | **interface** *type slot* / *port*}
3. **network-clock clear lockout** {**external** *slot* / *card* / *port* [**10m**| **2m**| **t1** {**sf** | **esf** | **d4**}] | **interface** *type slot* / *port*}
4. **network-clock switch force { external** *slot* / *card* / *port* [**10m** | **2m**] | **t0** | **t1** {**sf** | **esf** | **d4**} **t0** | **internal { external** *slot* / *card* / *port*[**10m** | **2m**] | **t0}** | **interface** *type slot* / *port* **external** *slot* / *card* / *port* [**10m** | **2m**] | **t0** }
5. **network-clock switch manual { interface** *type slot* /port *{* **external** *slot* / *card* / *port* [**10m** | **2m**] | **t0** } | **external** *slot* / *card* / *port*{**10m** | **2m** | **t0** | **t1** {**sf** | **esf** | **d4**} | **internal { external** *slot* / *card* / *port*[**10m** | **2m**] | **t0}** *}*
6. **network-clock clear switch** {**t0** | **external** *slot* / *card* / *port* [**10m** | **2m**]}
7. **configure terminal**
8. **network-clock synchronization automatic**
9. **network-clock synchronization ssm option** {**1**| **2**{**GEN1**| **GEN2**}}
10. **network-clock input-source** *priority* {**external** *slot* / *card* / *port* [**10m** | **2m** | **t1** {**sf** | **esf** | **d4**}] | **interface** *type slot* / *port*}
11. **network-clock synchronization mode ql-enabled**
12. **network-clock hold-off** {**0**| *milliseconds*}
13. **network-clock wait-to-restore** *seconds*
14. **esmc process**
15. **network-clock external** *slot* / *card* / *port* **hold-off** {**0** | *milliseconds*}
16. **network-clock quality-level** {**tx**| **rx**} *value* {**interface** *type slot* / *port* | **external** *slot* / *card* / *port* [**10m** | **2m** | **t1** {**sf** | **esf** | **d4**}]
17. **network-clock output-source {line** | **system**} *priority interface type slot* / *port* **external** *slot* / *card* / *port*[**10m** | **2m** | **t1**{**sf** | **esf** | **d4**} **]**
18. **interface** *type number*
19. **synchronous mode**
20. **esmc mode** [**ql-disabled**| **tx**| **rx**] *value*
21. **network-clock source quality-level** *value* {**tx** | **rx**}
22. **network-clock hold-off** {**0** | *milliseconds*}
23. **network-clock wait-to-restore** *seconds*
24. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**network-clock set l ockout** {**external** *slot* / *card* / *port*[**10m**| **2m**| **t1**{**sf** | **esf** | **d4**}] | **interface** *type slot* / *port*} **Example:** `Router# network-clock set lockout GigabitEthernet7/1`	Sets the lockout state of input to "on." The input then is no longer considered available by the selection process.
Step 3	**network-clock clear lockout** {**external** *slot* / *card* / *port* [**10m**| **2m**| **t1** {**sf** | **esf** | **d4**}] | **interface** *type slot* / *port*} **Example:** `Router# network-clock clear lockout GigabitEthernet7/1`	Sets the lockout state of input to "off." The input then is considered available by the selection process.
Step 4	**network-clock switch force { external** *slot* / *card* / *port* [**10m** | **2m**] | **t0** | **t1** {**sf** | **esf** | **d4**} **t0** | **internal { external** *slot* / *card* / *port*[**10m** | **2m**] | **t0**} | **interface** *type slot* / *port* **externa**l *slot* / *card* / *port* [**10m** | **2m**] | **t0** } **Example:** `Router# network-clock switch force interface GigabitEthernet 7/1 t0`	Overrides the currently selected synchronization source when the synchronization source is enabled and not locked out. If the source selected by the forced switch command is disabled or locked out, the forced switch command is automatically rejected.
Step 5	**network-clock switch manual { interface** *type slot* /port *{* **externa**l *slot* / *card* / *port* [**10m** | **2m**] | **t0** } | **external** *slot* / *card* / *port*{**10m** | **2m** | **t0** | **t1** {**sf** | **esf** | **d4**} | **internal { external** *slot* / *card* / *port*[**10m** | **2m**] | **t0**} *}* **Example:** `Router# network-clock switch manual interface GigabitEthernet 7/1 t0`	Selects the synchronization source interface when it is enabled and not locked out. Manual switching is used to override the previously assigned synchronization source priorities.
Step 6	**network-clock clear switch** {**t0** | **external** *slot* / *card* / *port* [**10m** | **2m**]} **Example:** `Router# network-clock clear switch t0`	Clears the forced switch and manual switch commands. If the interface is not specified, the force/manual selected interface gets automatically cleared.
Step 7	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 8	**network-clock synchronization automatic** **Example:** `Router(config)# network-clock synchronization automatic`	Enables the network clock selection algorithm. This command disables the Cisco-specific network clock process and turns on the G.781-based automatic clock selection process.
Step 9	**network-clock synchronization ssm option** {**1**| **2**{**GEN1**| **GEN2**}} **Example:** `Router(config)# network-clock synchronization ssm option 2 GEN2`	Configures the router to work in a synchronization network. • Option 1 refers to synchronization networks designed for Europe. This is the default value. • Option 2 refers to synchronization networks designed for United States.
Step 10	**network-clock input-source** *priority* {**external** *slot* **/** *card* **/** *port* [**10m** | **2m** | **t1** {**sf** | **esf** | **d4**}] | **interface** *type slot* **/** *port*} **Example:** `Router(config)# network-clock input-source 1 interface GigabitEthernet 7/1`	Enables selecting an interface that is configured as clock source line, an external timing input interface, a GPS interface, or a packet-based timing recovered clock as the input clock for the system. Interface can be SyncE or channelized SONET.
Step 11	**network-clock synchronization mode ql-enabled** **Example:** `Router(config)# network-clock synchronization mode ql-enabled`	Configures the automatic selection process ql-enabled mode. • QL is disabled by default. • ql-enabled mode can be used only when the synchronization interface is capable to send SSM.
Step 12	**network-clock hold-off** {**0**| *milliseconds*} **Example:** `Router(config)# network-clock hold-off 0`	(Optional) Configures hold-off timer for the interface.
Step 13	**network-clock wait-to-restore** *seconds* **Example:** `Router(config)# network-clock wait-to-restore 70`	(Optional) Configures wait-to-restore timer for the SyncE interface.
Step 14	**esmc process** **Example:** `Router(config)# esmc process`	Enables the ESMC process.

	Command or Action	Purpose
Step 15	**network-clock external** *slot* / *card* / *port* **hold-off** {**0** | *milliseconds*} **Example:** `Router(config)# network-clock external 0/1/0 hold-off 0`	Overrides the hold-off timer value for the external interface.
Step 16	**network-clock quality-level** {**tx**| **rx**} *value* {**interface** *type slot* / *port* | **external** *slot* / *card* / *port* [**10m** | **2m** | **t1** {**sf** | **esf** | **d4**}] **Example:** `Router(config)# network-clock quality-level rx QL-STU GigabitEthernet 0/0/0`	Forces the QL value for line or external timing input and output.
Step 17	**network-clock output-source {line** | **system**} *priority interface type slot* / *port* **external** *slot* / *card* / *port*[**10m** | **2m** | **t1**{**sf** | **esf** | **d4**} **]** **Example:** `Router(config)# network-clock output-source line 1 GigabitEthernet1/2 external 0/0/1 10m`	Transmits the signal from the external timing input interface to the external timing output interface.
Step 18	**interface** *type number* **Example:** `Router(config)# interface GigabitEthernet 0/0`	Enters interface configuration mode.
Step 19	**synchronous mode** **Example:** `Router(config-if)# synchronous mode`	Configures the Ethernet interface to synchronous mode and automatically enables the ESMC and QL process on the interface.
Step 20	**esmc mode** [**ql-disabled**| **tx**| **rx**] *value* **Example:** `Router(config-if)# esmc mode rx QL-STU`	(Optional) Enables the ESMC process on the interface.
Step 21	**network-clock source quality-level** *value* {**tx** | **rx**} **Example:** `Router(config-if)# network-clock source quality-level QL-ST4 tx`	(Optional) Provides the forced QL value to the local clock selection process.

	Command or Action	Purpose
Step 22	**network-clock hold-off** {**0** | *milliseconds*} **Example:** `Router(config-if)# network-clock hold-off 0`	(Optional) Configures the hold-off timer for the interface.
Step 23	**network-clock wait-to-restore** *seconds* **Example:** **Example:** `Router(config-if)# network-clock wait-to-restore 70`	(Optional) Configures the wait-to-restore timer for the SyncE interface.
Step 24	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Enabling and Disabling an SNMP Trap in the SyncE Event

A Simple Network Management Protocol (SNMP) trap is defined for an SNMP agent to notify the Network Management Systems (NMS) about any unsolicited information. The SNMP trap notifies NMS when a critical SyncE event occurs on a device. If the SNMP trap is enabled in the SyncE configuration, the SNMP agent code generates a SyncE trap for the SyncE events.

Perform the following tasks to enable and disable the SNMP trap for the SyncE event:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps netsync**
4. **no snmp-server enable traps netsync**
5. **end**
6. **show running-config all | include traps**

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	**snmp-server enable traps netsync** **Example:** `Router(config)# snmp-server enable traps netsync`	Enables the SyncE traps.
Step 4	**no snmp-server enable traps netsync** **Example:** `Router(config)#` **no snmp-server enable traps netsync**	(Optional) Disables the SyncE traps.
Step 5	**end** **Example:** `Router(config)# end`	Exits global configuration mode.
Step 6	**show running-config all | include traps** **Example:** `Router# show running-config all | include trap`	(Optional) Displays the SyncE traps that are enabled on the router.

Configuration Examples for Synchronous Ethernet (SyncE) ESMC and SSM

Example Synchronous Ethernet (SyncE) ESMC and SSM

The following examples shows the SyncE configuration sequence (configuring an interface with two SyncE interfaces and two external interfaces):

```
Interface GigabitEthernet0/0/0
    synchronous mode
    clock source line
    network-clock wait-to-restore 720
!
Interface GigabitEthernet1/0/0
    synchronous mode
    clock source line
!
network-clock synchronization automatic
network-clock input-source 1 external 0/0/0 2m
network-clock input-source 2 external 1/0/0 2m
network-clock output-source line 1 interface GigabitEthernet0/0/0 external 0/0/0 2m
network-clock output-source line 1 interface GigabitEthernet1/0/0 external 1/0/0 2m
```

The following examples shows how to verify whether ESMC is enabled or not:

```
Router# show esmc

Interface: GigabitEthernet0/0/0
Administrative configurations:
  Mode: Synchronous
  ESMC TX: Enable
  ESMC RX : Enable
  QL RX configured : NA
  QL TX configured : NA
Operational status:
  Port status: UP
  QL Receive: QL-SSU-B
  ESMC Information rate : 1 packet/second
  ESMC Expiry: 5 second
```

The following examples shows how to view the network clock synchronization details:

```
Router# show network-clock synchronization detail

Automatic selection process : Enable
Equipment Clock : 2048 (EEC-Option1)
Clock Mode : QL-Enable
ESMC : Disabled
SSM Option : 1
T0 : Internal
Hold-off (global) : 300 ms
Wait-to-restore (global) : 300 sec
Revertive : No
Force Switch: FALSE
Manual Switch: FALSE
Number of synchronization sources: 1
Secondary src: Ethernet0/0
Slots disabled 0x0
Monitor source(s):  Ethernet0/0
Selected QL: QL-SEC
sm(netsync_ql_dis NETCLK_QL_ENABLE), running yes, state 1A
Last transition recorded: (begin)-> 1A (ql_mode_enable)-> 1A (src_added)-> 1A
```

```
Nominated Interfaces

 Interface            SigType     Mode/QL      Prio  QL_IN     ESMC Tx   ESMC Rx
*Internal             NA          NA/Dis       251   QL-SEC    NA        NA
 Et0/0                NA          Sync/En      2     QL-DNU    -         -

Interface:
-------------------------------------------
Local Interface: Internal
Signal Type: NA
Mode: NA(Ql-enabled)
SSM Tx: Disable
SSM Rx: Disable
Priority: 251
QL Receive: QL-SEC
QL Receive Configured: -
QL Receive Overrided: -
QL Transmit: -
QL Transmit Configured: -
Hold-off: 0
Wait-to-restore: 0
Lock Out: FALSE
Signal Fail: FALSE
Alarms: FALSE
Slot Disabled: FALSE

Local Interface: Et0/0
Signal Type: NA
Mode: Synchronous(Ql-enabled)
ESMC Tx: Enable
ESMC Rx: Enable
Priority: 2
QL Receive: QL-DNU
QL Receive Configured: -
QL Receive Overrided: -
QL Transmit: -
QL Transmit Configured: -
Hold-off: 300
Wait-to-restore: 300
Lock Out: FALSE
Signal Fail: FALSE
Alarms: FALSE
Slot Disabled: FALSE
Dont Use: FALSE
Configured Priority: 2
Force Switch: FALSE
Manual Switch: FALSE
Manual Switch In progress: FALSE
Holdoff_cfg: FALSE
Wtr_cfg: FALSE
Reason for alarm flag: 0
Msw in progress: FALSE
Intf_sig_nv: 0
Hold off Timer: Stopped
Wait to restore Timer: Stopped
Switchover Timer: Stopped
ESMC Tx Timer: Stopped
ESMC Rx Timer: Stopped
Tsm Delay Timer: Stopped
```

Example Enabling and Disabling an SNMP Trap in the SyncE Event

The following example shows how to enable and disable an SNMP trap in the SyncE event:

```
Router > enable
```

```
Router # configure terminal
Router(config)# snmp-server enable traps netsync
Router (config)# no snmp-server enable traps netsync
Router (config)# end
Router# show running-config all| include traps
snmp-server enable traps flowmon
snmp-server enable traps sonet
snmp-server enable traps netsync
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Interface and hardware component configuration commands	*Cisco IOS Interface and Hardware Component Command Reference*
Cisco 7600 Synchronous Ethernet	Cisco 7600 Series Ethernet Services Plus (ES+) and Ethernet Services Plus T (ES+T) Line Card Configuration Guide

Standards

Standard	Title
ITU-T G.8262	*Timing characteristics of synchronous ethernet equipment slave clock (EEC)*
ITU-T G.8264	*Timing distribution through Packet Networks*
ITU-T G.781	*Synchronization layer functions*

MIBs

MIB	MIBs Link
CISCO-NETSYNC-MIB	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
None	--

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 Synchronous Ethernet (SyncE) ESMC and SSM

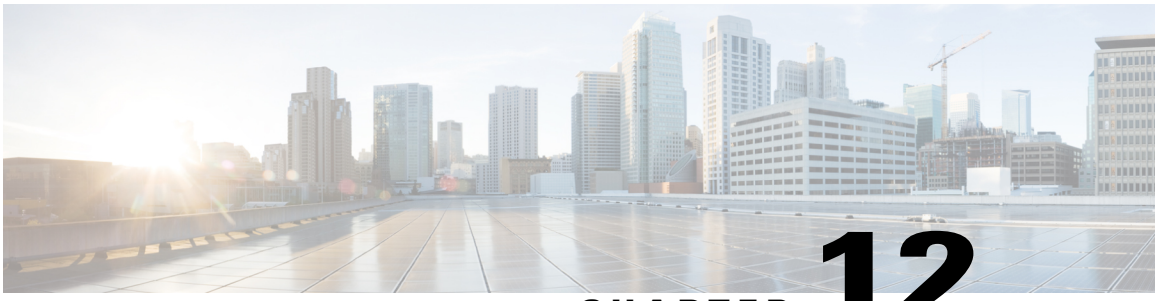
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train 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 13: Feature Information for Synchronous Ethernet (SyncE): ESMC and SSM

Feature Name	Releases	Feature Information
Generating SNMP Trap in SyncE Feature	15.1(2)S Cisco IOS XE Release 3.8S	This feature describes how to set SNMP traps in SyncE to notifies the NMS about any unsolicited information. The following commands were introduced or modified by this feature: **no snmp-server enable traps netsync, show running-config all| include trap, snmp-server enable traps netsync.**

Feature Name	Releases	Feature Information
Synchronous Ethernet (SyncE): ESMC and SSM	15.0(1)S Cisco IOS XE Release 3.8S	This feature supports ESMC and the SSM control protocol for SyncE to synchronize clock frequency over an Ethernet port with quality level selection. The following commands were introduced or modified by this feature: **esmc mode ql-disabled**, **esmc process**, **show esmc**, **show interfaces accounting**.



CHAPTER 12

1+1 SR-APS Without Bridging

The Automatic Protection Switching (APS) feature provides link redundancy and allows switchover of Packet over SONET (POS) circuits in the event of circuit failure and is often required when you connect Synchronous Optical Networking (SONET) equipment to telecommunications equipment. In the single router (SR) APS feature both protect and working interfaces must be on same router.

APS is a mechanism of using a protect POS interface in the SONET network as the backup for a working POS interface. When the working interface fails, the protect interface quickly assumes its traffic load. Based on the configuration, the two circuits can be terminated in the same router. The protection mechanism has a 1+1 architecture with bidirectional connection. Bridging refers to the transmission of user data to both working interface and protect interface. In nonbridging scenario the user data is sent to working interface only.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 1+1 SR-APS Without Bridging

Configure the working interface first, along with the IP address of the interface. This configuration helps to prevent the protect interface from becoming the active circuit during APS configuration. If the protect interface becomes active in case if it has been configured first by mistake, you can use **shut** or **no shut** command to make the working interface active.

Restrictions for 1+1 SR-APS Without Bridging

- Both the protect and working interfaces should be configured identically. No warning message will be displayed if the configurations are different between the interfaces.
- Behavior of the APS pair (protect and working interfaces) will be indeterministic if the configurations of protect and working interfaces are not identical.
- APS switch over within 50 milliseconds is not supported during online insertion and removal (OIR) or during crash of the shared port adapter (SPA) or carrier card (CC).
- APS switching simultaneously with Route Processor (RP) or forwarding plane (FP) high availability (HA) need not be within 50 milliseconds.

Information About 1+1 SR-APS Without Bridging

1+1 SR-APS Without Bridging

The APS feature provides link redundancy and allows switchover of POS circuits in the event of circuit failure and is often required when you connect SONET equipment to telecommunications equipment. In the SR-APS feature both protect and working interfaces must be on same router.

APS is a mechanism of using a protect POS interface in the SONET network as the backup for a working POS interface. When the working interface fails, the protect interface quickly assumes its traffic load. Based on the configuration, the two circuits can be terminated in the same router. The protection mechanism has a 1+1 architecture with bidirectional connection.

In the 1+1 architecture, there is one working interface (circuit) and one protect interface, and the same payload from the transmitting end is sent to both the receiving ends. The receiving end decides the interface that needs to be used. The line overhead (LOH) bytes (K1 and K2) in the SONET frame indicate both status and action. When one interface is down or the K1/K2 bytes have changed, APS brings up the protect interface using regular interface configuration messages.

Bridging refers to the transmission of user data to both the working interface and the protect interface. In nonbridging scenario the user data is sent to the working interface only. You must set the working interface to be the active interface. Cisco ASR 1000 Series Routers (ASR1000) supports only the nonbridging scenario.

In the nonbridging scenario the ASR1000 (with the APS enabled) transmits a signal to the remote end. The ASR1k transmits the signal (except K1/K2 bytes) only to the working interface and not to the protect interface. The K1/K2 bytes are transmitted only to the protect interface. However, ASR1000 can be connected to devices that support bridging APS, which means the devices transmit the same signal to both working and protect

interfaces of ASR1000. But the ASR1000 will send the user data (except K1/K2 bytes) only to the working interface of that device. The K1/K2 bytes are transmitted to the protect interface.

SR-APS uses Protect Group Protocol (PGP) between working and protect interfaces. The protect interface APS configuration should include an IP address of a loopback interface on the same router to communicate with the working interface using PGP. Using the PGP, POS interfaces can be switched in case of a degradation or loss of channel signal, or manual intervention. In bidirectional mode, the receive and transmit channels are switched as a pair.

In bidirectional APS the local and the remote connections negotiate the ingress interface to be selected for the data path. The egress interface traffic is not transmitted to both working and protect interfaces.

How to Configure 1+1 SR-APS Without Bridging

Configuring APS Working and Protect Interfaces

Perform this task to configure APS working and protect interfaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface pos** *slot/sub-slot/port*
4. **aps working** *circuit-number*
5. **aps protect** *circuit-number ip-address*
6. **end**
7. **show controllers pos**
8. **show interfaces pos**
9. **show aps**

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.

	Command or Action	Purpose
Step 3	**interface pos** *slot/sub-slot/port* **Example:** `Router(config)# interface pos 2/0/0`	Specifies the POS interface to be configured as the working interface and enters interface configuration mode.
Step 4	**aps working** *circuit-number* **Example:** `Router(config-if)# aps working 1`	Configures the interface as a working interface.
Step 5	**aps protect** *circuit-number ip-address* **Example:** `Router(config-if)# aps protect 1 209.165.200.224`	Configures the interface as a protect interface. Specifies the IP address of loopback interface on the same router that contains the working interface.
Step 6	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 7	**show controllers pos** **Example:** `Router(config)# show controllers pos`	Displays information about the POS controllers so that you can verify that the interface is configured correctly.
Step 8	**show interfaces pos** **Example:** `Router(config)# show interfaces pos`	Displays information about the configured interfaces.
Step 9	**show aps** **Example:** `Router(config)# show aps`	Displays information about APS on the configured router.

Configuring Other APS Options

Perform this task to configure other APS options.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface pos** *slot/sub-slot/port*
4. **aps force** *circuit-number*
5. **aps group** *group-number*
6. **aps lockout** *circuit-number*
7. **aps manual** *circuit-number*
8. **aps revert** *minutes*
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 pos** *slot/sub-slot/port* **Example:** `Router(config)# interface pos 2/0/0`	Specifies the POS interface to be configured as the working interface and enters interface configuration mode.
Step 4	**aps force** *circuit-number* **Example:** `Router(config-if)# aps force 1`	(Optional) Manually switches the specified circuit to a protect interface, unless a request of equal or higher priority is in effect.
Step 5	**aps group** *group-number* **Example:** `Router(config-if)# aps group 20`	(Optional) Allows more than one protect or working interface group to be supported on a router.
Step 6	**aps lockout** *circuit-number* **Example:** `Router(config-if)# aps lockout 1`	(Optional) Prevents a working interface from switching to a protect interface.

	Command or Action	Purpose
Step 7	**aps manual** *circuit-number* **Example:** `Router(config-if)# aps manual 1`	(Optional) Manually switches a circuit to a protect interface, unless a request of equal or higher priority is in effect.
Step 8	**aps revert** *minutes* **Example:** `Router(config-if)# aps revert 3`	(Optional) Enables automatic switchover from the protect interface to the working interface after the working interface becomes available.
Step 9	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Monitoring and Maintaining APS

Perform this task to monitor and maintain APS.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **show controllers pos**
4. **show interfaces pos**
5. **show aps**

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.

	Command or Action	Purpose
Step 3	**show controllers pos** **Example:** `Router(config)# show controllers pos`	Displays information about the POS controllers so that you can verify that the interface is configured correctly.
Step 4	**show interfaces pos** **Example:** `Router(config)# show interfaces pos`	Displays information about the configured interfaces.
Step 5	**show aps** **Example:** `Router(config)# show aps`	Displays information about APS on the configured router.

Configuring SONET Alarm Reporting

To configure the thresholds and the type of SONET alarms that are reported, use any of the following commands. The commands listed in this section are optional. To display the current Bit Error Rate (BER) threshold setting or to view the reporting of the SONET alarms, use the **show controllers pos** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface pos** *slot/sub-slot/port*
4. **pos threshold** {**b1-tca** | **b2-tca** | **b3-tca** | **sd-ber** | **sf-ber**} *rate*
5. **pos report** {**b1-tca** | **b2-tca** | **b3-tca** | **lais** | **lrdi** | **pais** | **plop** | **prdi** | **rdool** | **sd-ber** | **sf-ber** | **slof** | **slos**}
6. **end**

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 pos** *slot/sub-slot/port* **Example:** `Router(config)# interface pos 2/0/0`	Specifies the POS interface to be configured as the working interface and enters interface configuration mode.
Step 4	**pos threshold** {**b1-tca** | **b2-tca** | **b3-tca** | **sd-ber** | **sf-ber**} *rate* **Example:** `Router(config-if)# pos threshold b1-tca 4`	(Optional) Configures the BER threshold values for signal failure (SF), signal degrade (SD), or threshold crossing alarms (TCAs).
Step 5	**pos report** {**b1-tca** | **b2-tca** | **b3-tca** | **lais** | **lrdi** | **pais** | **plop** | **prdi** | **rdool** | **sd-ber** | **sf-ber** | **slof** | **slos**} **Example:** `Router(config-if)# pos report b2-tca`	(Optional) Enables reporting of selected SONET alarms.
Step 6	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring LAIS as an APS Switchover Trigger

When you place the working interface into administrative shutdown state, the switchover happens with or without **pos ais-shut**. When **pos ais-shut** is enabled on the interface, the interface sends the line alarm indication signal (LAIS) alarm to the remote end of the administrative shutdown, and the LAIS alarm makes the switchover bit faster. The **carrier-delay msec** *milliseconds* command and **ppp timeout retry seconds** [*milliseconds*] command are also used to make the APS switchover happen faster.

The **carrier-delay msec** *milliseconds* command delays the link down event processing for POS interfaces. For example, if the carrier delay is set to 50 milliseconds (ms), the router will ignore all link down events that are cleared within 50 msec. If the link goes down there will be no APS switchover for 50 ms. The default carrier delay is 2 seconds and there will be no APS switchover for 2 seconds after the link goes down. Hence the carrier delay is set to 50 ms for faster switchover.

The **ppp timeout retry seconds** [*milliseconds*] command sets the PPP retry timeout to the specified time. For example, if the timeout retry is set to 200 ms, the router tries to establish PPP link in 200 ms after it detects

the signal outage due to APS switchover. If the default retry timeout of 2 seconds is used, then the PPP link will be established 2 seconds after the APS switchover. Hence the PPP timeout retry is set to 50 ms for faster switchover.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface pos** *slot/sub-slot/port*
4. **pos ais-shut**
5. **carrier-delay msec** *milliseconds*
6. **ppp timeout retry seconds** [*milliseconds*]
7. **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 pos** *slot/sub-slot/port* **Example:** `Router(config)# interface pos 2/0/0`	Specifies the POS interface to be configured as the working interface and enters interface configuration mode.
Step 4	**pos ais-shut** **Example:** `Router(config-if)# pos ais-shut`	Sends line alarm indication signal (LAIS) alarm on Admin shut of the interface.
Step 5	**carrier-delay msec** *milliseconds* **Example:** `Router(config-if)# carrier-delay msec 50`	Delays the link down event processing for POS interfaces and makes the APS switchover faster.

	Command or Action	Purpose
Step 6	**ppp timeout retry seconds** [*milliseconds*] **Example:** `Router(config-if)# ppp timeout retry 0 200`	Sets the maximum waiting period for a response during PPP negotiation and makes the APS switchover faster.
Step 7	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for 1+1 SR-APS Without Bridging

Example Configuring 1+1 SR-APS Without Bridging

The following example shows the configuration sequence for 1+1 SR-APS:

```
interface loopback 1
ip address 1.1.1.1 255.255.255.0
interface pos 2/0/0
     aps group 1
     aps working 1
     pos ais-shut
     end
interface pos 3/0/0
     aps group 1
     aps protect 1 1.1.1.1
     pos ais-shut
     end
```

The following example shows the sample output of APS configured on a router with a working interface:

```
Router# show aps
POS2/1/1 APS Group 0: protect channel 0 (Inactive)
        Working channel 1 at 10.0.1.1 (Enabled)
        bidirectional, revertive (60 seconds)
        PGP timers (default): hello time=1; hold time=3
                hello fail revert time=120
        SONET framing; SONET APS signalling by default
        Received K1K2: 0x00 0x05
                No Request (Null)
        Transmitted K1K2: 0x00 0x05
                No Request (Null)
        Remote APS configuration: (null)
POS2/1/0 APS Group 0: working channel 1 (Active)
        Protect at 10.0.1.1
        PGP timers (from protect): hello time=1; hold time=3
        SONET framing
        Remote APS configuration: (null)
```

The following example shows the display of POS controllers:

```
Router# show controller pos 2/1/0
```

```
POS2/1/0
SECTION
  LOF = 0          LOS    = 1                            BIP(B1) = 0
LINE
  AIS = 2          RDI    = 2          FEBE = 14         BIP(B2) = 0
PATH
  AIS = 2          RDI    = 2          FEBE = 4          BIP(B3) = 6
  PLM = 0          UNEQ   = 0          TIM  = 0          TIU     = 0
  LOP = 1          NEWPTR = 2          PSE  = 0          NSE     = 0
Active Defects: None
Active Alarms:  None
Alarm reporting enabled for: SF SLOS SLOF B1-TCA B2-TCA PLOP B3-TCA
Framing: SONET
APS
 working (active)
  COAPS = 13          PSBF = 0
  State: PSBF_state = False
  Rx(K1/K2): 00/00  Tx(K1/K2): 00/00
  Rx Synchronization Status S1 = 00
  S1S0 = 00, C2 = CF
  Remote aps status (none); Reflected local aps status (none)
CLOCK RECOVERY
  RDOOL = 0
  State: RDOOL_state = False
PATH TRACE BUFFER: STABLE
  Remote hostname : SPA-APS2
  Remote interface: POS2/2/0
  Remote IP addr  : 10.1.1.1
  Remote Rx(K1/K2): 00/00  Tx(K1/K2): 00/00
BER thresholds:  SF = 10e-3  SD = 10e-6
TCA thresholds:  B1 = 10e-6  B2 = 10e-6  B3 = 10e-6
Clock source:  internal
```

The following example shows the configuration information and statistics for a POS interface:

```
Router# show interface pos 2/1/0
POS2/1/0 is up, line protocol is up  (APS working - active)
  Hardware is SPA-4XOC12-POS
  Internet address is 10.1.1.2/24
  MTU 4470 bytes, BW 155000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Scramble disabled
  Last input 00:00:02, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/375/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  30 second input rate 0 bits/sec, 0 packets/sec
  30 second output rate 0 bits/sec, 0 packets/sec
     102477 packets input, 2459448 bytes, 0 no buffer
     Received 0 broadcasts (0 IP multicasts)
     0 runts, 4 giants, 0 throttles 0 parity
     4 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     102486 packets output, 2459934 bytes, 0 underruns
     0 output errors, 0 applique, 2 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers swapped out
     10 carrier transitions
```

Additional References

Related Documents

Related Topic	Document Title
APS commands	*Cisco IOS Interface and Hardware Component Command Reference*

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	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
None	

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

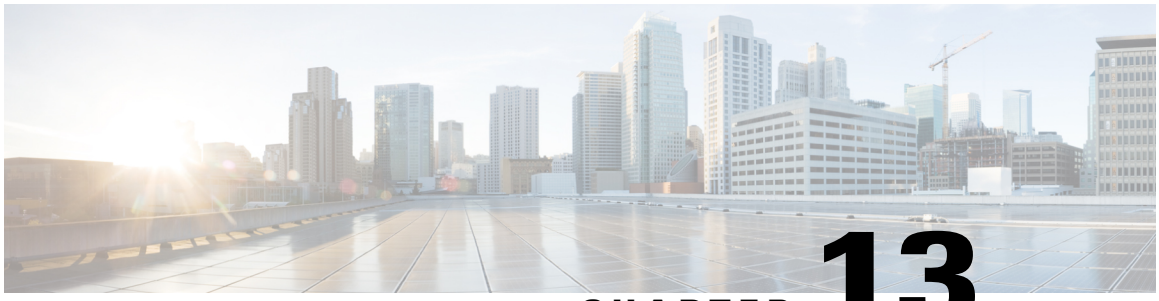
Feature Information for 1+1 SR-APS Without Bridging

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 14: Feature Information for 1+1 SR-APS Without Bridging

Feature Name	Releases	Feature Information
1+1 SR-APS Without Bridging	Cisco IOS XE Release 3.1S	This feature provides support to 1+1 single router APS without bridging. There were no commands introduced or modified by this feature.



CHAPTER 13

IPv6 Rapid Deployment

The IPv6 rapid deployment feature allows a service provider to provide a unicast IPv6 service to customers over its IPv4 network by using encapsulation of IPv6 in IPv4.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About IPv6 Rapid Deployment

IPv6 Rapid Deployment Tunnels

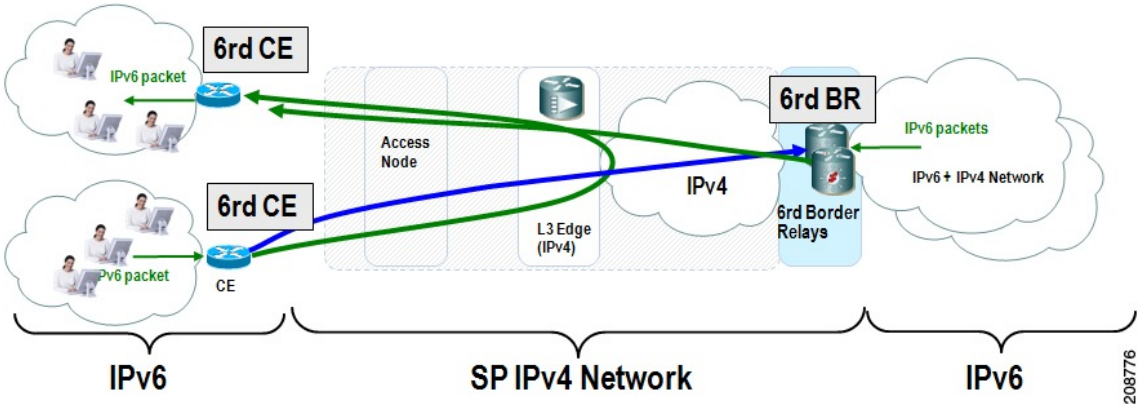
The 6RD feature is an extension of the 6to4 feature. The 6RD feature allows a service provider (SP) to provide a unicast IPv6 service to customers over its IPv4 network by using encapsulation of IPv6 in IPv4.

The main differences between 6RD and 6to4 tunneling are as follows:

- 6RD does not require addresses to have a 2002::/16 prefix; therefore, the prefix can be from the SP's own address block. This function allows the 6RD operational domain to be within the SP network. From the perspective of customer sites and the general IPv6 internet connected to a 6RD-enabled SP network, the IPv6 service provided is equivalent to native IPv6.
- All 32 bits of the IPv4 destination need not be carried in the IPv6 payload header. The IPv4 destination is obtained from a combination of bits in the payload header and information on the router. Furthermore, the IPv4 address is not at a fixed location in the IPv6 header as it is in 6to4.

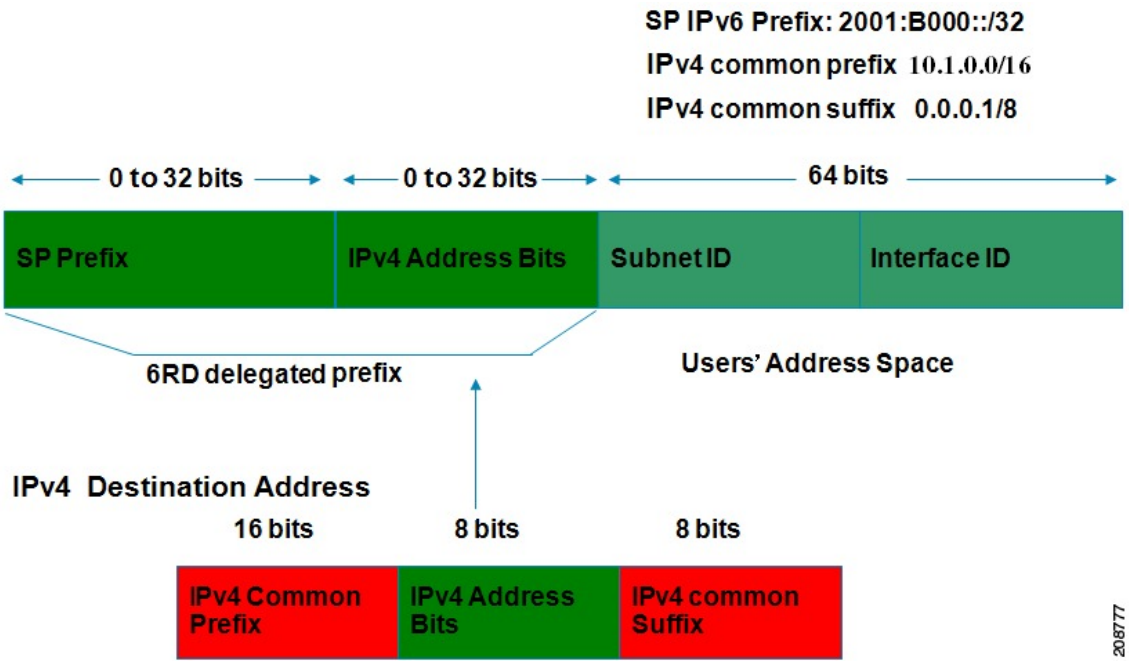
The 6RD SP prefix was selected by the SP for the IPv6 deployment shown in the figure below. The 6RD delegated prefix is derived from the SP prefix and the IPv4 address bits, and is used by the CE for hosts within its site.

Figure 4: 6RD Deployment



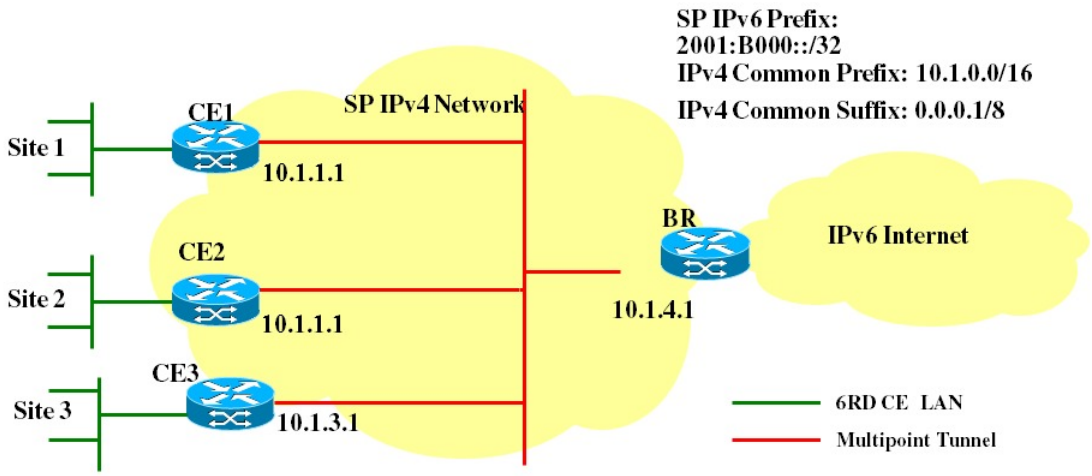
The figure below shows how 6RD prefix delegation works.

Figure 5: 6RD Prefix Delegation Explanation



The figure below shows a 6RD prefix delegation topology.

Figure 6: 6RD Prefix Delegation and Explanation



SP Prefix	2001:B000::/32
IPv4 Common Prefix	10.1.0.0/16
IPv4 Common Suffix	0.0.0.1/8
CE1: Delegated 6RD prefix	2001:B000:0100::/40
CE2: Delegated 6RD prefix	2001:B000:0200::/40
BR: Delegated 6RD prefix	2001:B000:0400::/40
CE1 (IPv4) tunnel transport source	10.1.1.1
CE2 (IPv4) tunnel transport source	10.1.2.1
BR (IPv4) tunnel transport source	10.1.4.1

208778

How to Configure IPv6 Rapid Deployment

Configuring 6RD Tunnels

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **tunnel source** {*ip-address*| *interface-t ype interface-number*}
5. **tunnel mode ipv6ip** [**6rd** | **6to4** | **auto-tunnel** | **isatap**]
6. **tunnel 6rd prefix** *ipv6-prefix* / *prefix-length*
7. **tunnel 6rd ipv4** {**prefix-length** *length*} {**suffix-length** *length*}

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 1`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	**tunnel source** {*ip-address*| *interface-t ype interface-number*} **Example:** `Router(config-if)# tunnel source loopback 1`	Specifies the source interface type and number for the tunnel interface.
Step 5	**tunnel mode ipv6ip** [**6rd**	**6to4**	**auto-tunnel**	**isatap**]	Configures a static IPv6 tunnel interface.

	Command or Action	Purpose
	Example: `Router(config-if)# tunnel mode ipv6ip 6rd`	• The **auto-tunnel** keyword is not supported on Cisco ASR 1000 series routers.
Step 6	**tunnel 6rd prefix** *ipv6-prefix* / *prefix-length* **Example:** `Router(config-if)# tunnel 6rd prefix 2001:B000::/32`	Specifies the common IPv6 prefix on IPv6 rapid 6RD tunnels.
Step 7	**tunnel 6rd ipv4** {**prefix-length** *length*} {**suffix-length** *length*} **Example:** `Router(config-if)# tunnel 6rd ipv4 prefix-length 16 suffix 8`	Specifies the prefix length and suffix length of the IPv4 transport address common to all the 6RD routers in a domain.

Configuration Examples for IPv6 Rapid Deployment

Example: Configuring 6RD Tunnels

The following example shows the running configuration of a 6RD tunnel and the corresponding output of the **show tunnel 6rd** command:

```
interface Tunnel1
 ipv6 address 2001:B000:100::1/32
 tunnel source loopback 1
 tunnel mode ipv6ip 6rd
 tunnel 6rd prefix 2001:B000::/32
 tunnel 6rd ipv4 prefix-len 16 suffix-len 8
end
Router# show tunnel 6rd tunnel 1
Interface Tunnel1:
  Tunnel Source: 10.1.1.1
  6RD: Operational, V6 Prefix: 2001:B000::/32
  V4 Common Prefix Length: 16, Value: 10.1.0.0
  V4 Common Suffix Length: 8, Value: 0.0.0.1
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

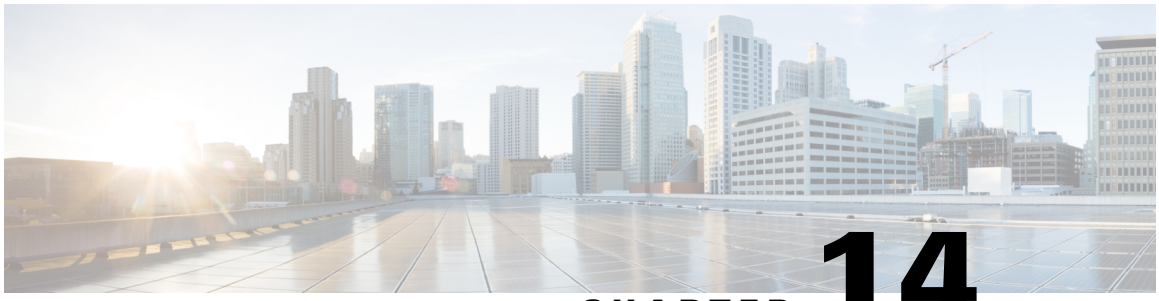
Feature Information for IPv6 Rapid Deployment

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 15: Feature Information for IPv6 Rapid Deployment

Feature Name	Releases	Feature Information
IP Tunneling: 6RD IPv6 Rapid Deployment	Cisco IOS XE Release 3.1S	The 6RD feature allows a service provider to provide a unicast IPv6 service to customers over its IPv4 network by using encapsulation of IPv6 in IPv4. The following commands were introduced or modified: **tunnel 6rd ipv4**, **tunnel 6rd prefix**, **tunnel mode ipv6ip**, **tunnel source**.



CHAPTER 14

IPv6 Automatic 6to4 Tunnels

This feature provides support for IPv6 automatic 6to4 tunnels. An automatic 6to4 tunnel allows isolated IPv6 domains to be connected over an IPv4 network to remote IPv6 networks.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About IPv6 Automatic 6to4 Tunnels

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) link. 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 to 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. Following the embedded IPv4 address are 16 bits that can be used to number networks within the site. The border router at each end of a 6to4 tunnel must support both the 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 be the global 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. As with other tunnel mechanisms, appropriate entries in a Domain Name System (DNS) that map between hostnames and IP addresses for both IPv4 and IPv6 allow the applications to choose the required address.

How to Configure IPv6 Automatic 6to4 Tunnels

Configuring Automatic 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 those tunnel types on the same router, we strongly recommend that they do not share the same tunnel source.

The reason that a 6to4 tunnel and an IPv4-compatible tunnel cannot share an interface is that 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. So when a packet with an IPv4 protocol type of 41 arrives on an interface, that packet is mapped to an IPv6 tunnel interface based on the IPv4 address. However, if both the 6to4 tunnel and the IPv4-compatible tunnel share the same source interface, the router is not able to determine the IPv6 tunnel interface to which it should assign the incoming packet.

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

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. **ipv6 address** {*ipv6-address* / *prefix-length* | *prefix-name sub-bits*/*prefix-length*
5. **tunnel source** {*ip-address*| *interface-t ype interface-number*}
6. **tunnel mode ipv6ip** [**6rd** | **6to4** | **auto-tunnel** | **isatap**
7. **exit**
8. **ipv6 route** [**vrf** *vrf-name*] *ipv6-prefix* / *prefix-length*{*ipv6-address* | *interface-type interface-number* [*ipv6-address*]} [**nexthop-vrf** [*vrf-name1* | **default**]] [*administrative-distance*] [*administrative-multicast-distance* | **unicast** | **multicast**] [*next-hop-address*] [**tag** *tag*]

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 1`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	**ipv6 address** {*ipv6-address* / *prefix-length*	*prefix-name sub-bits*/*prefix-length* **Example:** `Router(config-if)# ipv6 address 3ffe:b00:c18:1::3/127`	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface.
Step 5	**tunnel source** {*ip-address*| *interface-t ype interface-number*} **Example:** `Router(config-if)# tunnel source loopback 1`	Specifies the source interface type and number for the tunnel interface.
Step 6	**tunnel mode ipv6ip** [**6rd**	**6to4**	**auto-tunnel**	**isatap**	Configures a static IPv6 tunnel interface.

	Command or Action	Purpose
	Example: `Router(config-if)# tunnel mode ipv6ip 6rd`	• The **auto-tunnel** keyword is not supported on Cisco ASR 1000 series routers.
Step 7	**exit** **Example:** `Router(config-if) exit`	Exits interface configuration mode, and enters global configuration mode.
Step 8	**ipv6 route** [**vrf** *vrf-name*] *ipv6-prefix* / *prefix-length* {*ipv6-address* | *interface-type interface-number* [*ipv6-address*]} [**nexthop-vrf** [*vrf-name1* | **default**]] [*administrative-distance*] [*administrative-multicast-distance* | **unicast** | **multicast**] [*next-hop-address*] [**tag** *tag*] **Example:** `Router(config)# ipv6 route 2002::/16 tunnel 0`	Configures a static route for the IPv6 6to4 prefix 2002::/16 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.

Configuration Examples for IPv6 Automatic 6to4 Tunnels

Example: Configuring 6to4 Tunnels

The following example configures 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 GigabitEthernet0/0/0
 description IPv4 uplink
 ip address 192.168.99.1 255.255.255.0
!
interface GigabitEthernet1/0/0
 description IPv6 local network 1
 ipv6 address 2002:c0a8:6301:1::1/64
!
interface GigabitEthernet2/0/0
 description IPv6 local network 2
 ipv6 address 2002:c0a8:6301:2::1/64
!
interface Tunnel0
 description IPv6 uplink
 no ip address
 ipv6 address 2002:c0a8:6301::1/64
 tunnel source GigabitEthernet0/0/0
```

```
 tunnel mode ipv6ip 6to4
!
ipv6 route 2002::/16 tunnel 0
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

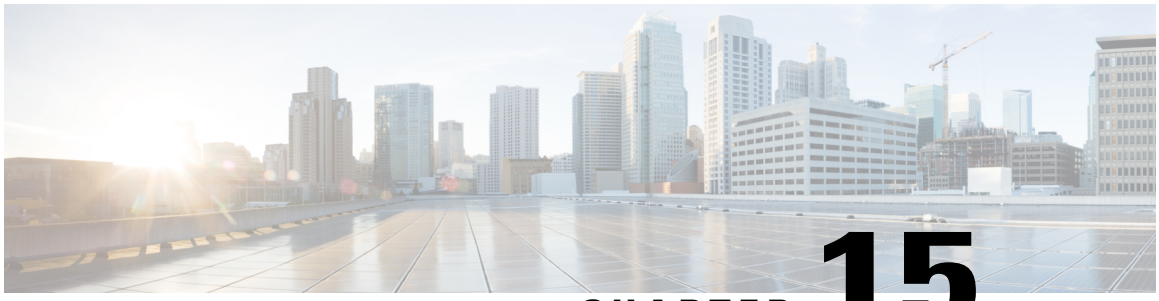
Feature Information for IPv6 Automatic 6to4 Tunnels

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 16: Feature Information for IPv6 Automatic 6to4 Tunnels

Feature Name	Releases	Feature Information
IPv6 Tunneling: Automatic 6to4 Tunnels	Cisco IOS XE Release 2.1	An automatic 6to4 tunnel allows isolated IPv6 domains to be connected over an IPv4 network to remote IPv6 networks. The following commands were introduced or modified: **tunnel mode ipv6ip**, **tunnel source**.



CHAPTER 15

IPv6 over IPv4 GRE 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.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

Information About IPv6 over IPv4 GRE Tunnels

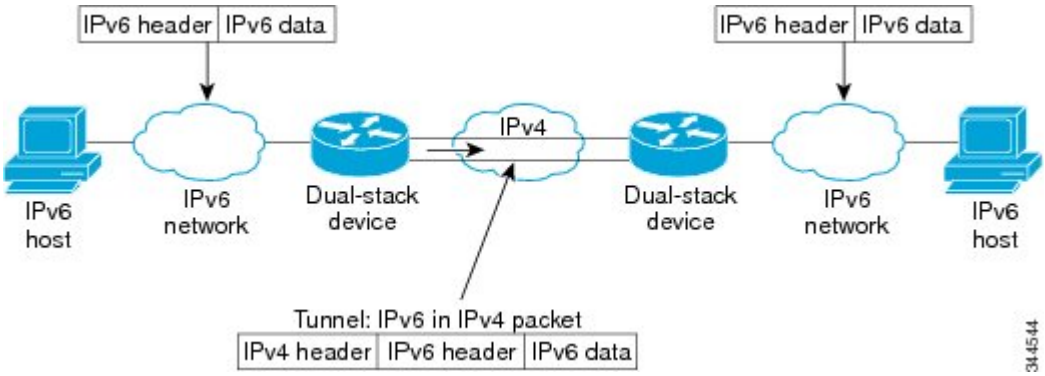
Overlay Tunnels for IPv6

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 7: 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 17: 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.

Tunneling Type	Suggested Usage	Usage Notes
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.
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 18: Tunnel Configuration Parameters by Tunneling Type

Tunneling Type	Tunnel Configuration Parameter		
Tunnel Mode	**Tunnel Source**	**Tunnel Destination**	**Interface Prefix or Address**

Tunneling Type	Tunnel Configuration Parameter			
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.

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.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel** *tunnel-number*
4. Enter one of the following commands:
   - **ipv6 address** {*ipv6-address*/*prefix-length* | *prefix-name sub-bits*/*prefix-length*}
   - **ipv6 address** *ipv6-prefix*/*prefix-length* [**eui-64**]
5. **tunnel source** {*ip-address* | *ipv6-address* | *interface-type interface-number*}
6. **tunnel destination** {*hostname* | *ip-address* | *ipv6-address*}
7. **tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ipv6** | **ipip** [**decapsulate-any**] | **iptalk** | **ipv6** | **mpls** | **nos**}
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 0`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	Enter one of the following commands: • **ipv6 address** {*ipv6-address*/*prefix-length*	*prefix-name sub-bits*/*prefix-length*} • **ipv6 address** *ipv6-prefix*/*prefix-length* [**eui-64**]	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface. • If you specify the **eui-64** keyword, the software configures an IPv6 address for an interface and enables

	Command or Action	Purpose
	Example: `Device(config-if)# ipv6 address 3ffe:b00:c18:1::3/127`	IPv6 processing on the interface using an EUI-64 interface ID in the low-order 64 bits of the address.
Step 5	**tunnel source** {*ip-address* | *ipv6-address* | *interface-type interface-number*} **Example:** `Device(config-if)# tunnel source gigabitethernet 0/0/0`	Specifies the source IPv4 address, IPv6 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** {*hostname* | *ip-address* | *ipv6-address*} **Example:** `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**} **Example:** `Device(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 8	**end** **Example:** `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 GigabitEthernet 0/0/0
 tunnel destination 2001:DB8:1111:2222::1/64
 tunnel mode gre ipv6
!
interface GigabitEthernet0/0/0
 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 GigabitEthernet 0/0/0
 tunnel destination 2001:DB8:1111:2222::2/64
 tunnel mode gre ipv6
!
interface GigabitEthernet0/0/0
 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 GigabitEthernet 0/0/0
Router(config
-if)
#
tunnel destination 10.0.0.2
Router(config
-if)
#
tunnel mode gre ipv6
Router(config
-if)
#
exit
!
Router(config
)
```

```
#
interface GigabitEthernet0/0/0
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
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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

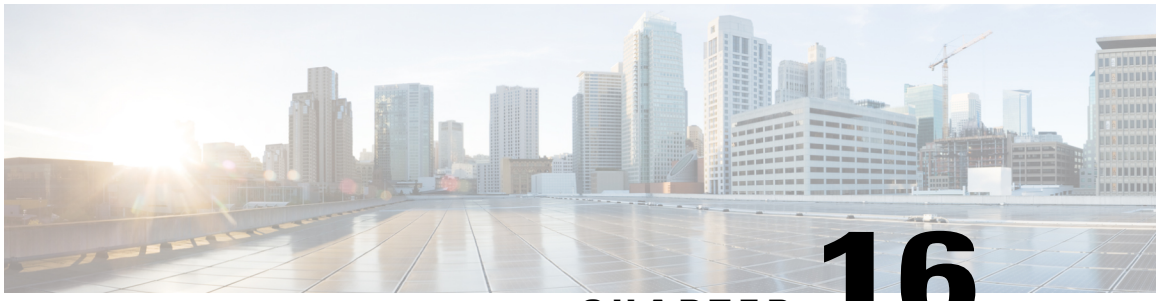
Feature Information for IPv6 over IPv4 GRE Tunnels

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 19: Feature Information for IPv6 over IPv4 GRE Tunnels

Feature Name	Releases	Feature Information
IPv6 over IPv4 GRE Tunnels	Cisco IOS XE Release 2.1	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 following commands were introduced or modified: **tunnel destination**, **tunnel mode ipv6ip**, **tunnel source**.



CHAPTER 16

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. Generic routing encapsulation (GRE) is a unicast protocol that offers the advantages of encapsulating broadcast and multicast traffic (multicast streaming or routing protocols) or other non-IP protocols and of being protected by IPsec.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 GRE IPv6 Tunnels

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

- There is limited support for tunnel transport in virtual routing and forwarding (VRF). The limited support in VRF is applicable to IPv6 point-to-point GRE without tunnel protection.

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.

GRE IPv6 Tunnel Protection

GRE IPv6 tunnel protection allows devices to work as security gateways, establish IPsec tunnels between other security gateway devices, and provide crypto IPsec protection for traffic from internal networks when the traffic is sent across the public IPv6 Internet. The GRE IPv6 tunnel protection functionality is similar to the security gateway model that uses GRE IPv4 tunnel protection.

How to Configure GRE IPv6 Tunnels

Configuring 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 (this is not shown in the task below). The host or device at each end of the configured tunnel must support both 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. **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 0`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 4	**tunnel source** {*ipv6-address* | *interface-type interface-number*} **Example:** `Device(config-if)# tunnel source ethernet 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, the interface must be configured with an IPv6 address. **Note** Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.
Step 5	**tunnel destination** *ipv6-address* **Example:** `Device(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 IPv6 Command Reference.
Step 6	**tunnel mode gre ipv6** **Example:** `Device(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 interface. Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.

	Command or Action	Purpose
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring GRE IPv6 Tunnel Protection

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. **tunnel protection ipsec profile** *profile-name*
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 0`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 4	**tunnel source** {*ipv6-address* | *interface-type interface-number*}	Specifies the source IPv6 address or the source interface type and number for the tunnel interface.

	Command or Action	Purpose
	Example: `Device(config-if)# tunnel source ethernet 0`	• If an interface type and number are specified, the interface must be configured with an IPv6 address. **Note** Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.
Step 5	**tunnel destination** *ipv6-address* **Example:** `Device(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 IPv6 Command Reference.
Step 6	**tunnel mode gre ipv6** **Example:** `Device(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 interface. Only the syntax used in this context is displayed. For more details, see the IPv6 Command Reference.
Step 7	**tunnel protection ipsec profile** *profile-name* **Example:** `Device(config-if)# tunnel protection ipsec profile ipsec-profile`	Associates the tunnel interface with an IPsec profile. **Note** For the *profile-name* argument, specify the IPsec profile configured in global configuration mode.
Step 8	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for GRE IPv6 Tunnels

Example: Configuring 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
```

Example: Configuring GRE IPv6 Tunnel Protection

The following example shows how to associate the IPsec profile “ipsec-profile” with a GRE IPv6 tunnel interface. The IPsec profile is configured using the **crypto ipsec profile** command.

```
crypto ipsec profile ipsec-profile
 set transform-set ipsec-profile
!
interface Tunnel1
 ip address 192.168.1.1 255.255.255.252
 tunnel source FastEthernet2/0
 tunnel destination 10.13.7.67
 tunnel protection ipsec profile ipsec-profile
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Master Commands List, All Releases
Tunnel commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	Interface and Hardware Component Command Reference
IPv6 commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	IPv6 Command Reference

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

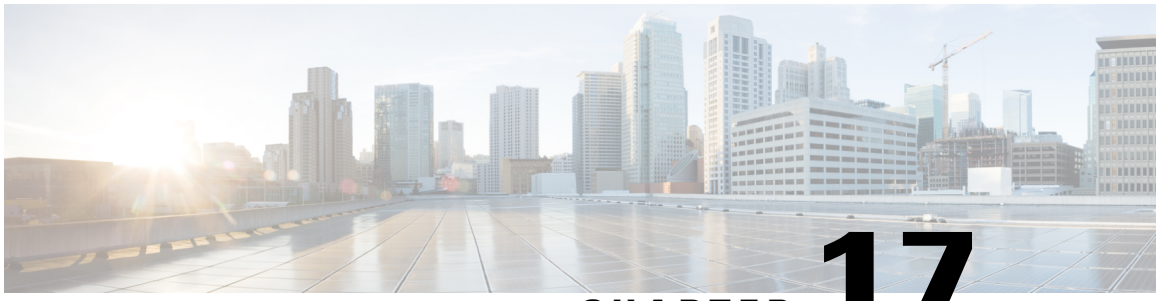
Feature Information for GRE IPv6 Tunnels

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 20: Feature Information for GRE IPv6 Tunnels

Feature Name	Releases	Feature Information
GRE IPv6 Tunnels	Cisco IOS XE Release 3.7S	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.



CHAPTER 17

ISATAP Tunnel Support for IPv6

ISATAP is an automatic overlay tunneling mechanism that uses the underlying IPv4 network as a NBMA link layer for IPv6

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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.

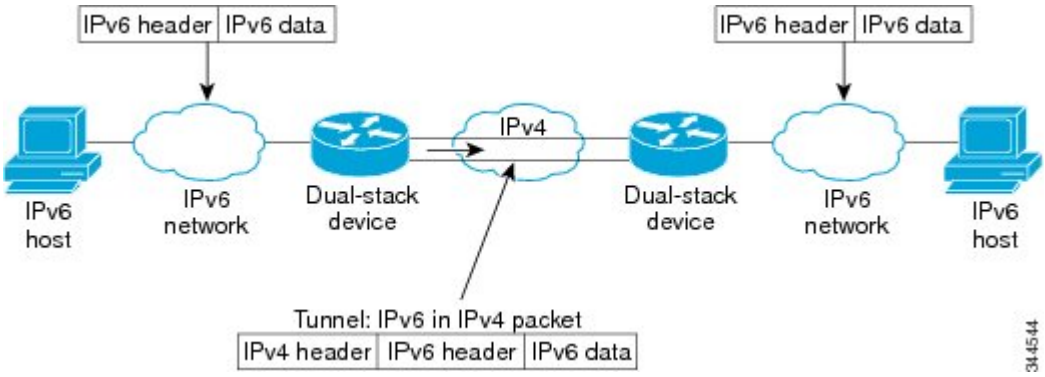
Information About ISATAP Tunnel Support for IPv6

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 8: 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 21: 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.

Tunneling Type	Suggested Usage	Usage Notes
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.
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 22: Tunnel Configuration Parameters by Tunneling Type

Tunneling Type	Tunnel Configuration Parameter		
Tunnel Mode	**Tunnel Source**	**Tunnel Destination**	**Interface Prefix or Address**

Tunneling Type	Tunnel Configuration Parameter			
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.

ISATAP Tunnels

ISATAP is an automatic overlay tunneling mechanism that uses the underlying IPv4 network as a 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 or 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. This feature allows clients to automatically configure themselves as they would do if they were connected to a GigabitEthernet or FastEthernet. It can also be configured to provide connectivity out of the site. ISATAP uses a well-defined IPv6 address format composed of any unicast IPv6 prefix (/64), which 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 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 describes an ISATAP address format.

Table 23: IPv6 ISATAP Address Format

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 what an actual ISATAP address would look like if the prefix is 2001:DB8:1234:5678::/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:DB8:1234:5678:0000:5EFE:0AAD:8108

How to Configure ISATAP Tunnel Support for IPv6

Configuring ISATAP Tunnels

Before You Begin

The **tunnel source** command used in the configuration of an ISATAP tunnel must point to an interface with an IPv4 address configured. The ISATAP IPv6 address and prefix (or prefixes) advertised are configured as 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-address* / *prefix-length* | *prefix-name sub-bits*/*prefix-length*
5. **no ipv6 nd ra suppress**
6. **tunnel source** {*ip-address*| *interface-type interface-number*}
7. **tunnel mode ipv6ip** [**6rd** | **6to4** | **auto-tunnel** | **isatap**

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 tunnel** *tunnel-number* **Example:** `Router(config)# interface tunnel 1`	Specifies a tunnel interface and number, and enters interface configuration mode.
Step 4	**ipv6 address** {*ipv6-address* / *prefix-length* | *prefix-name sub-bits*/*prefix-length* **Example:** `Router(config-if)# ipv6 address 2001:DB8:6301::/64 eui-64`	Specifies the IPv6 address assigned to the interface and enables IPv6 processing on the interface.
Step 5	**no ipv6 nd ra suppress** **Example:** `Router(config-if)# no ipv6 nd ra suppress`	Sending of IPv6 router advertisements is disabled by default on tunnel interfaces. This command reenables the sending of IPv6 router advertisements to allow client autoconfiguration.
Step 6	**tunnel source** {*ip-address*| *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source gigabitethernet 1/0/1`	Specifies 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 [6rd | 6to4 | auto-tunnel | isatap** **Example:** `Router(config-if)# tunnel mode ipv6ip isatap`	Specifies an IPv6 overlay tunnel using a ISATAP address. • The **auto-tunnel** keyword is not supported on Cisco ASR 1000 series routers.

Configuration Examples for ISATAP Tunnel Support for IPv6

Example: Configuring ISATAP Tunnels

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

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

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*Cisco IOS IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

Standards and RFCs

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

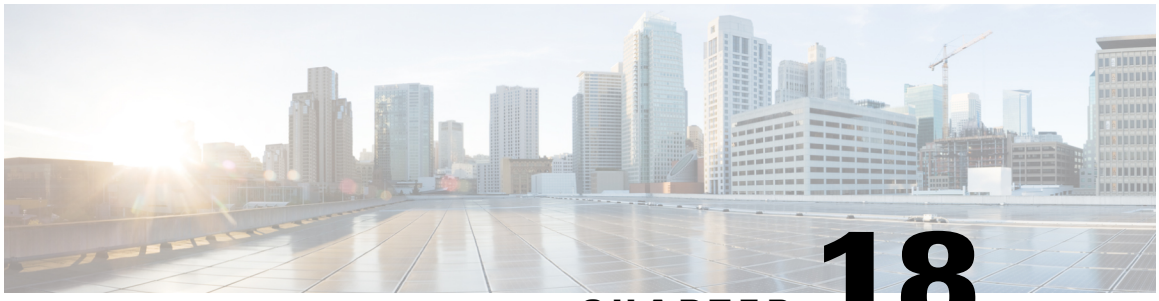
Feature Information for ISATAP Tunnel Support for IPv6

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 24: Feature Information for ISATAP Tunnel Support for IPv6

Feature Name	Releases	Feature Information
ISATAP Tunnel Support for IPv6	Cisco IOS XE Release 2.1	ISATAP is an automatic overlay tunneling mechanism that uses the underlying IPv4 network as a NBMA link layer for IPv6. The following commands were introduced or modified: **ipv6 nd ra suppress**, **tunnel mode ipv6ip**, **tunnel source**.



CHAPTER 18

VRF-Aware Tunnels

Virtual Routing and Forwarding (VRF)-aware tunnels are used to connect customer networks separated by untrusted core networks or core networks with different infrastructures (IPv4 or IPv6).

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 module.

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 VRF-Aware Tunnels

- You must configure customer edge networks. See the Configuring Customer Edge Networks for Tunneling section.
- You must configure the customer and transport VRFs. See the Defining a VRF Instance section.

Information About VRF-Aware Tunnels

Tunnel IP Source and Destination VRF Membership

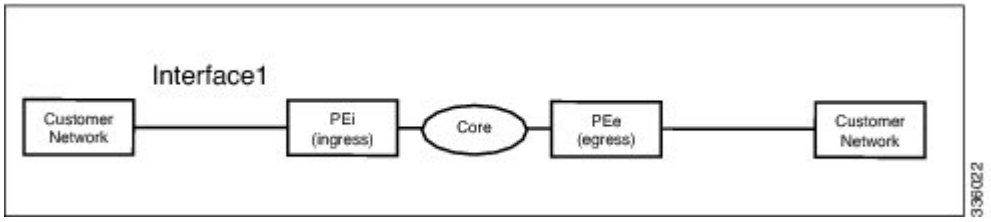
You can configure the source and destination of a tunnel to belong to any VPN routing and forwarding (VRFs) tables. A VRF table stores routing data for each VPN. The VRF table defines the VPN membership of a customer site that is attached to the network access server (NAS). Each VRF table comprises an 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.

You can configure the tunnel source and destination to belong to any VRF or to a global table. The tunnel becomes disabled if no route to the tunnel destination is defined.

VRF-Aware Tunnels

Virtual Routing and Forwarding (VRF)-aware tunnels are used to connect customer networks that are separated by untrusted IPv4 or IPv6 core networks.

Figure 9: VRF-Aware Tunnels



In the above topology, a tunnel is configured in the core network. Provider edge (PE) device PEi, is the tunnel head for packets entering on Interface 1. PE device PEe, is the tunnel tail for packets entering on Interface 1.

The VRF configured on Interface 1 is the customer VRF. Packets entering through Interface 1 are routed using this VRF. Packets exiting the tunnel are forwarded to this VRF. The routing by the customer VRF is called inner IP packet routing.

The VRF configured on the tunnel using the **tunnel vrf** command is the transport VRF. The transport VRF is the VRF that applies to the encapsulated payload and is used to look up the tunnel endpoints. This VRF is the same as the VRF associated with the physical interface over which the tunnel sends packets. The routing by the transport VRF is the outer IP packet routing.

The tunnel endpoint can be configured as an address from the global routing table or an address from a configured transport VRF table.

VRF-Aware IPv6 over IPv6 Tunnels

You can forward IPv6 packets on an untrusted IPv6 infrastructure by creating Virtual Routing and Forwarding (VRF)-aware IPv6 tunnels in it. These tunnels can have endpoints in a VRF table or in a global routing table. The tunnel modes used are **tunnel mode gre ipv6** and **tunnel mode ipv6**.

VRF-Aware IPv4 over IPv6 Tunnels

You can forward IPv4 packets on an untrusted IPv6 infrastructure by creating Virtual Routing and Forwarding (VRF)-aware IPv4 tunnels in it. These tunnels can have endpoints in a VRF table or in a global routing table. The tunnel modes used are **tunnel mode gre ipv6** and **tunnel mode ipv6**.

VRF-Aware IPv6 over IPv4 Tunnels

You can forward IPv6 packets on an untrusted IPv4 infrastructure by creating Virtual Routing and Forwarding (VRF)-aware IPv6 tunnels in it. These tunnels can have endpoints in a VRF table or in a global routing table. The tunnel modes used are **tunnel mode gre ipv4** (default mode) and **tunnel mode ipv4**.

How to Configure VRF-Aware IPv6 Tunnels

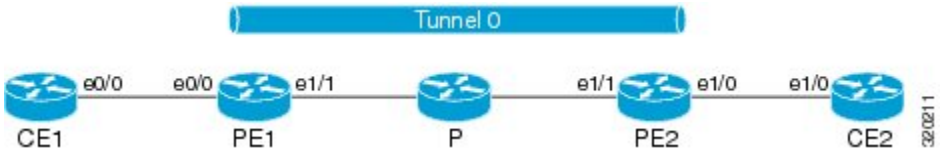
To configure a VRF-aware tunnel, you need to perform the following steps:

1. **Define the customer and transport VRF—**Define a customer VRF if the tunnel is VRF-aware. Define a transport VRF if the tunnel endpoint needs to be in a VRF. See the Defining a VRF Instance section.
2. **Set up the network—**Configure relevant interfaces and configure relevant routes. Ensure that a valid route exists between the PE devices and the PE device and the customer network.
3. **Configure the tunnel between the PE devices—**See the Configuring a VRF-Aware Tunnel section.
    1. **Configure the tunnel address**
    2. **Configure the tunnel source—**This is an interface on the PE device.
    3. **Configure the tunnel destination—**This is tunnel source of the other PE device. For proper configuration of the tunnel, ensure that the tunnel destination is reachable from the PE device with a ping command (A valid route must exist to the tunnel destination).
    4. **Configure the tunnel mode**
4. **Configure customer edge network**. See the Configuring Customer Edge Networks for Tunneling section.
5. **Configure static routes using the tunnel—**Configure routes on the PE devices to remote CE networks using the configured tunnel.

Configuring a VRF-Aware Tunnel

This task configures a tunnel between PE1 and PE2, as shown in the image below. The configuration task need to be repeated on both PE devices, PE1 and PE2.

Figure 10: Configuring a VRF-Aware Tunnel



SUMMARY STEPS

1. **interface** *type number*
2. **vrf forwarding** *transport-vrf-name*
3. • **ip address** *ip-address mask* or
   • **ipv6 address** *ipv6-address*/*prefix-length*
4. **exit**
5. Configure static routes between provider edge devices.
6. **interface tunnel** *number*
7. **vrf forwarding** *customer-vrf-name*
8. • **ip address** *ip-address mask* or
   • **ipv6 address** *ipv6-address*/*prefix-length*
9. **tunnel source** *interface-type interface-number*
10. **tunnel destination** [*ip-address* | *ipv6-address*]
11. **tunnel vrf** *transport-vrf-name*
12. **tunnel mode** {**aurp | cayman | dvmrp | eon | gre | gre multipoint | gre ipv6 | ipip [decapsulate-any] | ipsec ipv4 | iptalk | ipv6 | ipsec ipv6 | mpls | nos | rbscp**}
13. **exit**
14. • **ip route** [**vrf** *vrf-name*] *prefix mask interface-type interface-number* [*next-hop-ip-address*] or
    • **ipv6 route** [**vrf** *vrf-name*] *destination-ipv6-prefix interface-type interface-number* [*next-hop-ipv6-address*]
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/1`	Configures the interface used as a tunnel source.
Step 2	**vrf forwarding** *transport-vrf-name* **Example:** `Device(config-if)# vrf forwarding red`	(Optional) Associates the transport VRF with the tunnel. **Note** This step is not required if the tunnel endpoints are in the global routing table.
Step 3	• **ip address** *ip-address mask* or • **ipv6 address** *ipv6-address*/*prefix-length*	Sets an IP address for the tunnel source interface. • The address configured in this step for PE1 is used as the tunnel endpoint or tunnel destination while configuring the tunnel on PE2 and vice versa.

	Command or Action	Purpose

	Example:
	`Device(config-if)# ip address 10.22.22.22 255.255.255.255`
	or
	`Device(config-if)# ipv6 address 2001:DB8:3::1/64`
		• This address may be in the global routing table or in the VRF.

Step 4	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 5	Configure static routes between provider edge devices.	Provider edge devices are reachable with a **ping** or **ping vrf** command.
Step 6	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Configures the tunnel interface and enters interface configuration mode. The same tunnel needs to be configured on PE2.
Step 7	**vrf forwarding** *customer-vrf-name* **Example:** `Device(config-if)# vrf forwarding green`	(Optional) Associates the customer VRF instance with the tunnel. • Packets exiting the tunnel are forwarded to this VRF (inner IP packet routing). **Note** This step is required only for VRF-aware tunnels.

Step 8	• **ip address** *ip-address mask* or
	• **ipv6 address** *ipv6-address*/*prefix-length*
	Example:
	`Device(config-if)# ip address 10.4.1.1 255.255.255.0`
	or
	`Device(config-if)# ipv6 address 2001:DB8:3::1/64`
		Configures an IPv4 or IPv6 address for the tunnel.
		• This address is used as the next-hop address while configuring static routes. Ensure that PE1 and PE2 have addresses within the same network.

Step 9	**tunnel source** *interface-type interface-number* **Example:** `Device(config-if)# tunnel source ethernet 1/1`	Sets the source address for a tunnel interface.
Step 10	**tunnel destination** [*ip-address* | *ipv6-address*] **Example:** `Device(config-if)# tunnel destination 10.44.44.44`	(Optional) Specifies the destination for a tunnel interface. • The tunnel source address of device PE2 is used as the tunnel destination address of PE1 and vice versa. • If an IPv6 infrastructure exists between the two PE devices, use an IPv6 address. If an IPv4 infrastructure exists

	Command or Action	Purpose
		between the two PE devices, use an IPv4 address (IPv6 over IPv4 tunnel).
Step 11	**tunnel vrf** *transport-vrf-name* **Example:** `Device(config-if)# tunnel vrf red`	(Optional) Associates the transport VRF with the tunnel. • This VRF is the same as the VRF associated with the physical interface over which the tunnel sends packets (outer IP packet routing). **Note** This step is not required if the tunnel endpoints are in the global routing table.
Step 12	**tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ipv6** | **ipip [decapsulate-any]** | **ipsec ipv4** | **iptalk** | **ipv6** | **ipsec ipv6** | **mpls** | **nos** | **rbscp**} **Example:** `Device(config-if)# tunnel mode ipv6`	(Optional) Sets the encapsulation mode for the tunnel interface. **Note** This step is not required if the tunnel mode is GRE IPv4 as this is the default mode.
Step 13	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.

Step 14	• **ip route** [**vrf** *vrf-name*] *prefix mask interface-type interface-number* [*next-hop-ip-address*] or
	• **ipv6 route** [**vrf** *vrf-name*] *destination-ipv6-prefix interface-type interface-number* [*next-hop-ipv6-address*]
	
	Example:
	`Device(config)# ip route 10.44.44.0 255.255.255.0 10.22.22.23`
	`Device(config)# ip route vrf red 10.44.44.0 255.255.255.0 10.22.22.23`
	or
	`Device(config)# ipv6 route 2001:DB8:2:2::/64 2001:DB8:2:1::2`
	`Device(config)# ipv6 route vrf green 2001:DB8:2:2::/64 2001:DB8:2:1::2`
		Establishes static routes to remote customer networks by using the configured tunnel.
		• Use the tunnel address as the next hop.
		• For PE1, configure a static route to network PE2-CE2. For PE2, configure a static route to network PE1-CE1.

Step 15	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

What to Do Next

Verify the IPv6 Tunnels. See Verifying VRF-Aware Tunnels

Defining a VRF Instance

Perform this task to make a device Virtual Routing and Forwarding (VRF)-aware and to configure VRF-aware tunnels.

SUMMARY STEPS

1. **vrf definition** *vrf-name*
2. **rd** *route-distinguisher*
3. **route-target export** *route-target-ext-community*
4. **route-target import** *route-target-ext-community*
5. **address-family {ipv4 | ipv6}**
6. **exit-address-family**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition green`	Enters IP VRF configuration mode for defining a VRF routing table instance.
Step 2	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 1:1`	Specifies a route distinguisher (RD) for a VRF instance.
Step 3	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf)# route-target export 1:1`	Exports routing information to the target VPN extended community.
Step 4	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf)# route-target import 1:1`	Imports routing information to the target VPN extended community.
Step 5	**address-family {ipv4 | ipv6}** **Example:** `Device(config-vrf)# address-family ipv6`	Enters VRF address-family configuration mode to configure a routing session using standard IPv4 or IPv6 address prefixes.

	Command or Action	Purpose
Step 6	**exit-address-family** **Example:** `Device(config-vrf-af)# exit-address-family`	Exits VRF address-family configuration mode and enters IP VRF configuration mode.
Step 7	**exit** **Example:** `Device(config-vrf)# exit`	Exits IP VRF configuration mode and enters global configuration mode.

Configuring Customer Edge Networks for Tunneling

Perform this task to configure a customer edge (CE) network. In this configuration, the CE network is a network with CE devices connected to a provider edge (PE) device. PE1 and CE1 are connected and PE2 and CE2 are connected. Addresses must be configured accordingly.

Before You Begin

To define a customer VRF, see the Defining a VRF Instance section.

SUMMARY STEPS

1. **interface** *type number*
2. **vrf forwarding** *customer-vrf-name*
3. • **ip address** *ip-address mask* or
   • **ipv6 address** *ipv6-address/prefix-length*
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**interface** *type number* **Example:** `Device(config)# interface Ethernet 0/0`	Configures an interface type and enters interface configuration mode.
Step 2	**vrf forwarding** *customer-vrf-name* **Example:** `Device(config-if)# vrf forwarding green`	(Optional) Associates a VRF instance or a virtual network with the tunnel. **Note** This step is required only if the interface needs to be associated with a VRF.

	Command or Action	Purpose

Step 3	• **ip address** *ip-address mask* or
	• **ipv6 address** *ipv6-address/prefix-length*
	
	Example:
	`Device(config-if)# ip address 10.22.22.22 255.255.255.0`
	or
	`Device(config-if)# ipv6 address 2001:DB8:1::1/64`
		Configures an address for the interface.
		• Ensure that CE devices connected to the PE device are on the same network.

Step 4	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.

Verifying VRF-Aware Tunnels

Use the following commands to verify Virtual Routing and Forwarding (VRF)-aware tunnels:

SUMMARY STEPS

1. **show tunnel interface**
2. **show ip route** *ip-address*
3. **show ip route vrf** *vrf-name ip-address*
4. **ping ipv6** *ipv6-address* **source** *ipv6-address*
5. **ping vrf** *vrf-name ipv6-address* **source** *ipv6-address*
6. **debug ipv6 icmp**

DETAILED STEPS

Step 1 **show tunnel interface**

This command displays detailed information about all tunnel interfaces.

Example:

The following is sample output from a provider edge (PE) device with Generic Routing Encapsulation (GRE) tunnel mode:

```
Device# show tunnel interface

Tunnel0
   Mode:GRE/IP, Destination 10.44.44.44, Source Loopback2
   IP transport: output interface Ethernet1/0 next hop 10.0.0.2,
   Tunnel header destination 10.44.44.44
   Application ID 1: unspecified
```

```
   Linestate - current up, cached up
   Internal linestate - current up, evaluated up
```

Example:
The following is sample output from a PE device with IPv6/IP tunnel mode:

```
Device# show tunnel interface

Tunnel0
   Mode:IPv6/IP, Destination 44.44.44.44, Source Loopback2
   IP transport: output interface Ethernet1/0 next hop 2.0.0.2,
   Tunnel header destination 44.44.44.44
   Application ID 1: unspecified
   Linestate - current up, cached up
   Internal linestate - current up, evaluated up
```

The output is displayed and the tunnel mode is observed.

Step 2 **show ip route** *ip-address*
This command displays detailed routing information to a tunnel destination address.

Example:
The following is sample output from a PE device with the tunnel endpoint in the global routing table:

```
Device# show ip route 10.44.44.44

  Routing entry for 10.44.44.44/32
  Known via "ospf 1", distance 110, metric 21, type intra area
  Last update from 10.0.0.2 on Ethernet1/0, 01:10:25 ago
  Routing Descriptor Blocks:
  * 10.0.0.2, from 10.44.44.44, 01:10:25 ago, via Ethernet1/0
      Route metric is 21, traffic share count is 1
```

The following is sample output from a PE device having tunnel endpoints in the VRF table:

```
Device# show ip route 10.44.44.44

% Network not in table
```

The output is displayed and you can observe if the tunnel destination is in the global routing table or not.

Step 3 **show ip route vrf** *vrf-name ip-address*
This command displays detailed routing information to a destination IP address.

Example:
The following is sample output from PE1:

```
Device# show ip route vrf green 10.4.4.4

Routing entry for 10.4.4.4/32
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.0.0.2, via Ethernet1/0
      Route metric is 0, traffic share count is 1
```

The tunnel destination address 10.4.4.4 is not in the global routing table.

Step 4 **ping ipv6** *ipv6-address* **source** *ipv6-address*
This command displays the status of the connectivity between two devices.

Example:
The following is sample output from a customer edge (CE) device CE1 with a **ping** command issued to CE2:

```
Device# ping ipv6 2001:DB8:2::1 source 2001:DB8:1::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/2/4 ms
```

Step 5 **ping vrf** *vrf-name ipv6-address* **source** *ipv6-address*
The VRF-ping tests the VPN connection.

Example:
The following is sample output from CE1 with a **ping vrf** command issued to CE2:

```
Device# ping vrf green ipv6 2001:DB8:2::1 source 2001:DB8:1::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2::1, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:1::2%green
!!!!!
```

If the displayed output indicates success, the VPN is configured correctly.

Step 6 **debug ipv6 icmp**
This command displays debugging messages for IPv6 Internet Control Message Protocol (ICMP) transactions.

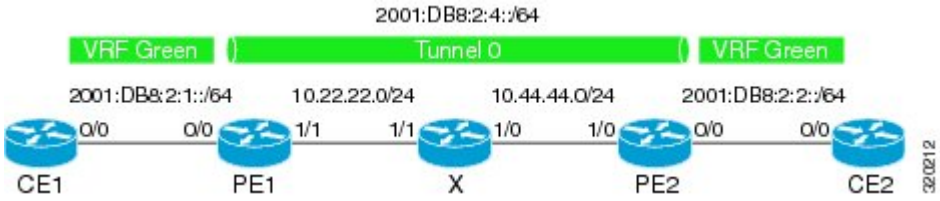
Example:
The following is sample output:

```
Device# debug ipv6 icmp

  ICMP Packet debugging is on

*Apr  6 14:08:10.743: ICMPv6: Received echo request, Src=2001:DB8:1::2, Dst=2001:DB8:2::1
*Apr  6 14:08:10.743: ICMPv6: Sent echo reply, Src=2001:DB8:2::1, Dst=2001:DB8:1::2
...
```

If the displayed output indicates success, the VPN is configured correctly.

Configuration Examples for VRF-Aware Tunnels

Example: Configuring a VRF-Aware Tunnel (Tunnel Endpoint in Global Routing Table)



Example: Configuring CE1

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
interface Ethernet0/0
 vrf forwarding green
 no ip address
 ipv6 address 2001:DB8:2:1::1/64
 no shutdown
 exit
!
!
ipv6 route vrf green 2001:DB8:2:2::/64 2001:DB8:2:1::2
ipv6 route vrf green 2001:DB8:2:4::/64 2001:DB8:2:1::2
!
```

Example: Configuring PE1

```
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
interface Tunnel0
 no ip address
 vrf forwarding green
 ipv6 address 2001:DB8:2:4::1/64
 tunnel source 10.22.22.22
 tunnel destination 10.44.44.44
 exit
```

```
!
interface Ethernet0/0
 vrf forwarding green
 no ip address
 ipv6 address 2001:DB8:2:1::2/64
 no shutdown
 exit
!
interface Ethernet1/1
 no ip address
 ip address 10.22.22.22 255.255.255.0
 no shutdown
 exit
!
ip route 10.44.44.0 255.255.255.0 10.22.22.23
ipv6 route vrf green 2001:DB8:2:2::/64 Tunnel0 2001:DB8:2:4::2
```

Example: Configuring PE2

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
interface Tunnel0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:4::2/64
 tunnel source 10.44.44.44
 tunnel destination 10.22.22.22
 exit
!
interface Ethernet0/0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:2::1/64
 no shutdown
 exit
!
interface Ethernet1/0
 no ip address
 ip address 10.44.44.44 255.255.255.0
 no shutdown
 exit
!
ip route 10.22.22.0 255.255.255.0 10.44.44.43
!
ipv6 route vrf green 2001:DB8:2:1::/64 Tunnel0 2001:DB8:2:4::1
!
```

Example: Configuring CE2

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
```

```
address-family ipv6
exit-address-family
exit
!
interface Ethernet0/0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:2::2/64
 no shutdown
 exit
!
!
ipv6 route vrf green 2001:DB8:2:1::/64 2001:DB8:2:2::1
ipv6 route vrf green 2001:DB8:2:4::/64 2001:DB8:2:2::1
!
```

Example: Configuring Device X

```
!
interface Ethernet1/0
 no ip address
 ip address 10.44.44.43 255.255.255.0
 no shutdown
 exit
!
interface Ethernet1/1
 no ip address
 ip address 10.22.22.23 255.255.255.0
 no shutdown
 exit
!
```

Example: Verifying the Tunnel Configuration

From CE1

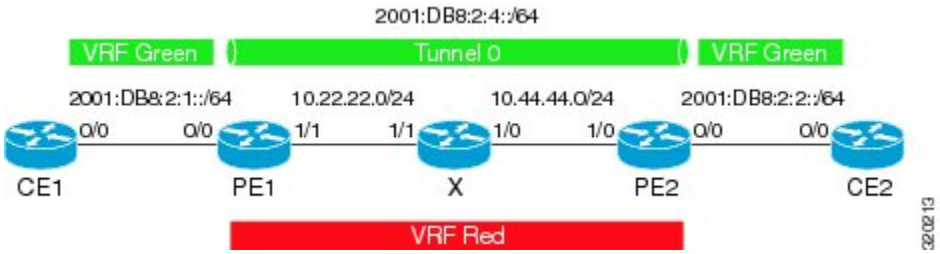
```
Device# ping vrf green ipv6 2001:db8:2:2::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/6 ms

Device# ping vrf green ipv6 2001:db8:2:2::2 source 2001:db8:2:1::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:2:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

From PE1

```
Device# show tunnel interface

Tunnel0
   Mode:GRE/IP, Destination 10.44.44.44, Source 10.22.22.22
   IP transport: output interface Ethernet1/1 next hop 10.22.22.23
   Application ID 1: unspecified
   Linestate - current up
   Internal linestate - current up, evaluated up
   Tunnel Source Flags: Local
   Transport IPv4 Header DF bit cleared
   OCE: IP tunnel decap
   Provider: interface Tu0, prot 47
     Performs protocol check [47]
```

```
     Protocol Handler: GRE: opt 0x0
       ptype: ipv4 [ipv4 dispatcher: punt]
       ptype: ipv6 [ipv6 dispatcher: from if Tu0]
       ptype: mpls [mpls dispatcher: drop]
       ptype: otv [mpls dispatcher: drop]
       ptype: generic [mpls dispatcher: drop]
There are 0 tunnels running over the EON IP protocol
There are 0 tunnels running over the IPinIP protocol
There are 0 tunnels running over the NOSIP protocol
There are 0 tunnels running over the IPv6inIP protocol
There are 0 tunnels running over the RBSCP/IP protocol


Device# show ip route 10.44.44.44

Routing entry for 10.44.44.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.22.22.23
      Route metric is 0, traffic share count is 1

Device# debug ipv6 icmp

  ICMP Packet debugging is on
*Jan  1 10:57:37.882: ICMPv6: Sent R-Advert, Src=FE80::A8BB:CCFF:FE00:5200, Dst=FF02::1
*Jan  1 11:00:18.634: ICMPv6: Received R-Advert, Src=FE80::A8BB:CCFF:FE00:5200,Dst=FF02::1
```

Example: Configuring a VRF-Aware Tunnel (Tunnel Endpoint in VRF)



Example: Configuring CE1

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
interface Ethernet0/0
 vrf forwarding green
 no ip address
 ipv6 address 2001:DB8:2:1::1/64
 no shutdown
 exit
!
!
ipv6 route vrf green 2001:DB8:2:2::/64 2001:DB8:2:1::2
ipv6 route vrf green 2001:DB8:2:4::/64 2001:DB8:2:1::2
!
```

Example: Configuring PE1

```
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
vrf definition red
rd 2:2
route-target export 2:2
route-target import 2:2
address-family ipv4
exit-address-family
exit
!
interface Tunnel0
 no ip address
 vrf forwarding green
 ipv6 address 2001:DB8:2:4::1/64
 tunnel source 10.22.22.22
 tunnel destination 10.44.44.44
 tunnel vrf red
 exit
!
interface Ethernet0/0
 vrf forwarding green
 no ip address
 ipv6 address 2001:DB8:2:1::2/64
 no shutdown
 exit
!
interface Ethernet1/1
 vrf forwarding red
 no ip address
 ip address 10.22.22.22 255.255.255.0
 no shutdown
 exit
!
ip route vrf red 10.44.44.0 255.255.255.0 10.22.22.23
ipv6 route vrf green 2001:DB8:2:2::/64 Tunnel0 2001:DB8:2:4::2
```

Example: Configuring PE2

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
vrf definition red
rd 2:2
route-target export 2:2
route-target import 2:2
address-family ipv4
exit-address-family
exit
!
```

```
interface Tunnel0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:4::2/64
 tunnel source 10.44.44.44
 tunnel destination 10.22.22.22
 tunnel vrf red
 exit
!
interface Ethernet0/0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:2::1/64
 no shutdown
 exit
!
interface Ethernet1/0
 vrf forwarding red
 no ip address
 ip address 10.44.44.44 255.255.255.0
 no shutdown
 exit
!
ip route vrf red 10.22.22.0 255.255.255.0 10.44.44.43
!
ipv6 route vrf green 2001:DB8:2:1::/64 Tunnel0 2001:DB8:2:4::1
!
```

Example: Configuring CE2

```
!
ipv6 unicast-routing
ipv6 cef
!
vrf definition green
rd 1:1
route-target export 1:1
route-target import 1:1
address-family ipv6
exit-address-family
exit
!
interface Ethernet0/0
 vrf forwarding green
 no ipv6 address
 ipv6 address 2001:DB8:2:2::2/64
 no shutdown
 exit
!
!
ipv6 route vrf green 2001:DB8:2:1::/64 2001:DB8:2:2::1
ipv6 route vrf green 2001:DB8:2:4::/64 2001:DB8:2:2::1
!
```

Example: Configuring Device X

```
!
interface Ethernet1/0
 vrf forwarding red
 no ip address
 ip address 10.44.44.43 255.255.255.0
 no shutdown
 exit
!
interface Ethernet1/1
 vrf forwarding red
 no ip address
```

```
 ip address 10.22.22.23 255.255.255.0
 no shutdown
 exit
!
```

Example: Verifying the Tunnel Configuration

From CE1

```
Device# ping vrf green ipv6 2001:db8:2:2::2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/6 ms

Device# ping vrf green ipv6 2001:db8:2:2::2 source 2001:db8:2:1::1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:DB8:2:2::2, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:2:1::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

From PE1

```
Device# show tunnel interface

Tunnel0
   Mode:GRE/IP, Destination 10.44.44.44, Source 10.22.22.22
   IP transport: output interface Ethernet1/1 next hop 10.22.22.23
   Application ID 1: unspecified
   Linestate - current up
   Internal linestate - current up, evaluated up
   Tunnel Source Flags: Local
   Transport IPv4 Header DF bit cleared
   OCE: IP tunnel decap
   Provider: interface Tu0, prot 47
     Performs protocol check [47]
     Protocol Handler: GRE: opt 0x0
       ptype: ipv4 [ipv4 dispatcher: punt]
       ptype: ipv6 [ipv6 dispatcher: from if Tu0]
       ptype: mpls [mpls dispatcher: drop]
       ptype: otv [mpls dispatcher: drop]
       ptype: generic [mpls dispatcher: drop]
There are 0 tunnels running over the EON IP protocol
There are 0 tunnels running over the IPinIP protocol
There are 0 tunnels running over the NOSIP protocol
There are 0 tunnels running over the IPv6inIP protocol
There are 0 tunnels running over the RBSCP/IP protocol

Device# show ip route 10.44.44.44

% Network not in table

Device# show ip route vrf red 10.44.44.44

Routing Table: red
Routing entry for 10.44.44.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.22.22.23
      Route metric is 0, traffic share count is 1

Device# debug ipv6 icmp

  ICMP Packet debugging is on
*Jan  1 10:57:37.882: ICMPv6: Sent R-Advert, Src=FE80::A8BB:CCFF:FE00:5200, Dst=FF02::1
*Jan  1 11:00:18.634: ICMPv6: Received R-Advert, Src=FE80::A8BB:CCFF:FE00:5200,Dst=FF02::1
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*Cisco IOS IPv6 Configuration Guide*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IPv6 commands	Cisco IOS IPv6 Command Reference
Cisco IOS IPv6 features	Cisco IOS IPv6 Feature Mapping

Standard/RFC	Title
RFCs for IPv6	*IPv6 RFCs*

Standards and RFCs

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 VRF-Aware 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 25: Feature Information for VRF-Aware Tunnels

Feature Name	Releases	Feature Information
VRF-Aware Tunnels	Cisco IOS XE Release 3.8S	Virtual Routing and Forwarding (VRF)-aware tunnels are used to connect customer networks separated by untrusted core networks or core networks with different infrastructures (IPv4 or IPv6). The following command was modified to support IPv6 transport: **tunnel vrf**.

Prerequisites for VRF-Aware Tunnels

- You must configure customer edge networks. See the Configuring Customer Edge Networks for Tunneling section.
- You must configure the customer and transport VRFs. See the Defining a VRF Instance section.