Implementing Generic Routing Encapsulation

Generic Routing Encapsulation (GRE) is a tunneling protocol developed by Cisco Systems that encapsulates a wide variety of network layer protocols inside virtual point-to-point links over an Internet Protocol internetwork.

Feature History for Configuring Link Bundling on Cisco IOS XR Software

Release	Modification
Release 4.3.0	These feature were supported on the Cisco ASR 9000 Series Aggregation Services Routers: • MPLS/L3VPNoGRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • RSVP/TEoGRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • VRF aware GRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • L2VPN (VPWS and VPLS) on GRE for ASR 9000 Enhanced Ethernet Line Card only
Release 5.1.1	Support for GRE Tunnel Key and Tunnel Key-Ignore was introduced.
Release 5.2.2	Support for GRE tunnel on an IPv6 transport network.
Release 5.3.2	Support for GRE IPv4 Transport Over MPLS was introduced.
Release 6.0.1	Support for GRE IPv6 Transport Over MPLS was introduced.

Prerequisites for Configuring Generic Routing Encapsulation

Before configuring Link Bundling, be sure that these tasks and conditions are met:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Generic Routing Encapsulation

To implement the GRE feature, you must understand these concepts:

GRE Overview

Generic Routing Encapsulation (GRE) tunneling protocol provides a simple generic approach to transport packets of one protocol over another protocol by means of encapsulation.

GRE encapsulates a payload, that is, an inner packet that needs to be delivered to a destination network inside an outer IP packet. GRE tunnel endpoints send payloads through GRE tunnels by routing encapsulated packets through intervening IP networks. Other IP routers along the way do not parse the payload (the inner packet); they only parse the outer IP packet as they forward it towards the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded to it's ultimate destination.

MPLS networks provide VPN functionality by tunneling customer data through public networks using routing labels. Service Providers (SP) provide MPLS L3VPN, 6PE/6VPE and L2VPN services to their customers who have interconnected private networks.

MPLS and L3VPN are supported over regular interfaces on Cisco ASR 9000 Series Aggregation Services Routers through GRE tunnels over an IPv4 transport network. MPLS support is extended over IPv4 GRE tunnels between routers as the provider core may not be fully MPLS aware.

GRE Features

The following sections list the GRE features:

Note An IPv6 GRE tunnel does not support features that involve transport of MPLS packets through a GRE tunnel.

MPLS/L3VPN over GRE

The MPLS VPN over GRE feature provides a mechanism for tunneling Multiprotocol Label Switching (MPLS) packets over a non-MPLS network. This feature utilizes MPLS over generic routing encapsulation (MPLSoGRE) to encapsulate MPLS packets inside IP tunnels. The encapsulation of MPLS packets inside IP tunnels creates a virtual point-to-point link across non-MPLS networks.

L3VPN over GRE basically means encapsulating L3VPN traffic in GRE header and its outer IPv4 header with tunnel destination and source IP addresses after imposing zero or more MPLS labels, and transporting it across the tunnel over to the remote tunnel end point. The incoming packet can be a pure IPv4 packet or an MPLS packet. If the incoming packet is IPv4, the packet enters the tunnel through a VRF interface, and if the incoming packet is MPLS, then the packet enters through an MPLS interface. In the IPv4 case, before encapsulating in the outer IPv4 and GRE headers, a VPN label corresponding to the VRF prefix and any IGP

label corresponding to the IGP prefix of the GRE tunnel destination is imposed on the packet. In the case of MPLS, the top IGP label is swapped with any label corresponding to the GRE tunnel destination address.

PE-to-PE Tunneling

The provider-edge-to-provider-edge (PE-to-PE) tunneling configuration provides a scalable way to connect multiple customer networks across a non-MPLS network. With this configuration, traffic that is destined to multiple customer networks is multiplexed through a single GRE tunnel.

Note A similar nonscalable alternative is to connect each customer network through separate GRE tunnels (for example, connecting one customer network to each GRE tunnel).

As shown in the following figure, the PE devices assign VPN routing and forwarding (VRF) numbers to the customer edge (CE) devices on each side of the non-MPLS network.

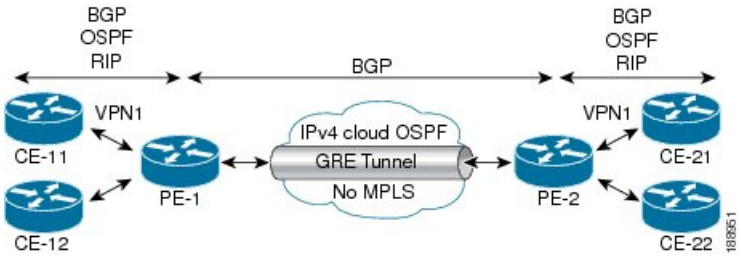
The PE devices use routing protocols such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), or Routing Information Protocol (RIP) to learn about the IP networks behind the CE devices. The routes to the IP networks behind the CE devices are stored in the associated CE device's VRF routing table.

The PE device on one side of the non-MPLS network uses the routing protocols (that operate within the non-MPLS network) to learn about the PE device on the other side of the non-MPLS network. The learned routes that are established between the PE devices are then stored in the main or default routing table.

The opposing PE device uses BGP to learn about the routes that are associated with the customer networks that are behind the PE devices. These learned routes are not known to the non-MPLS network.

The following figure shows BGP defining a static route to the BGP neighbor (the opposing PE device) through the GRE tunnel that spans the non-MPLS network. Because routes that are learned by the BGP neighbor include the GRE tunnel next hop, all customer network traffic is sent using the GRE tunnel.

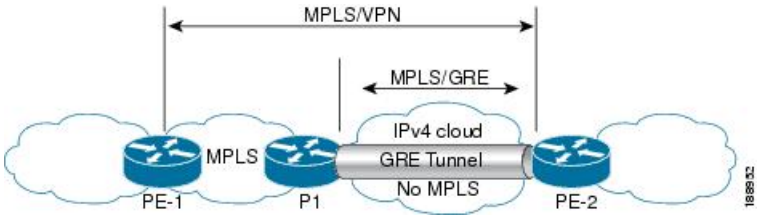
Figure 1: PE-to-PE Tunneling



P-to-PE Tunneling

As shown in the following figure, the provider-to-provider-edge (P-to-PE) tunneling configuration provides a way to connect a PE device (P1) to an MPLS segment (PE-2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single GRE tunnel.

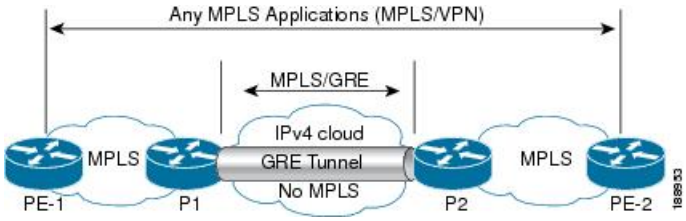
Figure 2: P-to-PE Tunneling



P-to-P Tunneling

As shown in the following figure, the provider-to-provider (P-to-P) configuration provides a method of connecting two MPLS segments (P1 to P2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single GRE tunnel.

Figure 3: P-to-P Tunneling



6PE/6VPE

Service Providers (SPs) use a stable and established core with IPv4/MPLS backbone for providing IPv4 VPN services. The 6PE/6VPE feature facilitates SPs to offer IPv6 VPN services over this backbone without an IPv6 core. The provide edge (PE) routers run MP-iBGP (Multi-Protocol iBGP) to advertise v6 reachability and v6 label distribution. For 6PE, the labels are allocated per IPv6 prefix learnt from connected customer edge (CE) routers and for 6VPE, the PE router can be configured to allocate labels on a per-prefix or per-CE/VRF level.

6PE/6VPE over GRE

While IPv4/MPLS allows SPs to transport IPv6 traffic across IPv4 core (IPv6 unaware), MPLS over GRE allows MPLS traffic to be tunneled through MPLS unaware networks. These two features together facilitate IPv6 traffic to be transported across IPv6 as well as MPLS unaware core segments. Only the PE routers need to be aware of MPLS and IPv6 (Dual stack).

The 6PE/6VPE over GRE feature allows the use of IPv4 GRE tunnels to provide IPv6 VPN over MPLS functionality to reach the destination v6 prefixes via the BGP next hop through MPLS & IPv6 unaware core.

MPLS Forwarding

When IPv6 traffic is received from one customer site, the ingress PE device uses MPLS to tunnel IPv6 VPN packets over the backbone toward the egress PE device identified as the BGP next hop. The ingress PE device prefixes the IPv6 packets with the outer and inner labels before placing the packet on the egress interface.

Under normal operation, a P device along the forwarding path does not lookup the frame beyond the first label. The P device either swaps the incoming label with an outgoing one or removes the incoming label if the next device is a PE device. Removing the incoming label is called penultimate hop popping. The remaining

label (BGP label) is used to identify the egress PE interface toward the customer site. The label also hides the protocol version (IPv6) from the last P device, which it would otherwise need to forward an IPv6 packet.

A P device is ignorant of the IPv6 VPN routes. The IPv6 header remains hidden under one or more MPLS labels. When the P device receives an MPLS-encapsulated IPv6 packet that cannot be delivered, it has two options. If the P device is IPv6 aware, it exposes the IPv6 header, builds an Internet Control Message Protocol (ICMP) for IPv6 message, and sends the message, which is MPLS encapsulated, to the source of the original packet. If the P device is not IPv6 aware, it drops the packet.

6PE/6VPE over GRE

As discussed earlier, 6PE/6VPE over GRE basically means enabling IPv6/IPv6 VPN over MPLS over GRE.

The ingress PE device uses IPv4 generic routing encapsulation (GRE) tunnels combined with 6PE/6VPE over MPLS to tunnel IPv6 VPN packets over the backbone toward the egress PE device identified as the BGP next hop.

The PE devices establish MP-iBGP sessions and MPLS LDP sessions just as in the case of 6PE/6VPE. The difference here is that these sessions are established over GRE tunnels, which also means that the PEs are just one IGP hop away. The P routers in the tunnel path only need to forward the traffic to the tunnel destination, which is an IPv4 address.

This is how the IPv6 LSP is setup for label switching the IPv6 traffic:

- After the LDP and BGP sessions are established, the PEs exchange IPv6 prefixes that they learn from the CEs and the corresponding IPv6 labels, just as in the case of IPv4 VPN.
- The IPv6 labels occupy the inner most position in the label stack.
- The IPv4 labels corresponding to the PE IPv4 addresses occupy the outer position in the stack.
- When IPv6 traffic needs to be forwarded from PE1 to PE2, the outer PE2 IPv4 label is used to label switch the traffic to PE2, and the inner IPv6 label is used to send the packet out of the interface connected to the CE.

GRE Tunnel Key

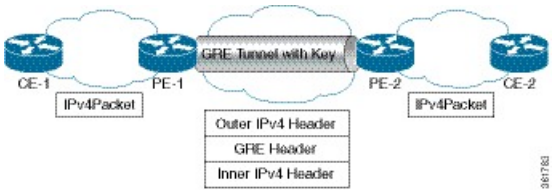
The GRE Tunnel Key feature enables the encapsulation router to add a four-byte key, as part of the GRE header, during encapsulation. In the decapsulation router, the GRE key of an incoming packet should match the key value configured under the GRE tunnel. During decapsulation, if a mismatch between the key value of the incoming GRE packet and the key value configured under the GRE tunnel is identified, the incoming packet is dropped.

Note

- GRE tunnel key feature is supported only on Cisco ASR 9000 Enhanced Ethernet line cards. It is mandatory to have ingress and egress line cards as Enhanced Ethernet line cards.
- Either the same key or different keys can be configured under multiple GRE tunnels for a given router. However, more than one tunnel, having the same tunnel source and destination but a different tunnel key is not supported because the source and destination pair for various configured tunnels must be unique irrespective of the key value. Also, two tunnels with the same tunnel source and destination, but one tunnel being with key and the other tunnel being without key is not supported.
- Different traffic streams passing through the same GRE tunnel contains the same GRE key configured for that tunnel.
- Use the **tunnel key** command to configure the key value at both ends of a GRE tunnel.

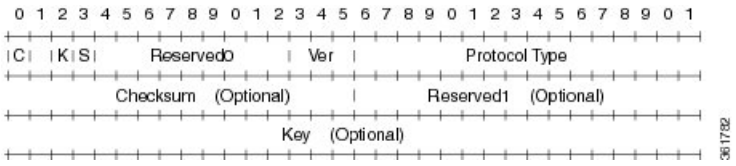
The following figure shows a simple representation of the GRE tunnel key configuration:

Figure 4: GRE Tunnel with Key



The following figure shows the complete format of the GRE header with the key field:

Figure 5: GRE Header



GRE Tunnel Key-Ignore

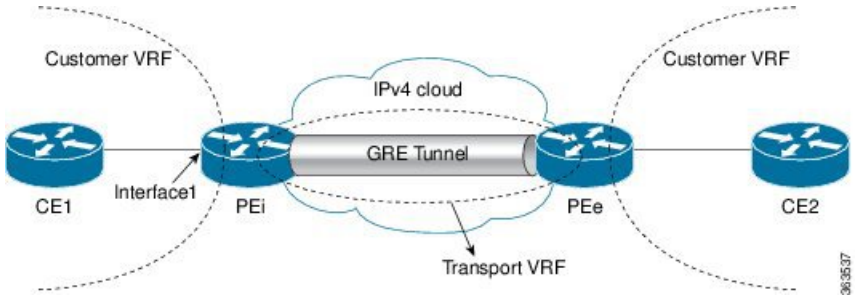
If a GRE key is configured on only one endpoint router of a GRE tunnel, the other router that has no GRE key configured discards any incoming tunnel packet that has a GRE key. To enable this router to ignore GRE keys and accept incoming data plane packets on the GRE tunnel, run the **tunnel key-ignore** command. Control plane packets over a GRE tunnel are accepted only if there is no GRE tunnel key configured on both the tunnel endpoints or both the endpoints are configured with a GRE key and the control plane packet passes the GRE key validation. Hence, in the above scenario, both the routers discard any incoming control plane packets from the GRE tunnel.

Note Do not configure a GRE key on the GRE tunnel endpoint router if you have configured the router to ignore GRE keys. Configuring a GRE key overrides the **tunnel key-ignore** command and thus cancels the skipping of GRE key validation. This results in the router accepting from the incoming tunnel traffic only those packets that have the matching GRE key.

GRE tunnel in VRF domains

You can configure an IPv4/IPv6 GRE tunnel between two interfaces that belong to a Virtual Forwarding and Routing (VRF) instance. This contains or limits the tunnel path within this specific VRF instance. For example, packets can be sent internally within a default or non-default VRF instance separated through an intermediate VRF that contains the GRE tunnel.

Figure 6: GRE tunnel in a VRF instance



In the above topology, a GRE tunnel is configured in the core network, which is an IPv4 cloud. For packets entering through Interface1, the provider edge (PE) devices PEi and PEe are the tunnel head and tunnel exit respectively.

The VRF configured on Interface1 is the customer VRF. Packets entering this interface are routed using this customer VRF to the tunnel. The routing by the customer VRF is called inner IP packet routing. You can configure the tunnel to be visible to the customer VRF instance using the **vrf** *vrf-name* command. This enables only the configured VRF instance to use the tunnel, that is, forward traffic from PEi into this tunnel and also receive all incoming PEi tunnel packets.

The VRF configured on the tunnel using the **tunnel vrf** command is the transport VRF. The packet entering the tunnel is encapsulated with the tunnel source and destination addresses. The transport VRF routes this encapsulated payload between the tunnel endpoints. The routing by the transport VRF is the outer IP packet routing. If no transport VRF is configured for the tunnel, the PEi device looks up the tunnel endpoint addresses in the default VRF instance, that is, the global routing table.

Restrictions on a GRE tunnel

The following restrictions are applicable for a GRE tunnel:

- GRE over BVI is not supported.
- MPLS packets cannot be transported within an IPv6 GRE tunnel. Therefore, the following features are not supported on an IPv6 GRE tunnel:
  - MPLS/L3VPN over GRE
  - 6PE/6VPE
  - 6PE/6VPE over GRE
- Multicast packets cannot be transported within an IPv6 GRE tunnel.
- Multicast packets cannot be transported within an IPv4 GRE tunnel that is configured in a transport VRF.
- Keep-Alive packets are not supported on an IPv6 GRE tunnel. You can use the Bidirectional Forwarding Detection (BFD) protocol to detect link failures in an IPv6 GRE tunnel.
- The IPv4 addresses are mandatory for configuring GRE tunnels under the VRF, as this would ensure the traffic flows through the tunnel in an expected manner. Use either an IP unnumbered interface or a loopback interface belonging to that VRF for establishing the GRE tunnels under a VRF. Though the tunnel may come up without the aforementioned configuration, the traffic may not pass over the GRE tunnel, since the IP information on the tunnel interface is not available for forwarding the traffic correctly.

Also, for the VRF information to be written in hardware database the IP information is required. Therefore, the IP unnumbered GRE tunnels may not work as expected as they may not forward traffic on the device.

GRE IPv4/IPv6 Transport Over MPLS

The Generic Routing Encapsulation (GRE) IPv4/IPv6 transport over Multiprotocol Label Switching (MPLS) feature provides a mechanism to configure GRE tunnels, where the tunnel destination IPv4/IPv6 address is reachable through an MPLS label switched path (LSP). With this feature, IPv4, IPv6, routing protocols - OSPF, ISIS, and L2VPN and L3VPN packets are accepted as payload packets for GRE encapsulation . IPv4/IPv6 is supported as the GRE delivery protocol.

This feature overcomes the restriction of not being able to configure the tunnel destination endpoint through an MPLS LSP during tunnel configuration.

The GRE IPv4/IPv6 transport over MPLS feature facilitates creation of GRE tunnels over LSPs, through L3VPN inter-AS (autonomous system) options:

- External Border Gateway Protocol (EBGP) redistribution of labeled VPN IPv4/IPv6 routes from an AS to a neighboring AS.
- Multi-hop EBGP redistribution of labeled VPN IPv4/IPv6 routes between source and destination ASs, with EBGP redistribution of labeled IPv4/IPv6 routes from an AS to a neighboring AS.

Multipoint GRE IPv4/IPv6 transport over MPLS is also supported.

The GRE IPv4/IPv6 transport over MPLS feature is supported on the following types of Cisco ASR 9000 line cards:

- Cisco ASR 9000 Enhanced Ethernet line card
- Cisco ASR 9000 High Density 100GE Ethernet line card

Limitations

- GRE IPv4/IPv6 transport over MPLS-TE tunnels is not supported.
- GREoMPLS with IP Fast Reroute (IPFRR).

How to Configure Generic Routing Encapsulation

Configuring a GRE Tunnel

Perform this task to configure a GRE tunnel.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **vrf** *vrf-name*

4. **ipv4 address** *ipv4-address mask*
5. **tunnel mode gre** {*ipv4* | *ipv6*}
6. **tunnel source** { *ip-address* | **type** *path-id* }
7. **tunnel destination** *ip-address*
8. **tunnel vrf** *transport-vrf-name*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 4000
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vrf1
```

(Optional) Specifies the VRF domain that can route packets into and from the tunnel.

Note This step is not required if the tunnel is available for global routing and therefore, is not specific to a VRF.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 5 **tunnel mode gre** {*ipv4* | *ipv6*}

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel mode gre ipv4
```

Specify whether the transport network is an IPv4 or IPv6 network. The default GRE tunnel mode is IPv4.

Note The tunnel source and destination addresses should match the tunnel mode. A mismatch in configuration causes the tunnel to fail without any error message.

Step 6 **tunnel source** { *ip-address* | **type** *path-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1
```

Specifies the source of the tunnel interface.

Note It is recommended that the tunnel source is identified using the interface ID and not the IP address. Using the interface ID enables the router to mark the tunnel as down when the interface is down and the routing protocol tries to find and use an alternate route to the tunnel route.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 145.12.5.2
```

Defines the tunnel destination.

Step 8 **tunnel vrf** *transport-vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel vrf vrf99
```

(Optional) Associates the transport VRF with the tunnel. The transport VRF contains the interfaces over which the tunnel sends as well as receives packets (outer IP packet routing).

Note This step is not required if the tunnel endpoints belong to the global routing table.

Step 9 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Tunnel Key

Perform this task to configure the tunnel key for the GRE encapsulated packets. You need to perform same configuration steps on the other endpoint router of the tunnel ensuring that the key value is the same at both the local and remote GRE interfaces.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **ipv4 address** *ipv4-address subnet-mask*
4. **tunnel key** *value*
5. *(Optional)* **tunnel tos** *tos-value*
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 10
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 101.0.9.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **tunnel key** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel key 10
```

Enables tunnel key.

Step 5 *(Optional)* **tunnel tos** *tos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel tos 96
```

Specifies the value of the TOS field in the tunnel encapsulating packets.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source Loopback10
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 33.0.9.33
```

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Tunnel Key-Ignore

Perform this task to configure the tunnel key-ignore for the GRE encapsulated packets. You need to perform same configuration steps on the other endpoint router of the tunnel.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **ipv4 address** *ipv4-address subnet-mask*

4. **tunnel key-ignore**
5. *(Optional)* **tunnel tos** *tos-value*
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 10
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 101.0.9.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **tunnel key-ignore**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel key-ignore
```

Enables tunnel key-ignore.

Step 5 *(Optional)* **tunnel tos** *tos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel tos 96
```

Specifies the value of the TOS field in the tunnel encapsulating packets.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source Loopback10
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 33.0.9.33
```

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a VRF Interface

Perform this task to configure a VRF interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 100
```

Enters interface configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vrf_A
```

Configures a VRF instance and enters VRF configuration mode.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0
```

Configures a primary IPv4 address for the specified interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VRF Routing Protocol

Perform this task to configure the VRF routing protocol.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | *type interface-path-id*}
5. **area** *area-id*
6. **interface** *type interface-path-id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 109
```

Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **router-id** {*router-id* | *type interface-path-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```

Configures the router ID for the OSPF routing process.

Step 5 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0
```

Configures the OSPF area as area 0.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/3/0/0
```

Associates interface GigabitEthernet 0/3/0/0 with area 0.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring IGP for Remote PE Reachability

Perform this task to configure IGP for remote PE reachability.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** {*router-id*}
4. **area** *area-id*
5. **interface tunnel-ip** *number*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Step 3 **router-id** {*router-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# router-id 1.1.1.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

Step 5 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring LDP on GRE Tunnel

Perform this task to configure LDP on a GRE tunnel.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** {*router-id*}
4. **interface tunnel-ip** *number*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **mpls ldp**

Example:

```
RP/0/RSP0/CPU0:router(config)# mpls ldp
```

Enables MPLS LDP configuration mode.

Step 3 **router-id** {*router-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# router-id 1.1.1.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 4 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MP-iBGP to Exchange VPN-IPv4 Routes

Perform this task to configure MP-iBGP to exchange VPN-IPv4 routes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **router-id** *ip-address*
4. **neighbor** *ip-address*
5. **remote-as** *as-number*
6. **update-source** *type interface-path-id*
7. **address-family { vpnv4 | vpnv6 unicast }**
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **router-id** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# router-id 1.1.1.1
```

Configures the local router with a specified router ID.

Step 4 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 4.4.4.4
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 5 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 1
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 6 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#update-source Loopback0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

Step 7 **address-family { vpnv4 | vpnv6 unicast }**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
```

Enters address family configuration submode for the specified address family.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for Generic Routing Encapsulation

This section provides examples to configure GRE:

Configuring an IPv4 GRE Tunnel: Example

This example shows how to configure an IPv4 GRE tunnel:

```
configure
  interface tunnel-ip1
   ipv4 address 12.0.0.1 255.255.255.0
   tunnel source Loopback0
   tunnel destination 200.200.200.1
end
```

Configuring an IPv6 GRE Tunnel: Example

```
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.2/24
 ipv6 address 10::2/64
 tunnel mode gre ipv6
 tunnel source GigabitEthernet 0/0/0/0
 tunnel destination 100::1
 tunnel vrf BLUE
 !
```

Verifying GRE tunnel Configuration: Example

```
vrf blue
 description connected to IXIA in blue VRF
 address-family ipv4 unicast
  import route-target
   100:1
  !
  export route-target
100:1
!

vrf red
 description connected to core interface in red VRF
 address-family ipv4 unicast
  import route-target
   200:1
  !
  export route-target
200:1
```

```
!


interface tunnel-ip1
 vrf blue
 ipv4 address 10.10.10.1 255.255.255.0
 tunnel source Loopback0
 keepalive
 tunnel vrf red
 tunnel destination 12.12.12.12


RP/0/RSP0/CPU0:ios#ping vrf red 12.12.12.12
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.12, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms


RP/0/RSP0/CPU0:ios#ping vrf blue 10.10.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/19 ms
```

Configuring Global VRF: Example

This example shows how to configure global VRF:

```
configure
 vrf VRF1
  address-family ipv4 unicast
  import route-target 120:1
  export route-target 120:2
exit
exit
  router bgp120
  vrf VRF1
  rd auto
end
```

Configuring a VRF Interface: Example

This example shows how to configure a VRF interface:

```
configure
 interface tunnel-ip 100
  vrf VRF1
  ipv4 address 1.1.1.1 255.255.255.0
  ipv6 address 100::2/64
end
```

Configuring VRF Routing Protocol: Example

This example shows how to configure VRF routing protocol:

```
configure
  router ospf109
   vrf VRF1
   router-id 172.20.10.10
   area0
   interface GigabitEthernet0/3/0/0
end
```

Configuring IGP for Remote PE Reachability: Example

This example shows how to configure IGP for remote provider edge (PE) reachability:

```
configure
  router ospf109
   router-id 172.20.10.10
   area0
   interface tunnel-ip1
end
```

Configuring LDP on GRE Tunnel: Example

This example shows how to configure LDP on a GRE tunnel:

```
configure
 mpls ldp
 router-id 172.20.10.10
  interface tunnel-ip1
 end
```

Configuring MP-iBGP to Exchange VPN-IPv4 Routes: Example

This example shows how to configure MP-iBGP to exchange VPN-IPv4 routes:

```
configure
 router bgp100
  router-id 172.20.10.10
  neighbor 2.2.2.2 remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
end
```