Configure GRE Tunnels

Tunneling provides a mechanism to transport packets of one protocol within another protocol. This chapter describes GRE tunneling protocol.

Release	Feature(s) Added
Release 7.3.1	GRE Tunnel feature was introduced.

GRE Tunnels

Table 1: Feature History Table

Feature Name	Release Information	Description
Disabling time-to-live (TTL) decrement at GRE encapsulation	Release 7.3.2	This feature allows you to disable the time-to-live (TTL) decrement of the incoming packets. The result is that encapsulation of the original incoming packet takes place without any change in the TTL value. This feature avoids dropping incoming packets with a TTL value equal to one after GRE encapsulation. Before this release, the TTL value of incoming packets was decremented by one before GRE decapsulation. This feature introduces the **tunnel ttl disable** command.

Feature Name	Release Information	Description
GRE Tunnel	Release 7.3.1	Generic Routing Encapsulation (GRE) provides a simple approach to transporting packets of one protocol over another protocol using encapsulation. This capability is now extended to the Cisco 8000 Series Routers. This feature supports: • Unidirectional GRE encapsulation • Unidirectional GRE decapsulation And introduces the following commands: • **show interface tunnel accounting (encap)** • **show interface tunnel accounting (decap)**
Outer-header hashing support for MPLSoGRE and IPoGRE traffic	Release 7.3.1	This feature allows load-balancing of GRE traffic in transit routers. A transit node distributes incoming GRE traffic evenly across all available ECMP links in a GRE tunnel topology. A hashing function uses GRE outer and inner header tuples such as source IP, destination IP, protocol, and router ID to determine traffic entropy. This capability is now extended to the Cisco 8000 Series Routers.

Generic Routing Encapsulation (GRE) is a tunneling protocol that 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 should be delivered to a destination network inside an outer IP packet. The GRE tunnel behaves as virtual point-to-point link that has two endpoints identified by the tunnel source and tunnel destination address. The 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 toward the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded to the packet's ultimate destination.

A tunnel configured using encapsulation mode performs encapsulation of IPv4/IPv6 payload inside the GRE header. A tunnel configured using decapsulation mode performs the opposite. Here, outer GRE header is decapsulated and the inner IPv4/IPv6/MPLS payload is forwarded to the next hop router. Both encapsulation and decapsulation tunnel interfaces collect statistics periodically. The statistics can be displayed on demand using the CLI commands `show interface tunnel-ip1 accounting` and `show policy-map type pbr address-family ipv4 statistics`. For more information, see Unidirectional GRE Encapsulation (GREv4), on page 5 and Unidirectional GRE Decapsulation (GREv4), on page 5.

To perform load-balancing of GRE traffic in transit routers, a transit node distributes incoming GRE traffic evenly across all available ECMP links in a GRE tunnel topology. Furthermore, to determine traffic entropy, a hashing function uses GRE outer and inner header tuples such as source IP, destination IP, protocol, and router ID.

Supported Features on a GRE Tunnel

GRE tunnel supports the following features:

- GRE or IP-in-IP tunnels support 16 unique source addresses. These 16 unique source addresses are repeated multiple times to configure 1000 encapsulation tunnels or 64 decapsulation tunnels.
- GRE encapsulation supports the following features:
  - IPv4/IPv6 over GRE IPv4 transport
  - MPLS PoP over GRE IPv4 transport
  - ABF (Access List Based Forwarding) v4/v6 over GRE
  - VRF (Virtual Routing and Forwarding) support over GRE
- GRE decapsulation supports the following features:
  - PBR-based GRE decapsulation configuration
  - CLI-based GRE decapsulation configuration
  - IPv4/IPv6 over GRE decapsulation
  - MPLS/SRTE over GRE decapsulation
  - A GRE tunnel in decapsulation mode has only tunnel source configured, without any tunnel destination address. This decapsulated GRE tunnel behaves like a P2MP (Point-to-multipoint) tunnel, which means that an incoming GRE packet can have any source IP address and matching destination IP address to the tunnel source configured. However, once a source IP address is used for decapsulated P2MP tunnel, it cannot be re-used with other decapsulation tunnels.
- The command `tunnel ttl disable` is supported. This command controls TTL decrement of a packet being encapsulated. After configuring this command fo a tunnel interface, TTL value of incoming packet is not decremented by one, and original incoming packet is encapsulated without changing the TTL. By default, `tunnel ttl disable` isn't configured. This means that the TTL of incoming packets is decremented by one before GRE encapsulation.

  For example, consider an incoming packet that had the TTL value equal to one. On GRE encapsulation, the TTL value is decremented by one and becomes zero. Therefore the router will discard the packet and send an ICMP message back to the originating host. Using this feature, you can disable TTL decrement and avoid the packet discard.

  Configuration Example

```
Router#configure
Router(config)#interface tunnel-ip30016
Router(config-if)#tunnel ttl disable
Router(config-if)#commit
```

Limitations for Configuring GRE Tunnels

This list describes the limitations for configuring GRE tunnels:

- GRE tunnels configured without any decapsulation or encapsulation mode support only ERPSAN feature.
- Don't create multiple GRE/IP-in-IP tunnels with the same pair of source and destination IP address or interface name. Configure all tunnels with unique source-destination pairs. In an encapsulation or decapsulation tunnel where only either source or destination is mentioned, the source-destination pair should also be unique when compared to other encapsulation or decapsulation tunnels.

- Bi-directional GRE tunnel isn't supported.
- Routing protocols over GRE tunnels aren't supported.
- Multicast over GRE isn't supported.
- GRE KA (Keep Alive) isn't supported.
- GRE parameters such as MTU (Maximum Transmission Unit) and key functionalities aren't supported.

Configure GRE Tunnels

Configuring a GRE tunnel involves creating a tunnel interface and defining the tunnel source and destination. This example shows how to configure a GRE tunnel between source and destination. The router supports only uni-directional GRE with either encapsulation or decapsulation mode.

```
Router# configure
Router(config)# interface tunnel-ip1
Router(config-if)# ipv4 address 101.0.1.2 255.255.255.0
Router(config-if)# ipv6 address 101:0:1::2/64
Router(config-if)# tunnel mode gre ipv4 [encap | decap]
Router(config-if)# tunnel source 2.2.1.1
Router(config-if)# tunnel destination 2.2.2.1/32
Router(config-if)# commit
Router(config-if)# exit
```

To configure ABFv4/v6 over GRE:

```
router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-ip1
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-ip1

ipv4 access-list abf-gre
  1 permit ipv4 any any nexthop1 ipv4 201.0.1.2
ipv6 access-list abf6-gre
  1 permit ipv6 any any nexthop1 ipv6 201:0:1::2

interface HundredGigE0/0/0/24
  ipv4 address 24.0.1.1/24
  ipv6 address 24:0:1::1/64
  ipv4 access-group abf-gre ingress
  ipv6 access-group abf6-gre ingress
!
```

To configure MPLS PoP label over GRE:

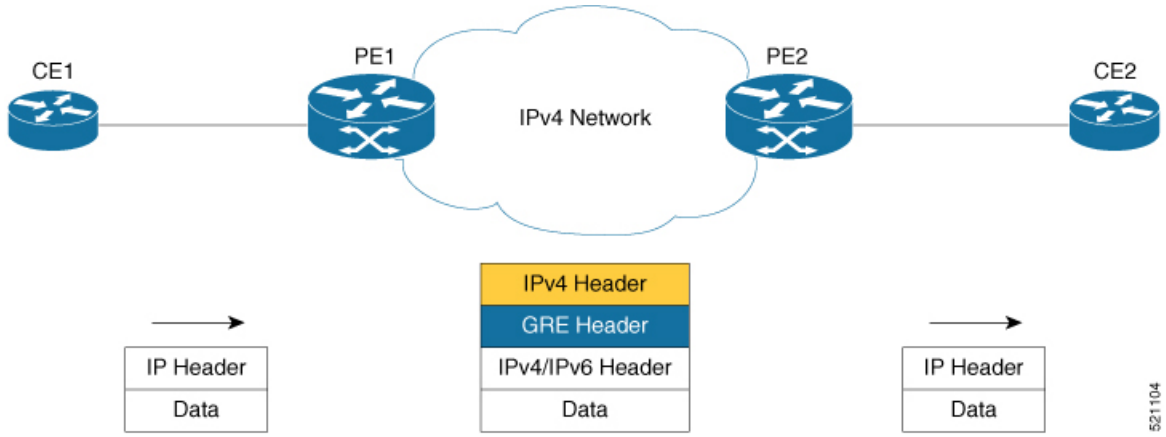
```
router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-ip1
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-ip1

mpls static
  interface HundredGigE0/0/0/24
  lsp gre
    in-label 30501 allocate
    forward path 1 resolve-nexthop 201.0.1.2 out-label pop
!
```

Note Bi-directional GRE tunnel supports only ERSPAN.

Unidirectional GRE Encapsulation (GREv4)

A tunnel configured using encapsulation mode performs encapsulation of IPv4/IPv6 payload inside the GRE header. The following figure shows GRE encapsulation. Routers in the IP cloud have no knowledge of encapsulated IP source address or destination address.



Configuration

The following example shows how to configure GRE tunnel encapsulation:

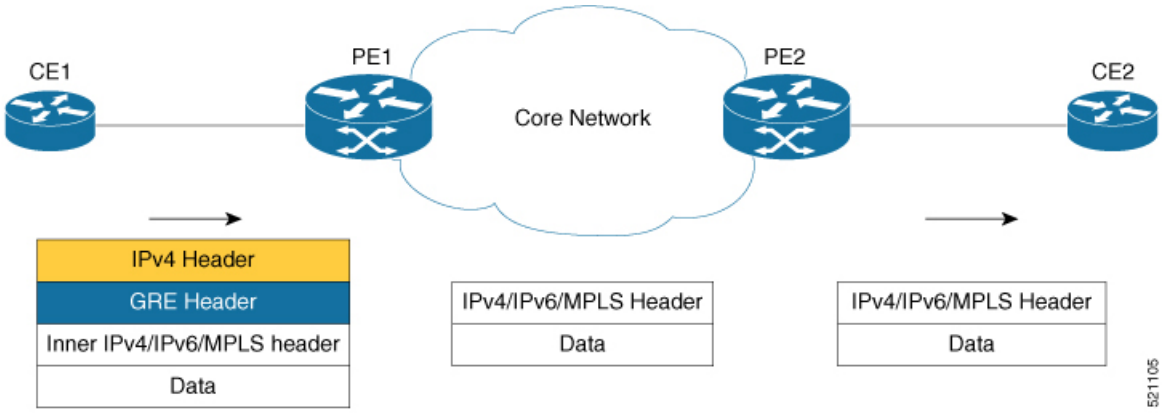
```
interface tunnel-ip1
  ipv4 address 101.0.1.1/24
  ipv6 address 101:0:1::1/64
  tunnel mode gre ipv4 encap
  tunnel source [ loopback1 | <any-ipaddres> | any-interface]
  tunnel destination [ 20.0.1.1/32 | 20.0.1.0/24 | 20.0.1.0/28]

router static
  address-family ipv4 unicast
  201.0.1.0/24 tunnel-1

router static
  address-family ipv6 unicast
  201:0:1::0/64 tunnel-1
```

Unidirectional GRE Decapsulation (GREv4)

In unidirectional GRE decapsulation, the outer GRE header is decapsulated and the inner IPv4/IPv6/MPLS payload is forwarded to the next hop router. The following figure shows GRE decapsulation. In the figure, PE1 strips off outer GRE header and inner payload is forwarded as regular IPv4/IPv6/MPLS forwarding.



Configuration

There are two methods to configure GRE tunnel decapsulation:

1. CLI-based tunnel decapsulation configuration

```
interface tunnel-ip1
  ipv4 address 101.0.1.1/24
  ipv6 address 101:0:1::1/64
  tunnel mode gre ipv4 decap
  tunnel source [ loopback1 | <any-ipaddres> | any-interface]
  tunnel destination [ 20.0.1.1/32 | 20.0.1.0/24 | 20.0.1.0/28]
```

2. PBR-based tunnel decapsulation configuration

```
class-map type traffic match-all test_gre1
                match protocol gre
                match destination-address ipv4 10.0.1.2 255.255.255.255
                match source-address ipv4 10.10.10.1 255.255.255.255
                end-class-map
policy-map type pbr P1-test
                class type traffic test_gre1  decapsulate gre
vrf-policy vrf default address-family ipv4 policy type pbr input P1-test
```