ERICSSON

Configuring Single Circuit Tunnels

SYSTEM ADMINISTRATOR GUIDE

69/1543-CRA 119 1170/1-V1 Uen D

Contents

1 Overview

This document shows how to configure, monitor, and administer single-circuit tunnel types which includes IP-in-IP, overlay, and single-circuit Generic Routing Encapsulation (GRE) tunnels.

This document applies to both the Ericsson SmartEdge® and SM family routers. However, the software that applies to the SM family of systems is a subset of the SmartEdge OS; some of the functionality described in this document may not apply to SM family routers.

For information specific to the SM family chassis, including line cards, refer to the SM family chassis documentation.

For specific information about the differences between the SmartEdge and SM family routers, refer to the Technical Product Description *SM Family of Systems* (part number 5/221 02-CRA 119 1170/1) in the **`Product Overview`** folder of this Customer Product Information library.

1.1 Terminology

The term tunnel refers to the tunnel encapsulation and its circuit.

Note: When IPv6 addresses are not referenced or explicitly specified, the term IP address can refer generally to IP Version 4 (IPv4) addresses, IPv6 addresses, or IP addressing. In instances where IPv6 addresses are referenced or explicitly specified, the term IP address refers only to IPv4 addresses. For a description of IPv6 addressing and the types of IPv6 addresses, see RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*.

1.2 GRE Tunnels

GRE is a simple, stateless protocol that allows for the tunneling of IP in IP. The SmartEdge router implementation of GRE over IPv4 is based on these IETF documents:

- RFC 1702, *Generic Routing Encapsulation over IPv4 Networks*
- RFC 2784, *Generic Routing Encapsulation*
- RFC 2893, *Transition Mechanisms for IPv6 Hosts and Routers*

1.2.1 Using GRE Tunnels with IPv6 Packets

GRE tunnels allow you to connect remote sites using IPv6 addresses over a public network that uses publicly routable IPv4 addresses. IPv6 packets traveling through the tunnel are encapsulated with a GRE header and then with an IPv4 header using addresses from the public IPv4 address as shown in Figure 1.

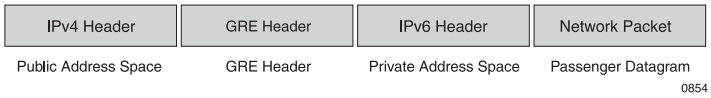

Figure 1 GRE Tunnel Packet Encapsulation for IPv6 Packets

1.2.2 Using GRE Tunnels with IPv4 Packets

GRE tunnels also allow you to connect remote sites using private IP addresses over a public network that uses publicly routable IP addresses. IP packets traveling through the tunnel are encapsulated with an IP header from the public address space as shown in Figure 2.

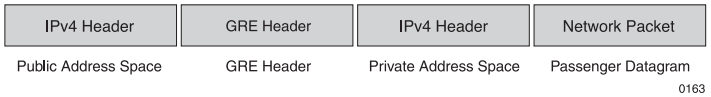

Figure 2 GRE Tunnel Packet Encapsulation for IPv4 Packets

1.3 IP-in-IP Tunnels

IP-in-IP tunnels transport IP payload packets encapsulated inside an outer IP header. This tunneling mode has been introduced to support Mobile IP services, which use IP-in-IP tunnels to transport packets from a mobile node (MN) when the packets are forwarded from the foreign agent (FA) to the home agent (HA) and, optionally, in the reverse direction.

1.4 Overlay Tunnels

Overlay tunnels encapsulate IPv6 packets in IPv4 packets for delivery across an IPv4 infrastructure (a core network or the Internet). By using overlay tunnels, you can communicate with isolated IPv6 networks without upgrading the IPv4 infrastructure between them.

Note: Overlay tunnels reduce the maximum transmission unit (MTU) by 20 octets (assuming the basic IPv4 packet header does not contain optional fields). A network using overlay tunnels is difficult to troubleshoot. Therefore, overlay tunnels connecting isolated IPv6 networks should not be considered as 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.

An overlay tunnel is used within a site or between sites; it 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 routers or between an end system and an edge router, or for connection to remote IPv6 networks. You can configure overlay tunnels between border routers or between a border router and a host. The host or router at each end of a tunnel must support both the IPv4 and IPv6 protocol stacks.

The SmartEdge router implementation of overlay tunnels is based on the RFC 2893, `Transition Mechanisms for IPv6 Hosts and Routers`. Figure 3 displays encapsulated IPv6 packets traveling through the tunnel.

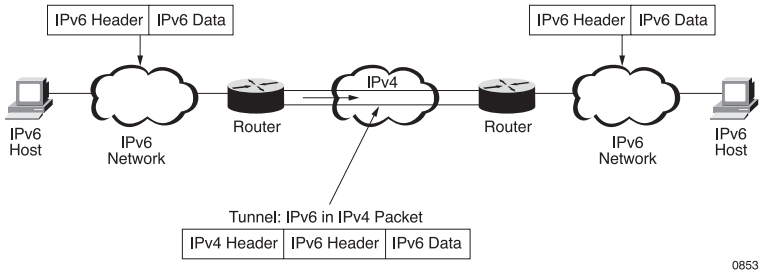

Figure 3 IPv6 Tunnel Packet Encapsulation

1.5 Related Information

For information about GRE tunnels that support multiple tunnel circuits, see *Configuring GRE Tunnels*.

2 Configuration and Operations Tasks

To configure tunnels and perform tunnel and tunnel circuit operations, perform the tasks in the following sections.

Note: Unless otherwise noted, the SmartEdge 100 router supports all commands described in this document.

2.1 Configure a Tunnel of any Type

To configure a tunnel of any type, perform the tasks described in Table 1.

Table 1 Configure a Tunnel of Any Type

Step	Task	Root Command	Notes
1.	Create or select the context for the tunnel interface and access context configuration mode.	*context*	Enter this command in global configuration mode.
2.	Create or select the local tunnel endpoint and access interface configuration mode. This interface becomes the local tunnel endpoint only when, in tunnel configuration mode (step 6), it is bound.	*Interface (context)*	Enter this command in the context mode created or selected in the previous step.
3.	Assign an IP address to the interface.	*ip address (interface)*	Enter this command in interface mode. This is an IPv4 address.
4.	Create the tunnel, specify its type, and access tunnel configuration mode.	*tunnel*	Enter this command in global configuration mode. An alternative method of configuring GRE tunnels is described in *Configuring GRE Tunnels*.
5.	Assign IP addresses to both the local and remote tunnel endpoints. For the local tunnel endpoint, enter the IP address configured in step 3.	*peer-end-point*	Enter this command in tunnel configuration mode. You must assign IP addresses to both the local and remote tunnel endpoints unless you are creating an overlay tunnel with a system-generated remote address.
6.	Bind the tunnel to the interface you created in step 2.	*bind interface*	Enter this command in tunnel configuration mode.

Table 1 Configure a Tunnel of Any Type

Step	Task	Root Command	Notes
7.	Specify optional tunnel attributes. Enter the following commands in tunnel configuration mode:		
	Clear the IP header DF flag.	*clear-df*	Not available for overlay tunnels.
	Associate a description with the tunnel.	*description (tunnel)*	
	Enable the sending of keepalive packets.	*keepalive (tunnel)*	This command is available only for GRE tunnels. **Note:** Do not configure your SmartEdge router router so that keepalive packets might be forwarded and received on a tunnel endpoint that is bound to a different context.
	Enable the logging of state changes.	*log-state-changes*	
	Set the MTU for the tunnel.	*mtu (tunnel)*	
8.	Enable the tunnel (begin operations on it).	*shutdown (Tunnel)*	Use the **no** form of this command to enable the tunnel.

Note: If the traffic receiving end of a GRE tunnel is a SmartEdge router, you must configure the router with a GRE tunnel whose local endpoint is the traffic receiving endpoint of the tunnel. This configuration is required in order to respond to GRE keepalives sent by the traffic sending end of the tunnel.

2.2 Tunnel and Tunnel Circuit Operations

To monitor and troubleshoot GRE tunnels and tunnel circuits, perform the appropriate task listed in Table 2. Enter the **clear** and **debug** commands in exec mode; enter the **show** commands in any mode.

Table 2 Tunnel and Tunnel Circuit Operations

Task	Root Command
Clear the circuit counters for one or more GRE tunnel circuits in the system.	*clear circuit counters*
Enable the generation of debug messages for circuit events.	*debug circuit*
Enable the generation of debug messages for dynamic tunnel client events.	*debug tunnel client*
Display circuit information for one or more tunnel circuits in the system.	*show circuit*

Table 2 Tunnel and Tunnel Circuit Operations

Task	**Root Command**
Display general counters and counters specific to the GRE tunnel circuit type for one or more GRE tunnel circuits in the system.	*show circuit counters*
Display configuration commands for a GRE tunnel circuit.	*show configuration (circuits)*
Display GRE and IP-IP tunnel, or tunnel circuit information, or the IP Version 6 (IPv6) tunnel.	*show tunnel*
Display information about dynamic tunnel clients that are registered with the tunnel manager.	*show tunnel client*
Display the bindings for GRE tunnels and the interfaces to which they are bound.	*show ip interface*

3 Configuration Examples

An example configuration for each type of tunnel is provided in the following sections:

- GRE Tunnel
- IP-in-IP Tunnel
- Overlay Tunnel

3.1 GRE Tunnel

The following example shows how to configure a GRE tunnel to **Hartford** on the SmartEdge router in **New York:**

```
!Create the interface for the GRE tunnel in the local context
[local]NewYork(config)#context local
[local]NewYork(config-ctx)#interface toHartford

!Assign a public IP address to the local tunnel interface
[local]NewYork(config-if)#ip address 172.17.1.1/30
[local]NewYork(config-if)#exit
[local]NewYork(config-ctx)#exit

!Configure the tunnel with the local IP address of its interface
[local]NewYork(config)#tunnel gre HartfordTnl
[local]NewYork(config-tunnel)#peer-end-point local 172.17.1.1 remote 172.17.1.2
[local]NewYork(config-tunnel)#description Hartford tunnel
[local]NewYork(config-tunnel)#no shutdown
```

3.2 IP-in-IP Tunnel

The following example shows how to configure an IP-in-IP tunnel, **HaTnl**, with its interface **toHA** in the **local** context:

```
! Create the interface for the IP-in-IP tunnel interface
[local]Redback(config)#context local
[local]Redback(config-ctx)#interface toHA
[local]Redback(config-if)#ip address 20.2.1.1/16
[local]Redback(config-if)#exit

!Configure the IP-in-IP tunnel (bind it to the tunnel interface in the local context)
[local]Redback(config)#tunnel ipip HaTnl
[local]Redback(config-tunnel)#peer-end-point local 20.1.1.1 remote 20.1.1.2
[local]Redback(config-tunnel)#bind interface toHa local
[local]Redback(config-tunnel)#end
```

3.3 Overlay Tunnel

The following example shows how to configure an overlay tunnel, **DenverTnl**, with its interface, **toDenver**, and the interface for its IPv6 tunnel, **IPv6-circuit**, in the **local** context:

```
!Create the IPv4 interface for the tunnel in the local context
[local]Redback(config)#context local
[local]Redback(config-ctx)#interface toDenver

!Assign an IPv4 public IP address to the local tunnel interface
[local]Redback(config-if)#ip address 172.16.1.1/30
[local]Redback(config-if)#exit

!Create the interface for the tunnel
[local]Redback(config-ctx)#interface IPv6-circuit
!Assign an IPv6 public address to the interface for the tunnel
[local]Redback(config-if)#ipv6 7001::1/64
[local]Redback(config-if)#exit
[local]Redback(config-ctx)#exit

!Create the tunnel with IPv4 addresses for its endpoints
!The local end uses the IPv4 address of the tunnel's interface.
[local]Redback(config)#tunnel ipv6v4-manual DenverTnl
[local]Redback(config-tunnel)#peer-end-point local 172.16.1.1 remote 172.16.1.2
[local]Redback(config-tunnel)#log-state-changes
[local]Redback(config-tunnel)#description IPv6 over IPv4 tunnel with a single circuit
[local]Redback(config-tunnel)#mtu 1024

!Bind the tunnel to the interface with the IPv6 address, in the local context
[local]Redback(config-tunnel)#bind interface IPv6-circuit local
[local]Redback(config-tunnel)#end
[local]Redback(config-tunnel)#no shutdown
[local]Redback(config-tunnel)#exit
```