

Configuring Layer 2 Services over GRE

This chapter describes how to configure layer 2 services over generic routing encapsulation (GRE) in your ERX edge router.



Note: *It is important to be familiar with configuring both Frame Relay and GRE tunnels before configuring layer 2 services over GRE.*

Topic	Page
Overview	4-1
How Layer 2 Services over GRE Work	4-3
Configuring Frame Relay over GRE	4-7
Configuration Example	4-10
Monitoring Layer 2 Services over GRE	4-13

Overview

Many Internet service providers are concerned with the expense and implementation time to transition from mature layer 2 networks, such as Frame Relay or ATM, to optical networks based on IP and/or MPLS. By emulating circuits or services using GRE tunnels, layer 2 services over GRE allow service providers to use their existing equipment base and move to a packet-based optical infrastructure. Emulation allows existing circuits and/or services to be carried across the new infrastructure without your having to employ new services, thus allowing the internetworking of disparate networks.

Encapsulating layer 2 services such as Frame Relay in GRE tunnels leverages use of the public Internet while continuing to provide end-to-end Frame Relay congestion control and management plane functionality that Frame Relay users expect. By encapsulating all traffic

received for a given Frame Relay data-link connection identifier (DLCI) into a GRE tunnel, a service provider must own both endpoints of the tunnel. If both ends are ERX systems, intervendor compatibility of a Network-to-Network Interface (NNI) is not a concern.

Figure 4-1 depicts a service provider's network that uses layer 2 services with GRE tunneling. Customers using Frame Relay or ATM connections to ERX edge routers are unaware that GRE tunneling is used.

The ERX layer 2 service currently supported is Frame Relay over GRE.

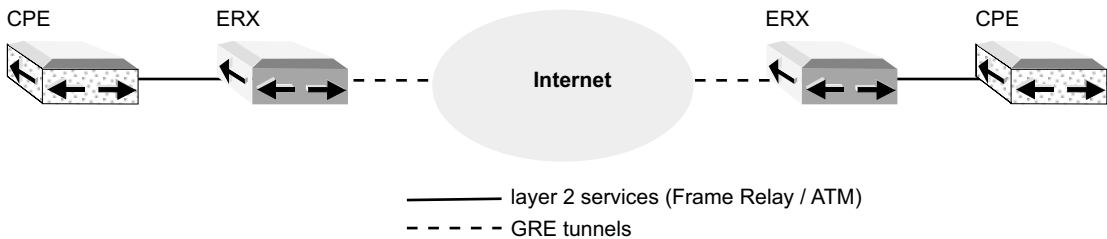


Figure 4-1 Layer 2 services over a provider's GRE network

References

For information about layer 2 services, see:

- Framework for Pseudo Wire Emulation Edge-to-Edge (PWE3) – draft-pate-pwe3-framework-01.txt (December 2002 expiration)

For information about Frame Relay, see:

- *ERX Physical and Link Layers Configuration Guide, Chapter 11, Configuring Frame Relay*
- RFC 1490 – Multiprotocol Interconnect over Frame Relay (July 1993)
- RFC 2115 – Management Information Base for Frame Relay DTEs Using SMIPv2 (September 1997)
- RFC 2233 – The Interfaces Group MIB using SMIPv2 (November 1997)

For information about GRE tunnels, see:

- *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 4, Configuring IP Tunnels*
- *ERX Physical and Link Layers Configuration Guide, Chapter 9, Managing Tunnel Service and IPSec Service Interfaces*
- *ERX Physical and Link Layers Configuration Guide, Chapter 20, Configuring SMDS*

- RFC 1701 – Generic Routing Encapsulation (October 1994)
- RFC 1702 – Generic Routing Encapsulation over IPv4 Networks (October 1994)
- RFC 2003 – IP Encapsulation within IP (October 1996)
- RFC 2784 – Generic Routing Encapsulation (GRE) (March 2000)
- RFC 2890 – Key and Sequence Number Extensions to GRE (September 2000)

How Layer 2 Services over GRE Work

Figure 4-2 shows the connection between customer premises equipment (CPE) and an ERX router. The figure illustrates layer 2 service layering in a Frame Relay scenario where GRE tunneling is used.

GRE encapsulates IP packets to enable data transmission through an IP tunnel. The resulting encapsulated packet contains a GRE header and a delivery header. As shown in Figure 1-2, the Tunnel Service line module (TSM) depicts IP over GRE. In actuality, GRE prepends an IP header based on the configured tunnel source and the IP destination address (DA).

Frame Relay traffic routed over GRE tunneling is transparent to the user. To accomplish transparency, two things must occur. First, an NNI link management interface (LMI) must be established on the GRE Frame Relay interface. All control frames are generated on DLCI 0 or 1023, depending on which type of LMI is configured; then they are passed into the GRE tunnel, as is other user traffic.

Second, LMI information from the UNI-data communications equipment (DCE) interface on the customer-facing interface must be translated and communicated to the tunnel ingress Frame Relay interface.

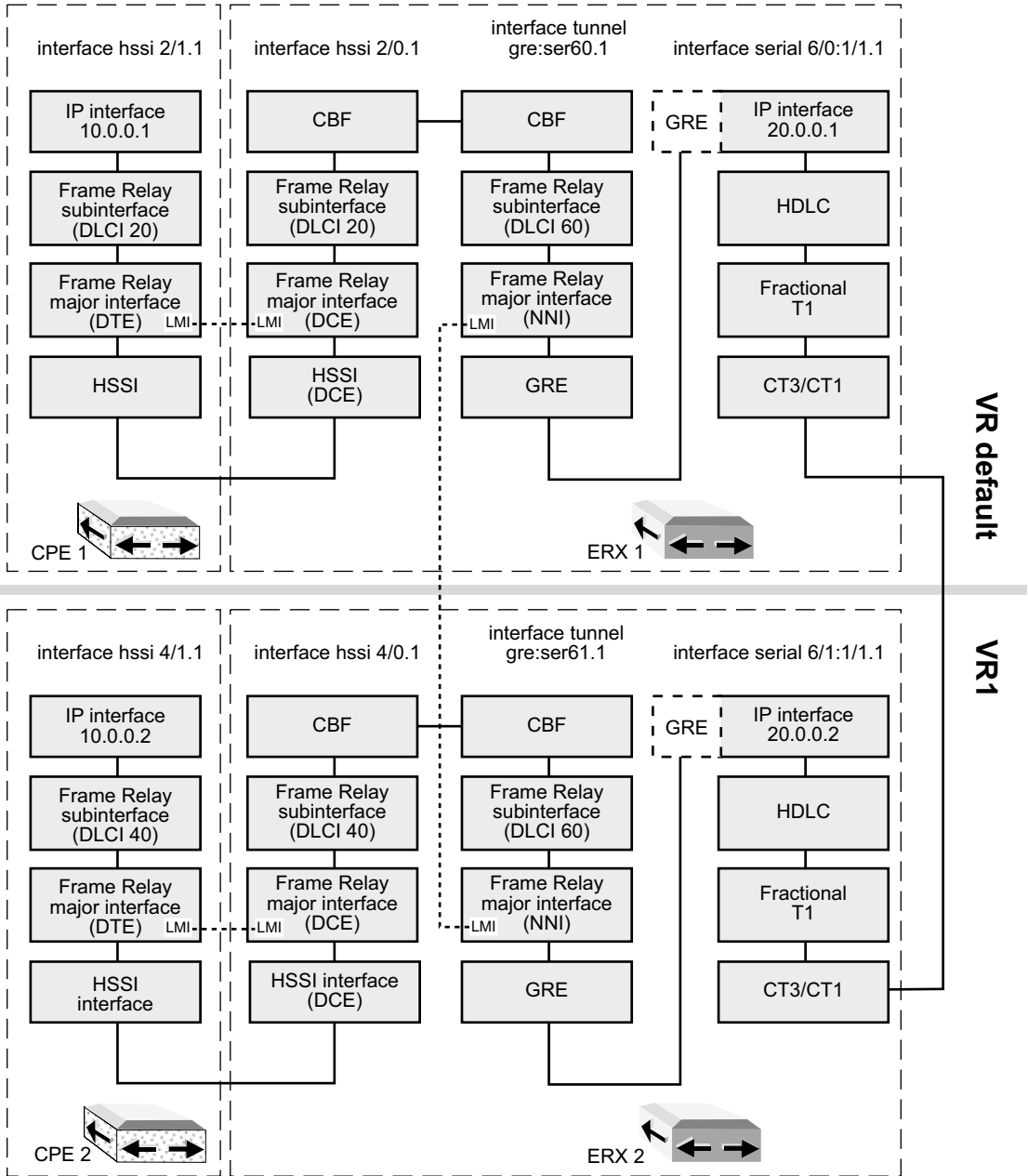


Figure 4-2 Frame Relay over GRE layering

As shown in Figure 4-2, the DLCIs at the endpoints of a CBF connection do not have to be the same. Different DLCI values are acceptable because

a Frame Relay subinterface stamps its DLCI into the packet received from the upper layer when it is a CBF connection.

DLCI values must match between physically connected interfaces, as well as tunnel-connected interfaces, however. In Figure 1-2, the peer NNI interface on the other end of the service provider network must have the same DLCIs configured as on the TSM.

Link Status Mapping

Previously, Frame Relay interface status information on the ERX system was passed only between physically connected peers. LMI traffic was passed between a customer and the ERX customer-facing interfaces, as well as between the GRE Frame Relay interfaces. Link status information, however, was not end-to-end.

In Figure 4-3, link state information can be exchanged only between pairs A and B, B and C, and C and D. If a link failure occurred between A and B, for example, C and D would not know. For this reason, link state mapping is provided between the internally connected Frame Relay interfaces on the ERX system.

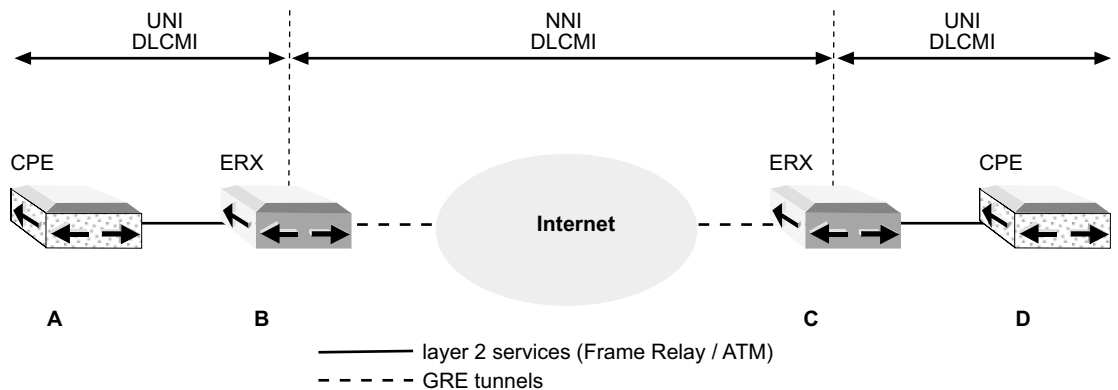


Figure 4-3 End-to-end data-link connection management interface (DLCMI) discontinuity without link state mapping

Through normal LMI polling, the peer major interface at the other end of the connection discovers any DLCI failures. If a DLCI failure on an NNI or a DCE UNI interface occurs, an asynchronous notification is sent to the peer interface, which achieves end-to-end link status monitoring.

When a single DLCI is marked inactive, the CBF-connected DLCI also is marked inactive and that subinterface is marked dormant. If a major interface goes down, LMI marks all DLCIs for that major interface as inactive and all subinterfaces for that major interface as

LowerLayerDown. As a result, all the CBF-connected subinterfaces are marked dormant and their associated circuits are marked inactive.

DLCI Mapping

In addition to mapping link state, DLCIs for raw Frame Relay frames are translated when they are transferred between CBF-connected interfaces.

The DLCI that represents the circuit between the CPE and the customer-facing ERX port is translated to the DLCI that represents the circuit in the GRE tunnel. Frame Relay frames that traverse a CBF connection experience this DLCI replacement while at the same time keeping the backward explicit congestion notification (BECN), forward explicit congestion notification (FECN), and discard eligibility (DE) bits intact in the header.

GRE Encapsulation

When a frame is received by a tunnel-facing Frame Relay subinterface, it is passed to GRE for encapsulation. Traffic for all configured DLCIs on a given Frame Relay major interface is multiplexed into a single GRE tunnel.

One of the features of Frame Relay is that packets are delivered in the order they are received. However, when Frame Relay traffic is tunneled through GRE onto the Internet, the in-order arrival of frames at the GRE tunnel endpoint cannot be guaranteed. Because of this uncertainty, it is strongly recommended that GRE sequence number generation be enabled.

GRE Sequence Numbers

GRE sequence numbers provide some protection against out-of-order delivery. Both sides of the GRE tunnel must enable sequence number generation to provide in-order delivery. Sequence number generation is disabled unless you enable it via the **tunnel sequence-datagrams** command. GRE packets that arrive with a sequence number in the header are always checked, regardless of whether or not sequence number generation is enabled.

GRE encapsulation is in compliance with RFC 2784 and RFC 2890.

Configuring Frame Relay over GRE

Before you configure Frame Relay over GRE, configure the CPE-facing and core-facing Frame Relay interfaces. Once Frame Relay interfaces have been created, you must connect the two Frame Relay subinterfaces by layering a CBF interface above both and establishing a CBF connection.



Note: To configure Frame Relay interfaces, see *ERX Physical and Link Layers Configuration Guide, Chapter 11, Configuring Frame Relay*.

To configure Frame Relay over GRE on the CPE-facing interface:

- 1 Specify the physical interface on which you want to configure Frame Relay. For example:

```
host1(config)#interface serial 1/0
host1(config-if)#frame-relay intf-type dce
```

- 2 Enable Frame Relay encapsulation.

```
host1(config-if)#encapsulation frame-relay ietf
```

- 3 Create and configure a Frame Relay subinterface.

```
host1(config-if)#interface serial 1/0.1
```

- 4 Configure a DLCI on a Frame Relay subinterface.

```
host1(config-if)#frame-relay interface-dlci 100 ietf
```

- 5 Create a CBF interface.

```
host1(config-subif)#cbf
```

- 6 Create and configure a GRE tunnel interface.

```
host1(config)# interface tunnel gre:east
host1(config-if)#tunnel source 20.0.0.2
host1(config-if)#tunnel destination 20.0.0.2
host1(config-if)#tunnel mtu 10240
host1(config-if)#tunnel sequence-datagrams
```

- 7 Enable Frame Relay encapsulation.

```
host1(config-if)#encapsulation frame-relay ietf
host1(config-if)#frame-relay intf-type nni
```

- 8 Create and configure a Frame Relay subinterface.

```
host1(config-if)#interface tunnel gre:east.1
```

- 9 Configure a DLCI on a Frame Relay subinterface.

```
host1(config-subif)#frame-relay interface-dlci 200 ietf
```

- 10 Create a CBF interface.

```
host1(config-subif)#cbf
```

- 11 Create a bidirectional CBF connection between the two CBF interfaces.

```
host1(config)#cbf connection serial 1/0.1 tunnel gre:east.1
```

cbf

- Use to create a CBF interface over the Frame Relay interface that you are configuring.

- Example

```
host1(config-if)#cbf
```

- Use the **no** version to remove the CBF interface.

cbf connection

- Use to create a bidirectional CBF connection between two CBF interfaces.
- Note that the bidirectional connection consists of a symmetrical pair of unidirectional connections, one from *interface1* to *interface2* and a second from *interface2* to *interface1*.
- Example – the following example creates a CBF connection between the CT3 in slot 1, port 0 and the GRE tunnel east.

```
host1(config)#cbf connection serial 1/0.1 tunnel gre:east
```

- Use the **no** version to remove an existing connection.

interface tunnel

- Use to create a GRE IP tunnel interface.
- Specify the type of tunnel as **gre**, and assign a name to the tunnel.
- Example

```
host1(config)#interface tunnel gre:east
```

- Use the **no** version to remove the tunnel.

tunnel destination

- Use to configure the remote end of the tunnel.
- Specify either the IP address of an interface on the remote system or the hostname of the remote system.
 - › The IP address is the address for the destination interface.
 - › The hostname is the name of the destination interface.
- Example 1

```
host1(config)#interface tunnel gre:tunnel2
```

```
host1(config-if)#tunnel destination 192.13.7.1
```

- Example 2

```
host1(config)#interface tunnel gre:tunnel2
host1(config-if)#tunnel destination remoteHost
```

- Use the **no** version to remove the destination of a tunnel.

tunnel mtu

- Use to set the MTU for the tunnel.
- Specify a value in the range 1024 to 10240 bytes.
- Example

```
host1(config-if)#tunnel mtu 1024
```

- Use the **no** version to restore the default, 10240 bytes.

tunnel sequence-datagrams

- Use to enable GRE sequence numbers.
- You should specify GRE sequence numbers at both ends of the GRE tunnel.
- Example

```
host1(config)#tunnel sequence-datagrams
```

- Use the **no** version to disable sequence numbers.

tunnel source

- Use to configure the source of the tunnel.
- Specify either the primary IP address or the type and number of a new interface.
- Do not specify an unnumbered interface.
- Example 1

```
host1(config)#interface tunnel gre:boston-tunnel-1
host1(config-if)#tunnel source 192.10.2.1
```

- Example 2

```
host1(config)#interface tunnel gre:boston-tunnel-1
host1(config-if)#tunnel source atm 5/0.12
```

- Use the **no** version to remove the source of a tunnel.

Configuration Example

The script provided in this section is one way to configure Frame Relay services over GRE. Explanation notes are provided within the script. You must change the script for your specific configuration.

Figure 4-4 provides a topology example, which further explains the configuration script.

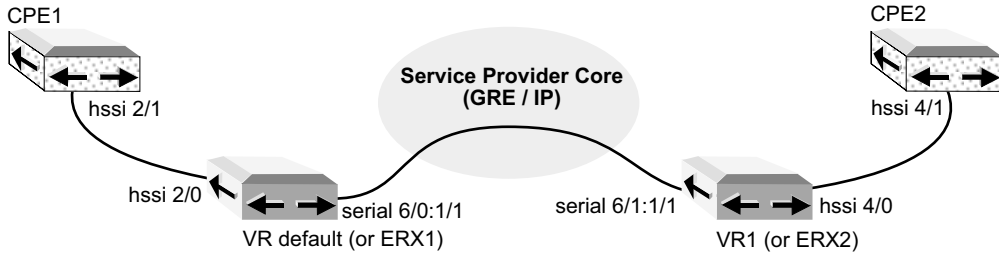


Figure 4-4 Configuration example of Frame Relay over GRE

```

! Juniper Networks Edge Routers ERX-700
! Version: (x.x.x)
! Copyright (c) 1999-2001 Juniper Networks, Inc. All rights
  reserved.
!!
!-----
!Create VR1 and VR default.
!-----

    virtual-router vrl
    !
    virtual-router default
!-----
!Configure CT3 module (VR default and VR1).
!-----

!
controller t3 6/0
  no shutdown
  clock source internal module
  t1 1 clock source internal module
  t1 1/1 timeslots 1-24 speed 64
!
controller t3 6/1
  no shutdown
  clock source internal module
  t1 1 clock source internal module
  t1 1/1 timeslots 1-24 speed 64

```

```
!
!-----
!Configure HSSI module (CPE1 and VR default) over which Frame Relay traffic
will pass.
!-----

interface hssi 2/0
  hssi internal-clock
  encapsulation frame-relay ietf
  frame-relay intf-type dce
  frame-relay lmi-type ansi
!
interface hssi 2/0.1
  frame-relay interface-dlci 20 ietf
  cbf
!
interface hssi 2/1
  encapsulation frame-relay ietf
  frame-relay lmi-type ansi
!
interface hssi 2/1.1
  frame-relay interface-dlci 20 ietf
  ip address 10.0.0.1 255.255.255.252
!
!-----
!Configure HSSI module (CPE2 and VR1) over which Frame Relay traffic will
pass.
!-----

interface hssi 4/0
  hssi internal-clock
  encapsulation frame-relay ietf
  frame-relay intf-type dce
  frame-relay lmi-type ansi
!
interface hssi 4/0.1
  frame-relay interface-dlci 40 ietf
  cbf
!
interface hssi 4/1
  encapsulation frame-relay ietf
  frame-relay lmi-type ansi
!-----
!Configure a GRE tunnel on VR default on which Frame Relay traffic can flow.
!-----

!
interface tunnel gre:ser60 transport-virtual-router default
  tunnel source 20.0.0.1
  tunnel destination 20.0.0.2
```

```
tunnel sequence-datagrams
tunnel mtu 10240
encapsulation frame-relay ietf
frame-relay intf-type nni
frame-relay lmi-type ansi
!
interface tunnel gre:ser60.1
frame-relay interface-dlci 60 ietf
cbf

!-----
!Configure a GRE tunnel on VR1 on which Frame Relay traffic can flow.
!-----

interface tunnel gre:ser61 transport-virtual-router vr1
tunnel source 20.0.0.2
tunnel destination 20.0.0.1
tunnel sequence-datagrams
tunnel mtu 10240
encapsulation frame-relay ietf
frame-relay intf-type nni
frame-relay lmi-type ansi
!
interface tunnel gre:ser61.1
frame-relay interface-dlci 60 ietf
cbf
!
interface serial 6/0:1/1
encapsulation hdlc
ip address 20.0.0.1 255.255.255.252
!

!-----
!Create bidirectional CBF connection.
!-----

cbf connection hssi 2/0.1 tunnel gre:ser60.1
cbf connection tunnel gre:ser61.1 hssi 4/0.1
!
! =====
!
virtual-router vr1
!
interface hssi 4/1.1
frame-relay interface-dlci 40 ietf
ip address 10.0.0.2 255.255.255.252
!
interface serial 6/1:1/1
encapsulation hdlc
ip address 20.0.0.2 255.255.255.252
```

Monitoring Layer 2 Services over GRE

You can monitor layer 2 services over GRE using the following **show cbf connection**, **show frame-relay subinterface tunnel**, and **show gre tunnel** commands.

show cbf connection

- Use to display all configured CBF connections.
- Example

```
host1#show cbf connection
      serial 5/1:1.1 <--> tunnel gre:tunnel0.1
      serial 5/1:1.2 <--> tunnel gre:tunnel1.1
host1#show cbf connection all
      serial 5/1:1.1 --> tunnel gre:tunnel0.1
      serial 5/1:1.2 --> tunnel gre:tunnel1.1
      tunnel gre:tunnel1.1 --> serial 5/1:1.2
      tunnel gre:tunnel0.1 --> serial 5/1:1.1
```

show frame-relay subinterface tunnel

- Use to display the state of a Frame Relay subinterface over a specified tunnel.
- The tunnel subinterface can be in one of the following states:
 - › Up – traffic can flow on the interface
 - › Offline – traffic cannot flow because hardware is unavailable
 - › Down – traffic cannot flow because of a problem in the interface at the current protocol layer
 - › LowerLayerDown – traffic cannot flow because of a problem in an interface at a lower protocol layer
 - › AdministrativelyDown – traffic cannot flow because of manual administrative intervention
 - › Dormant – indicates that the CBF-connected DLCI is inactive, possibly caused by a customer misconfiguration
- Use the optional **delta** keyword to specify that baselined statistics are to be shown.
- The **brief** keyword displays only the operational status of all configured subinterfaces.
- Field descriptions
 - › sub-interface – identifies the subinterface in *slot/port:channel/subchannel.subinterface* format
 - › status – status of the subinterface
 - › Time since last status change – time since the last status change on the subinterface
 - › In bytes – number of inbound bytes received on the subinterface
 - › In frames – number of inbound frames received on the interface

- › In errors – number of inbound errors received on the subinterface
- › In discards – number of inbound packets discarded
- › In unknown protos – number of packets received on the subinterface with unknown protocols
- › Out bytes – number of outbound bytes transmitted on the subinterface
- › Out frames – number of outbound frames transmitted on the interface
- › Out errors – number of outbound errors transmitted on the subinterface
- › Out discards – number of outbound packets discarded
- Example

```
host1#show frame-relay subinterface tunnel gre:tunnel0.1
Frame relay sub-interface tunnel gre:tunnel0.1, status is up
Number of sub-interface down transitions is 0
Time since last status change 01:42:15
  In bytes: 0                      Out bytes: 0
  In frames: 0                    Out frames: 0
  In errors: 0                    Out errors: 0
  In discards: 0                  Out discards: 0
  In unknown protos: 0
```

```
host1#show frame-relay subinterface brief
Frame relay sub-interface hssi2/0.1, status is up
Frame relay sub-interface hssi2/1.1, status is up
Frame relay sub-interface hssi4/0.1, status is up
Frame relay sub-interface tunnel gre:ser60.1, status is up
Frame relay sub-interface tunnel gre:ser61.1, status is up
Frame relay sub-interface hssi4/1.1, status is up
```

show gre tunnel

- Use to display information about a GRE tunnel or a list of GRE tunnel subinterfaces.
- You can view the number of GRE tunnels in a specific state.
- You can view the number of GRE tunnels associated with either an IP address or with an IP address on a virtual router.
- Field descriptions
 - › Tunnel name and status – name of tunnel and status: up or down
 - › Tunnel mtu – maximum transmission unit for the tunnel
 - › Tunnel source address – IP address of the source of the tunnel
 - › Tunnel destination address – IP address of the destination of the tunnel
 - › Tunnel transport virtual router – virtual router associated with the tunnel
 - › Tunnel checksum option – state of the checksum feature: enabled or disabled
 - › Tunnel sequence number option – state of the sequence number option: enabled or disabled

- › Tunnel up/down trap – indicates whether or not the ERX system sends traps to SNMP when the operational state of the tunnels changes
- › Tunnel server location – location of the tunnel server in slot/port format.
- › Tunnel administrative state – configured state of the tunnel: up or down
- › Statistics – details of packets received or transmitted by the tunnel
- › Packets – number of packets received or transmitted by the tunnel
- › Octets – number of octets received or transmitted by the tunnel
- › Discards – number of packets not accepted by the tunnel
- › Errors – number of packets with errors received or transmitted by the tunnel
- › Data rx – received data
- › Data tx – transmitted data

- Example

```

host1#show gre tunnel detail
GRE tunnel ser60 is Up
Tunnel operational configuration
  Tunnel mtu is '10240'
  Tunnel source address is '20.0.0.1'
  Tunnel destination address is '20.0.0.2'
  Tunnel transport virtual router is default
  Tunnel checksum option is disabled
  Tunnel sequence number option is enabled (628 tx / 628 rx)
  Tunnel up/down trap is enabled
  Tunnel-server location is 3/0
  Tunnel administrative state is Up
Statistics      packets      octets      discards      errors
  Data rx      627          97381        0             0
  Data tx      628          97423        0             0
GRE tunnel ser61 is Up
Tunnel operational configuration
  Tunnel mtu is '10240'
  Tunnel source address is '20.0.0.2'
  Tunnel destination address is '20.0.0.1'
  Tunnel transport virtual router is vr1
  Tunnel checksum option is disabled
  Tunnel sequence number option is enabled (628 tx / 628 rx)
  Tunnel up/down trap is enabled
  Tunnel-server location is 3/0
  Tunnel administrative state is Up
Statistics      packets      octets      discards      errors
  Data rx      627          97381        0             0
  Data tx      628          97381        1             0
2 GRE tunnels found

```

