VYATTA, INC. | Vyatta System

Tunnels

REFERENCE GUIDE

GRE Tunnels
IP-in-IP Tunnels
SIT Tunnels

Vyatta
Suite 200
1301 Shoreway Road
Belmont, CA 94002
vyatta.com
650 413 7200
1 888 VYATTA 1 (US and Canada)

COPYRIGHT

PROPRIETARY NOTICES

RELEASE DATE: October 2012

DOCUMENT REVISION. 6.5R1 v01

RELEASED WITH: 6.5R1

PART NO. A0-0229-10-0013

Contents

Quick Reference to Commands

Use this section to help you quickly locate a command.

Quick List of Examples

Use this list to help you locate examples you'd like to try or look at.

Preface

This document describes the various deployment, installation, and upgrade options for Vyatta software.

This preface provides information about using this guide. The following topics are presented:

Intended Audience

This guide is intended for experienced system and network administrators. Depending on the functionality to be used, readers should have specific knowledge in the following areas:

- Networking and data communications
- TCP/IP protocols
- General router configuration
- Routing protocols
- Network administration
- Network security
- IP services

Organization of This Guide

This guide has the following aid to help you find the information you are looking for:

- Quick Reference to Commands

  Use this list to help you quickly locate commands.

- Quick List of Examples

  Use this list to help you locate examples you'd like to try or look at.

This guide has the following chapters:

Chapter	Description	Page
Chapter 1: Tunneling Overview	This chapter gives a brief overview of tunneling support on the Vyatta system.	1
Chapter 2: Tunnel Configuration Examples	This chapter provides tunnel configuration examples.	6
Chapter 3: Tunnel Commands	This chapter lists the commands for configuring GRE and IP-in-IP tunnels.	17
Glossary of Acronyms		51

Document Conventions

This guide uses the following advisory paragraphs, as follows.

WARNING *Warnings alert you to situations that may pose a threat to personal safety.*

CAUTION *Cautions alert you to situations that might cause harm to your system or damage to equipment, or that may affect service.*

NOTE *Notes provide information you might need to avoid problems or configuration errors.*

This document uses the following typographic conventions.

`Monospace`	Examples, command-line output, and representations of configuration nodes.
`bold Monospace`	Your input: something you type at a command line.
bold	Commands, keywords, and file names, when mentioned inline. Objects in the user interface, such as tabs, buttons, screens, and panes.
italics	An argument or variable where you supply a value.
<key>	A key on your keyboard, such as <Enter>. Combinations of keys are joined by plus signs (“+”), as in <Ctrl>+c.
[key1 | key2]	Enumerated options for completing a syntax. An example is [enable | disable].
num1–numN	A inclusive range of numbers. An example is 1–65535, which means 1 through 65535, inclusive.
arg1..argN	A range of enumerated values. An example is eth0..eth3, which means eth0, eth1, eth2, or eth3.
arg[*arg*...] *arg*[,*arg*...]	A value that can optionally represent a list of elements (a space-separated list and a comma-separated list, respectively).

Vyatta Publications

Full product documentation is provided in the Vyatta technical library. To see what documentation is available for your release, see the *Guide to Vyatta Documentation*. This guide is posted with every release of Vyatta software and provides a great starting point for finding the information you need.

Additional information is available on www.vyatta.com and www.vyatta.org.

Chapter 1: Tunneling Overview

This chapter gives a brief overview of tunneling support on the Vyatta system.

This chapter presents the following topics:

Overview

An IP tunneling protocol is a mechanism for encapsulating packets from one network protocol into a packet from another protocol, creating a "tunnel." The transported protocol (the "passenger" protocol) is encapsulated by wrapping around it packet information for the tunneling protocol (the "carrier" protocol). The encapsulated packet is then forwarded to some destination and stripped of the encapsulating information, and the original packet is delivered.

The Vyatta system supports three commonly used tunneling protocols:

- Generic Routing Encapsulation (GRE) tunnels can be used to carry non-IP protocols such as Novell IPX, Banyan VINES, AppleTalk, and DECNet. They can also be used to carry multicast, broadcast, or IPv6 traffic. A variant, GRE for Bridging, is used to provide GRE tunnels that can be bridged.
- IP-in-IP tunnels can only be used to carry IPv4 traffic.
- Simple Internet Transition (SIT) tunnels can be used to transport IPv6 packets over IPv4 routing infrastructures.

Logical interfaces that send IP packets in a tunneled mode are called tunnel interfaces. Tunnel interfaces behave just like any other system interface: you can configure routing protocols, firewall, NAT, and other features on them, and you can manage them using standard operational tools and commands.

Note that GRE, IP-in-IP, and SIT tunnels are unencrypted.

GRE Tunnels

The GRE protocol provides a simple-general purpose mechanism for encapsulating packets from a wide variety of network protocols to be forwarded over another protocol. The original packet (the "passenger" packet) can be one of many arbitrary network protocols—for example a multicast packet, an IPv6 packet, or a non-IP LAN protocol such as AppleTalk, Banyan VINES, or Novell IPX. The delivery protocol can be one of a number of routable IP protocols.

The passenger packet is first encapsulated within a GRE packet, creating the GRE "tunnel." The GRE packet is then encapsulated itself within a delivery protocol such as OSPF or IPsec and forwarded to the remote destination.

You might use GRE if you want to:

- Connect networks running non-IP protocols, such as native LAN protocols, across the public IP network. Non-IP protocols such as Novell IPX or Appletalk are not routable across an IP network. A GRE tunnel allows you to create a virtual point-to-point link between two such networks over the public WAN.
- Route IPv6 packets across an IPv4 network, or connect any two similar networks across an infrastructure that uses different IP addressing.

- Encrypt multicast traffic. IPsec, which is a standard mechanism for providing security on IP networks, cannot encrypt multicast packets. However, multicast packets can be encapsulated within a GRE tunnel and then routed over a VPN connection, so that the encapsulated packets are protected by the IPsec tunnel.

GRE tunnels are stateless, which means that the protocol does not automatically monitor the state or availability of other endpoints. You can, however, direct the router to monitor the far end of the tunnel by sending keep-alive messages. If the other end of the tunnel becomes unavailable, its failure to respond to the messages will alert the router.

GRE provides no security other than a key that can be configured on each side of the tunnel. This key is carried in each packet in clear text, which means that GRE is not secure. If security is required, GRE can be used in conjunction with IPsec.

GRE uses IP protocol number 47.

GRE for Bridging Tunnels

One of the limitations of regular GRE-encapsulated tunnels is that the resulting tunnels cannot be added to a bridge group. GRE for bridging provides this ability. GRE for bridging should only be used in cases where tunnel interfaces are to be included in a bridge group.

To configure GRE for bridging, use the **gre-bridge** option of the interfaces tunnel <tunx> encapsulation command. For more information about GRE for bridging, see the *Vyatta Bridging Reference Guide*.

IP-in-IP Tunnels

The IP-in-IP encapsulation protocol is used to tunnel between networks that have different capabilities or policies. For example, an IP-in-IP tunnel can be used to forward multicast packets across a section of a network (such as an IPsec tunnel) that does not support multicast routing. An IP-in-IP tunnel can also be used to influence the routing of the packet, or to deliver a packet to a mobile device using Mobile IP.

In IP-in-IP encapsulation, a second IP header is inserted in front of the IP header of the original packet (the “passenger” packet). The new IP header has as source and destination addresses the addresses of the tunnel endpoints. The IP header of the payload packet identifies the original sender and receiver. When the encapsulated packet exits the tunnel, the outer IP header is stripped off, and the original IP packet is delivered to the final destination.

IP-in-IP encapsulation is simple and robust. It is useful for connecting IPv4 networks that otherwise would not be able to communicate; however, it has some limitations:

- IP-in-IP encapsulation does not support broadcast traffic
- IP-in-IP encapsulation does not support IPv6 traffic

For forwarding this kind of traffic, GRE may be more appropriate.

Like GRE, IP-in-IP has only the most basic security: a password-like key. This key is carried in each packet in clear text, which means that, on their own, IP-in-IP tunnels are not secure. For secure communications, IP-in-IP tunnels can be used together with IPsec.

SIT Protocol

The Simple Internet Transition (SIT) protocol was designed to provide mechanisms for transitioning networks from IPv4 to IPv6.

The mechanisms employed by SIT include:

- Use of the dual–IP layer (IPv4 and IPv6) technique in hosts and routers for direct interoperability with nodes implementing both protocols.
- Two IPv6 addressing structures that embed an IPv4 addresses within IPv6 addresses.
- A mechanism for tunneling IPv6 packets over IPv4 routing infrastructures. This technique uses the embedded IPv4 address structure, which eliminates the need for tunnel configuration in most cases.
- An optional mechanism for translating headers of IPv4 packets into IPv6, and the headers of IPv6 packet into IPv4. This technique allows nodes that implement only IPv6 to interoperate with nodes that implement only IPv4.

Tunnel Interfaces and IPsec

GRE, IP-in-IP, and SIT tunnels are not encrypted, using a simple password-like key that is exchanged in clear text in each packet. At the same time, IPsec policy-based tunnels cannot directly route non-IP or multicast protocols, and IPsec also has limitations from an operations point of view. Using tunnel interfaces in conjunction with IPsec VPN provides secure, routable tunnel connections between gateways, that have some advantages over traditional IPsec policy-based tunnel mode connections:

- Support for standard operational commands such as **show interfaces** and **show route**
- Support for operational tools such as **traceroute** and SNMP
- Dynamic tunnel failover using routing protocols
- Simplified IPsec policies and troubleshooting

For secure routable tunnels, GRE, IP-in-IP, and SIT tunnel interfaces can be used in conjunction with an IPsec connection, so that the IP tunnel can be protected by the IPsec tunnel.

IPsec is explained in detail in the *Vyatta VPN Reference Guide*. Please see that guide for more information.

Supported Standards

The Vyatta implementation of GRE complies with the following standards:

- RFC 1702: Generic Routing Encapsulation over IPv4 Networks
- RFC 2784: Generic Routing Encapsulation

The Vyatta implementation of IP-in-IP complies with the following standard:

- RFC 1853: IP in IP Tunneling

The use of tunnel interfaces with IPsec is documented in the following standard, which describes the use of IP-in-IP tunnels combined with IPsec transport mode encryption to provide secure routable tunnels:

- RFC 3884: Use of IPsec Transport Mode for Dynamic Routing

The Vyatta implementation of SIT complies with the following standard:

- RFC 4213: Basic Transition Mechanisms for IPv6 Hosts and Routers

Chapter 2: Tunnel Configuration Examples

This chapter provides tunnel configuration examples.

This chapter presents the following topics:

Before You Begin

- In the GRE examples that follow, we assume that you have two systems, with host names configured WEST and EAST. (The example systems are configured with the host name in upper case.)
- Any Ethernet interface to be used for tunnel modes must already be configured. In this example, you will need eth1 on WEST and eth0 on EAST, plus internal subnet information.
- The Loopback interface is used a the tunnel endpoint.

Please see *Vyatta LAN Interfaces Reference Guide* for information on configuring Ethernet interfaces, Loopback interfaces, and IP addresses.

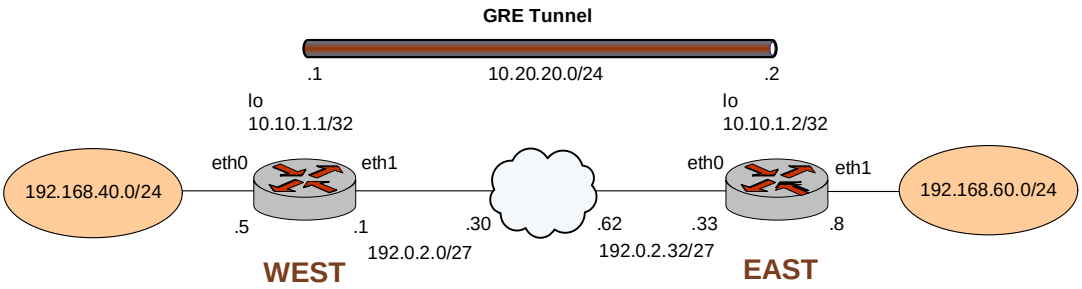
A Basic GRE Tunnel

This section presents a sample configuration for a basic GRE tunnel between Vyatta systems WEST and EAST. First WEST is configured, and then EAST.

This basic tunnel is not protected by a key: this means it is not secure, and would not be suitable for a production network.

When you have finished, these systems will be configured as shown in Figure 2-1.

Figure 2-1 Basic GRE tunnel



Configure WEST

The GRE tunnel in the example configuration extends from WEST through the wide-area network to EAST. In this example, you create the tunnel interface and the tunnel endpoint on WEST.

- The tunnel interface tun0 on WEST is assigned the IP address 10.20.20.1 on network 10.20.20.0/24.

- The source IP address of the tunnel endpoint (the **local-ip**) is the same as the address associated with the Loopback interface (lo) in this example.
- The IP address of the other end of the tunnel (the **remote-ip**) is the address of the Loopback interface on EAST.

Example 2-1 creates the tunnel interface and the tunnel endpoint on WEST. To do this, perform the following steps on WEST in configuration mode.

Example 2-1 Creating a basic GRE tunnel endpoint on WEST

Step	Command
Create the tunnel interface, and specify the IP address to be associated with it.	`vyatta@WEST# set interfaces tunnel tun0 address 10.20.20.1/24`
Specify the source IP address for the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 local-ip 10.10.1.1`
Specify the IP address of the other end of the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 remote-ip 10.10.1.2`
Specify the encapsulation mode for the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 encapsulation gre`
Assign a brief description for the tunnel interface.	`vyatta@WEST# set interfaces tunnel tun0 description “GRE tunnel to EAST”`
Commit the configuration.	`vyatta@WEST# commit`

View the configuration.	`vyatta@WEST# show interfaces tunnel tun0`
	`  address 10.20.20.1/24`
	`  description “Tunnel to EAST”`
	`  encapsulation gre`
	`  local-ip 10.10.1.1`
	`  remote-ip 10.10.1.2`

Create a static route to access the remote subnet via the tunnel.	`vyatta@WEST# set protocols static route 192.168.60.0/24 next-hop 10.20.20.2`
Commit the configuration.	`vyatta@WEST# commit`

View the configuration.	`vyatta@WEST# show protocols`
	`  static {`
	`  route 192.168.60.0/24 {`
	`  next-hop 10.20.20.2 {`
	`  }`
	`  }`
	`  }`

Configure EAST

In this example, you create the tunnel endpoint on EAST.

- The tunnel interface tun0 on EAST is assigned the IP address 10.20.20.2 on network 10.20.20.0/24.
- The source IP address of the tunnel endpoint (the **local-ip**) is the same as the address associated with the Loopback interface (lo) in this example.
- The IP address of the other end of the tunnel (the **remote-ip**) is the address of the Loopback interface on WEST.
- A static route is created to specify how to get to the remote LAN via the tunnel.

Example 2-2 creates the tunnel endpoint on EAST. To do this, perform the following steps on EAST in configuration mode.

Example 2-2 Creating a basic GRE tunnel endpoint on EAST

Step	Command
Create the tunnel interface, and specify the IP address to be associated with it.	`vyatta@WEST# set interfaces tunnel tun0 address 10.20.20.2/24`
Specify the source IP address for the tunnel.	`vyatta@EAST# set interfaces tunnel tun0 local-ip 10.10.1.2`
Specify the IP address of the other end of the tunnel.	`vyatta@EAST# set interfaces tunnel tun0 remote-ip 10.10.1.1`
Specify the encapsulation mode for the tunnel.	`vyatta@EAST# set interfaces tunnel tun0 encapsulation gre`
Assign a brief description for the tunnel interface.	`vyatta@EAST# set interfaces tunnel tun0 description “GRE tunnel to WEST”`
Commit the configuration.	`vyatta@EAST# commit`

View the configuration.	`vyatta@EAST# show interfaces tunnel tun0`
	`address 10.20.20.2/24`
	`description “Tunnel to WEST”`
	`encapsulation gre`
	`local-ip 10.10.1.2`
	`remote-ip 10.10.1.1`

Create a static route to access the remote subnet via the tunnel.	`vyatta@WEST# set protocols static route 192.168.40.0/24 next-hop 10.20.20.1`
Commit the configuration.	`vyatta@WEST# commit`

Example 2-2 Creating a basic GRE tunnel endpoint on EAST

View the configuration.

```
vyatta@WEST# show protocols
   static {
       route 192.168.40.0/24 {
           next-hop 10.20.20.1 {
           }
       }
   }
```

A More Complex GRE Tunnel

In this section, some additional parameters are specified for the tunnel interfaces defined in the previous section.

- A key is specified so that the hosts can authenticate one another. This key must match on the two endpoints.
- Time-to-live, ToS, and MTU values are specified for each endpoint.
- A firewall rule set is applied to each tunnel interface.

Configure WEST

Example 2-3 specifies additional values for the tunnel endpoint on WEST created in Example 2-1:

- A key 101088 is provide as a password-like mechanism. These values must match on each side.
- The time-to-live for packets is set to 220, the ToS field is set to 55, and MTU for packets is set to 1460.
- Two firewall rules set are applied to the tunnel interface:
  - The rule set tun0-fw-in is applied to packets ingressing through the tunnel interface.
  - The rule set tun0-fw-out is applied to packets egressing through the tunnel interface.

  (This example assumes that these firewall rule sets have already been defined. For information on defining firewall rule sets, please see the *Vyatta Firewall Reference Guide*.)

To configure the GRE tunnel endpoint, perform the following steps on WEST in configuration mode.

Example 2-3 Adding values to the GRE tunnel endpoint on WEST

Step	Command
Provide the authentication key	`vyatta@WEST# set interfaces tunnel tun0 key 101088`
Set the time-to-live.	`vyatta@WEST# set interfaces tunnel tun0 ttl 220`
Set the Type of Service.	`vyatta@WEST# set interfaces tunnel tun0 tos 55`
Set the MTU.	`vyatta@WEST# set interfaces tunnel tun0 mtu 1460`
Apply the firewall rule set for incoming packets.	`vyatta@WEST# set interfaces tunnel tun0 firewall in name tun0-fw-in`
Apply the firewall rule set for outgoing packets.	`vyatta@WEST# set interfaces tunnel tun0 firewall out name tun0-fw-out`
Commit the configuration.	`vyatta@WEST# commit`

View the configuration.	`vyatta@WEST# show interfaces tunnel tun0`
	`address 10.20.20.1/24`
	`description “Tunnel to EAST”`
	`encapsulation gre`
	`firewall`
	`in {`
	`name tun0-fw-in`
	`}`
	`out {`
	`name tun0-fw-out`
	`}`
	`}`
	`key 101088`
	`local-ip 10.10.1.1`
	`mtu 1460`
	`remote-ip 10.10.1.2`
	`tos 55`
	`ttl 220`

Configure EAST

Example 2-4 specifies additional values for the tunnel endpoint on EAST created in Example 2-2:

- A key 101088 is provide as a password-like mechanism. This value matches the key configured for WEST.

- The time-to-live for packets is set to 220, the ToS field is set to 55, and MTU for packets is set to 1460.
- Two firewall rules set are applied to the tunnel interface:
  - The rule set tun0-fw-in is applied to packets ingressing through the tunnel interface.
  - The rule set tun0-fw-out is applied to packets egressing through the tunnel interface.

  (This example assumes that these firewall rule sets have already been defined. For information on defining firewall rule sets, please see the *Vyatta Firewall Reference Guide*.)

To do this, perform the following steps on EAST in configuration mode.

Example 2-4 Adding values to the GRE tunnel endpoint on EAST

Step	Command
Provide the authentication key	`vyatta@EAST# set interfaces tunnel tun0 key 101088`
Set the time-to-live.	`vyatta@EAST# set interfaces tunnel tun0 ttl 220`
Set the Type of Service.	`vyatta@EAST# set interfaces tunnel tun0 tos 55`
Set the MTU.	`vyatta@EAST# set interfaces tunnel tun0 mtu 1460`
Apply the firewall rule set for incoming packets.	`vyatta@EAST# set interfaces tunnel tun0 firewall in name tun0-fw-in`
Apply the firewall rule set for outgoing packets.	`vyatta@EAST# set interfaces tunnel tun0 firewall out name tun0-fw-out`
Commit the configuration.	`vyatta@EAST# commit`

Example 2-4 Adding values to the GRE tunnel endpoint on EAST

View the configuration.

```
vyatta@EAST# show interfaces tunnel tun0
    address 10.20.20.2/24
    description “Tunnel to WEST”
    encapsulation gre
    firewall
        in {
            name tun0-fw-in
        }
        out {
            name tun0-fw-out
        }
    }
    key 101088
    local-ip 10.10.1.2
    mtu 1460
    remote-ip 10.10.1.1
    tos 55
    ttl 220
```

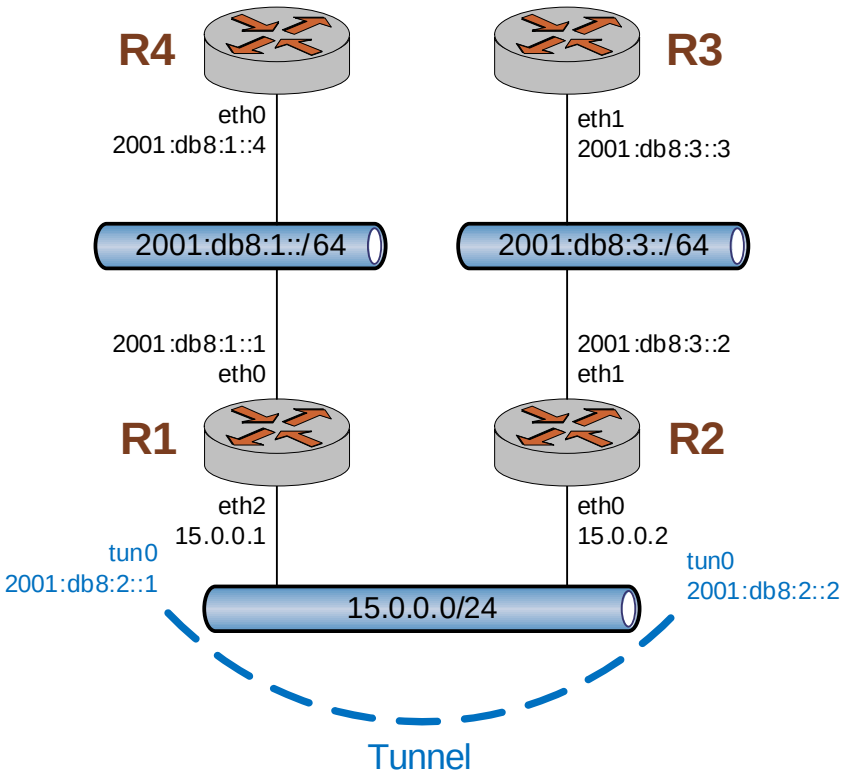
Bridging with GRE Tunnels

GRE Tunnels can be used to bridge LAN segments across a WAN. For more information, see the *Vyatta Bridging Reference Guide*.

Tunneling IPv6 Traffic in IPv4

Figure 2-2 shows a network with four nodes. R1 and R2 each have an interface configured as IPv6 and an interface configured as IPv4. In this example we will show configuration of the nodes using tunneling over IPv4 to enable R3 and R4 to communicate via R1 and R2.

Figure 2-2 Tunneling IPv6 traffic in IPv4 example



We assume that all interfaces have been configured with IP addresses as shown in the example. We will also assume that R1 and R2 have forwarding enabled.

Create SIT tunnel

R1 and R2 must be configured to create a tunnel between them in order to encapsulate the IPv6 traffic. To configure R1 to create a tunnel using SIT (Simple Internet Transition) encapsulation between 15.0.0.1 and 15.0.0.2, perform the following steps in configuration mode.

Example 2-5 Configure tunnel interface on R1

Step	Command
Create a tunnel with SIT encapsulation.	`vyatta@R1# set interfaces tunnel tun0 encapsulation sit`
Specify the local IP address.	`vyatta@R1# set interfaces tunnel tun0 local-ip 15.0.0.1`
Specify the remote IP address.	`vyatta@R1# set interfaces tunnel tun0 remote-ip 15.0.0.2`
Configure the IPv6 address on the interface.	`vyatta@R1# set interfaces tunnel tun0 address 2001:db8:2::1/64`

Example 2-5 Configure tunnel interface on R1

Commit the change.	`vyatta@R1# commit`

To configure R2 to create a tunnel using SIT (Simple Internet Transition) encapsulation between 15.0.0.2 and 15.0.0.1, perform the following steps in configuration mode.

Example 2-6 Configure tunnel interface on R2

Step	Command
Create a tunnel with SIT encapsulation.	`vyatta@R2# set interfaces tunnel tun0 encapsulation sit`
Specify the local IP address.	`vyatta@R2# set interfaces tunnel tun0 local-ip 15.0.0.2`
Specify the remote IP address.	`vyatta@R2# set interfaces tunnel tun0 remote-ip 15.0.0.1`
Configure the IPv6 address on the interface.	`vyatta@R2# set interfaces tunnel tun0 address 2001:db8:2::2/64`
Commit the change.	`vyatta@R2# commit`

At this point there is connectivity between R1 and R2 across the tunnel interface. The following shows a capture of a ping from 2001:db8:2::1 to 2001:db8:2::2. Notice that the IPv6 ping packet in encapsulated by the IPv4 header:

Example 2-7 Capture of ping

```
Frame 22 (138 bytes on wire, 138 bytes captured)
Ethernet II, Src: Vmware_d6:81:80 (00:0c:29:d6:81:80), Dst: Vmware_4e:fc:b6
(00:0c:29:4e:fc:b6)
    Destination: Vmware_4e:fc:b6 (00:0c:29:4e:fc:b6)
    Source: Vmware_d6:81:80 (00:0c:29:d6:81:80)
    Type: IP (0x0800)
Internet Protocol, Src: 15.0.0.1 (15.0.0.1), Dst: 15.0.0.2 (15.0.0.2)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 124
    Identification: 0x0000 (0)
    Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 255
    Protocol: IPv6 (0x29)
    Header checksum: 0x5d56 [correct]
    Source: 15.0.0.1 (15.0.0.1)
    Destination: 15.0.0.2 (15.0.0.2)
```

```
Internet Protocol Version 6
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 64
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: 2001:db8:2::1 (2001:db8:2::1)
    Destination: 2001:db8:2::2 (2001:db8:2::2)
Internet Control Message Protocol v6
    Type: 129 (Echo reply)
    Code: 0
    Checksum: 0x2fca [correct]
    ID: 0xe825
    Sequence: 0x001b
    Data (56 bytes)

0000  9b a8 25 49 58 0c 07 00 08 09 0a 0b 0c 0d 0e 0f   ..%IX...........
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37                          01234567
```

Chapter 3: Tunnel Commands

This chapter lists the commands for configuring GRE and IP-in-IP tunnels.

This chapter contains the following commands.

Configuration Commands	
interfaces tunnel <tunx>	Defines a tunnel interface.
interfaces tunnel <tunx> address <ipv4net>	Sets a primary or secondary IP address for a tunnel interface.
interfaces tunnel <tunx> description <descr>	Specifies a description for a tunnel interface.
interfaces tunnel <tunx> disable	Disables a tunnel interface without discarding configuration.
interfaces tunnel <tunx> encapsulation	Sets the encapsulation for a tunnel interface.
interfaces tunnel <tunx> local-ip <ipv4>	Sets the IP address for the local endpoint of a tunnel.
interfaces tunnel <tunx> mtu <mtu>	Sets the MTU size for a tunnel interface.
interfaces tunnel <tunx> parameters ip key <key>	Defines an authentication key for a tunnel interface.
interfaces tunnel <tunx> parameters ip tos <tos>	Specifies the value to be written into the ToS byte of the tunnel packet's IP header.
interfaces tunnel <tunx> parameters ip ttl <ttl>	Defines the time-to-live (TTL) value to be written into the tunnel packet's IP header.
interfaces tunnel <tunx> parameters ipv6 encaplimit <limit>	Defines the maximum number of times the tunnel packet can be encapsulated.
interfaces tunnel <tunx> parameters ipv6 flowlabel <flowlabel>	Defines the flowlabel of the encapsulating ipv6 header.
interfaces tunnel <tunx> parameters ipv6 hoplimit <limit>	Defines the hoplimit value to be written into the tunnel packet's IPv6 header.
interfaces tunnel <tunx> parameters ipv6 tclass <class>	Specifies the value to be written into the tclass byte of the tunnel packet's IPv6 header.
Operational Commands	
clear interfaces tunnel counters	Clears tunnel interface statistics.
show interfaces tunnel	Displays information about tunnel interfaces.

Commands for using other system features with tunnel interfaces can be found in the following locations.

Related Commands Documented Elsewhere	
Firewall	Commands for configuring firewall on tunnel interfaces are described in the *Vyatta Firewall Reference Guide*.
OSPF	Commands for configuring the Open Shortest Path First routing protocol on tunnel interfaces are described in the *Vyatta OSPF Reference Guide*.
RIP	Commands for configuring the Routing Information Protocol on tunnel interfaces are described in the *Vyatta RIP Reference Guide*.
Ripng	Commands for configuring the Routing Information Protocol next generation (RIPng) on tunnel interfaces are described in the *Vyatta RIPng Reference Guide*.

clear interfaces tunnel counters

Clears tunnel interface statistics.

Syntax

clear interfaces tunnel [*tunx*] **counters**

Command Mode

Operational mode.

Parameters

tunx	Optional. Clears information for the specified tunnel interface. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.

Default

None.

Usage Guidelines

Use this command to clear statistics for tunnel interfaces.

interfaces tunnel <tunx>

Defines a tunnel interface.

Syntax

set interfaces tunnel *tunx*

delete interfaces tunnel [*tunx*]

show interfaces tunnel [*tunx*]

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
    tunnel tunx {
    }
}
```

Parameters

tunx	Mandatory. An identifier for the tunnel interface you are defining. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.

Default

None.

Usage Guidelines

Use this command to create a tunnel interface for encapsulating traffic.

Use the **set** form of this command to create a tunnel interface.

Use the **delete** form of this command to remove a tunnel interface and all its configuration.

Use the **show** form of this command to view tunnel configuration.

interfaces tunnel <tunx> address <ipv4net>

Sets a primary or secondary IP address for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **address** *ipv4net*

delete interfaces tunnel *tunx* **address** [*ipv4net*]

show interfaces tunnel *tunx* **address**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      address ipv4net
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
ipv4net	Multi-node. An IPv4 network address in the format *ip-address/prefix*. You can define more than one IP address for a tunnel interface by creating multiple **address** configuration nodes.

Default

None.

Usage Guidelines

Use this command to assign a primary or secondary IP address to a tunnel interface. At least one address must be configured for the tunnel interface to function.

Use the **set** form of this command to create an IP address for a tunnel interface. Note that you cannot use set to change an existing address; you must delete the address to be changed and create a new one.

Use the **delete** form of this command to remove an IP network address for a tunnel interface. At least one address must remain for the tunnel to function.

Use the **show** form of this command to view address configuration for a tunnel interface.

interfaces tunnel <tunx> description <descr>

Specifies a description for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **description** *descr*

delete interfaces tunnel *tunx* **description**

show interfaces tunnel *tunx* **description**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      description descr
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
descr	A brief description for the interface. The default is an empty string.

Default

None.

Usage Guidelines

Use this command to record a brief description for a tunnel interface. If the description contains spaces, it must be enclosed in double quotes.

Use the **set** form of this command to record a brief description description for the tunnel interface.

Use the **delete** form of this command to remove a description for the tunnel interface.

Use the **show** form of this command to view a description for the tunnel interface.

interfaces tunnel <tunx> disable

Disables a tunnel interface without discarding configuration.

Syntax

set interfaces tunnel *tunx* **disable**

delete interfaces tunnel *tunx* **disable**

show interfaces tunnel *tunx*

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      disable
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.

Default

The tunnel interface is enabled.

Usage Guidelines

Use this command to disable a tunnel interface without discarding configuration

Use the **set** form of this command to disable the tunnel interface.

Use the **delete** form of this command to enable the tunnel interface.

Use the **show** form of this command to view the configuration for the tunnel interface.

interfaces tunnel <tunx> encapsulation

Sets the encapsulation for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **encapsulation** {**gre** | **gre-bridge** | **ipip** | **ipip6** | **ip6ip6** | **sit**}

delete interfaces tunnel *tunx* **encapsulation**

show interfaces tunnel *tunx* **encapsulation**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
    tunnel tunx {
        encapsulation [gre|gre-bridge|ipip|ipip6|ip6ip6|sit]
        }
    }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
gre	Uses Generic Routing Encapsulation (GRE) to encapsulate transported packets.
ipip	Uses IP-in-IP to encapsulate transported packets.
ipip6	Uses IP-in-IPv6 to encapsulate transported packets.
ip6ip6	Uses IPv6-in-IPv6 to encapsulate transported packets.

Default

GRE is the encapsulation type.

Usage Guidelines

Use this command to set the encapsulation type for a tunnel.

The Generic Routing Encapsulation (GRE) protocol provides a simple-general purpose mechanism for encapsulating packets from a wide variety of network protocols to be forwarded over another protocol. The original packet (the "passenger" packet) can be one of many arbitrary network protocols—for example a multicast packet, an IPv6 packet, or a non-IP LAN protocol such as AppleTalk, Banyen VINES, or Novell IPX. The delivery protocol can be one of a number of routable IP protocols.

One of the limitations of regular GRE encapsulated tunnels is that they cannot be added to a bridge group. GRE for Bridging (using the **gre-bridge** keyword) provides this ability. It should only be used in cases where tunnel interfaces are to be included in a bridge group. See the *Vyatta Bridging Reference Guide* for further information.

The IP-in-IP encapsulation protocol is used to tunnel between networks that have different capabilities or policies. For example, an IP-in-IP tunnel can be used to forward multicast packets across a section of a network (such as an IPsec tunnel) that does not support multicast routing. An IP-in-IP tunnel can also be used to influence the routing of the packet, or to deliver a packet to a mobile device using Mobile IP.

The SIT encapsulation is used to tunnel IPv6 across an IPv4 network.

Use the **set** form of this command to set the encapsulation type for a tunnel interface.

Use the **delete** form of this command to remove restore the default encapsulation type for a tunnel interface.

Use the **show** form of this command to view encapsulation configuration for a tunnel interface.

interfaces tunnel <tunx> local-ip <ipv4>

Sets the IP address for the local endpoint of a tunnel.

Syntax

set interfaces tunnel *tunx* **local-ip** *ipv4*

delete interfaces tunnel *tunx* **local-ip**

show interfaces tunnel *tunx* **local-ip**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      local-ip ipv4
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
ipv4	Mandatory. The IPv4 address to be used as the tunnel endpoint on the local router. The IP address must already be configured for the interface.

Default

None.

Usage Guidelines

Use this command to specify the IP address to use as the local endpoint of the tunnel.

Use the **set** form of this command to set address of the local endpoint of the tunnel.

Use the **delete** form of this command to remove the local endpoint of the tunnel. Note that the tunnel will not function without both endpoints configured.

Use the **show** form of this command to view local tunnel endpoint configuration.

interfaces tunnel <tunx> mtu <mtu>

Sets the MTU size for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **mtu** *mtu*

delete interfaces tunnel *tunx* **mtu**

show interfaces tunnel *tunx* **mtu**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      mtu mtu
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
mtu	Optional. The MTU, in octets, for the tunnel interface. The range is 0 to 8042, where 0 means fragmentation is never performed. The default is 1476.

Default

Tunnel interface packets have an MTU of 1476.

Usage Guidelines

Use this command to set the maximum transfer unit (MTU) for encapsulated packets traversing the tunnel.

This MTU is applied to the packets embedded in the encapsulating protocol; it is not the MTU of the "carrier" packets themselves. The MTU of carrier packets is dictated by the MTU of the physical interface transmitting and receiving the tunnel packets.

Use the **set** form of this command to set the MTU value for encapsulated packets.

Use the **delete** form of this command to restore the default MTU value for encapsulated packets.

Use the **show** form of this command to view the encapsulated packet MTU configuration.

interfaces tunnel <tunx> parameters ip key <key>

Defines an authentication key for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **parameters ip key** *key*

delete interfaces tunnel *tunx* **parameters ip key**

show interfaces tunnel *tunx* **parameters ip key**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ip {
            key key
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
key	A key for authenticating the local endpoint to the remote endpoint. The key must match on both ends of the connection for the tunnel to be established.

Default

No key is configured; authentication is not required.

Usage Guidelines

Use this command to provide a simple password-like numeric key for authenticating tunnel endpoints to one another. For the tunnel to be established, keys must be identical at both ends of the tunnel.

Use the **set** form of this command to specify a key for the tunnel interface.

Use the **delete** form of this command to remove the key for the tunnel interface.

Use the **show** form of this command to view the key for the tunnel interface.

interfaces tunnel <tunx> parameters ip tos <tos>

Specifies the value to be written into the ToS byte of the tunnel packet's IP header.

Syntax

set interfaces tunnel *tunx* **parameters ip tos** *tos*

delete interfaces tunnel *tunx* **parameters ip tos**

show interfaces tunnel *tunx* **parameters ip tos**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ip {
            tos tos
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
tos	Optional. The value to be written into the ToS byte in tunnel packet IP headers (the carrier packet). The range is 0 to 99, where 0 means tunnel packets copy the ToS value from the packet being encapsulated (the passenger packet). The default is 0.

Default

The ToS byte of the encapsulated packet is copied into the ToS byte of the tunnel packet's IP header.

Usage Guidelines

Use this command to specify the value to be written in the 8-bit Type of Service (ToS) byte of the IP header for packets traversing a tunnel interface. The ToS byte of a packet's IP header specifies the forwarding behavior to be applied to the packet.

Use the **set** form of this command to specify the ToS value to write into a tunnel packet's IP header.

Use the **delete** form of this command to restore the default behavior for the ToS byte.

Use the **show** form of this command to view ToS byte configuration.

interfaces tunnel <tunx> parameters ip ttl <ttl>

Defines the time-to-live (TTL) value to be written into the tunnel packet's IP header.

Syntax

set interfaces tunnel *tunx* **parameters ip ttl** *ttl*

delete interfaces tunnel *tunx* **parameters ip ttl**

show interfaces tunnel *tunx* **parameters ip ttl**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ip {
            ttl ttl
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
ttl	Optional. The value to be written into the TTL field in tunnel packet IP headers (the carrier packet). The range is 0 to 255, where 0 means tunnel packets copy the TTL value from the packet being encapsulated (the passenger packet). The default is 255.

Default

The TTL byte of the encapsulated packet is copied into the TTL byte of the tunnel packet's IP header.

Usage Guidelines

Use this command to specify the value to be written in the TTL field of the IP header for packets traversing a tunnel interface. The TTL field of a packet's IP header used to limit the lifetime of an IP packet and to prevent indefinite packet looping.

Use the **set** form of this command to specify the TTL value to write into a tunnel packet's IP header.

Use the **delete** form of this command to restore the default behavior for the TTL field.

Use the **show** form of this command to view TTL field configuration.

interfaces tunnel <tunx> parameters ipv6 encaplimit <limit>

Defines the maximum number of times the tunnel packet can be encapsulated.

Syntax

set interfaces tunnel *tunx* **parameters ipv6 encaplimit** *limit*

delete interfaces tunnel *tunx* **parameters ipv6 encaplimit**

show interfaces tunnel *tunx* **parameters ipv6 encaplimit**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ipv6 {
            encaplimit limit
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
limit	Optional. The maximum levels of encapsulation that are permitted to be prepended to the packet. The range is 0 to 255. The default is 4. If no value is specified, the command is disabled.

Default

None.

Usage Guidelines

Use this command to specify the maximum number of times the tunnel packet can be encapsulated.

Use the **set** form of this command to specify the maximum number of times the packet can be encapsulated.

Use the **delete** form of this command to restore the default maximum number of times the tunnel packet can be encapsulated.

Use the **show** form of this command to view tunnel encapsulation limit configuration.

interfaces tunnel <tunx> parameters ipv6 flowlabel <flowlabel>

Defines the flowlabel of the encapsulating ipv6 header.

Syntax

set interfaces tunnel *tunx* **parameters ipv6 flowlabel** *flowlabel*

delete interfaces tunnel *tunx* **parameters ipv6 flowlabel**

show interfaces tunnel *tunx* **parameters ipv6 flowlabel**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ipv6 {
            flowlabel flowlabel
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
flowlabel	Optional. The flowlabel o f the encapsulating ipv6 header. The range is 0 to 0xfffff. The default is inherit.

Default

The flowlabel field of the encapsulated packet is copied into the flowlabel field of the tunnel packet's IPv6 header.

Usage Guidelines

Use this command to specify the flowlabel of the encapsulating ipv6 header.

Use the **set** form of this command to specify the flowlabel of the encapsulating ipv6 header.

Use the **delete** form of this command to restore the default flowlabel of the encapsulating ipv6 header.

Use the **show** form of this command to view flowlabel encapsulation configuration.

interfaces tunnel <tunx> parameters ipv6 hoplimit <limit>

Defines the hoplimit value to be written into the tunnel packet's IPv6 header.

Syntax

set interfaces tunnel *tunx* **parameters ipv6 hoplimit** *limit*

delete interfaces tunnel *tunx* **parameters ipv6 hoplimit**

show interfaces tunnel *tunx* **parameters ipv6 hoplimit**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ipv6 {
            hoplimit limit
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
hoplimit	Optional. The value to be written into the hoplimit field in tunnel packet IPv6 headers (the carrier packet). The range is 0 to 255, where 0 means tunnel packets copy the hoplimit value from the packet being encapsulated (the passenger packet). The default is 64.

Default

The hoplimit byte of the encapsulated packet is copied into the hoplimit byte of the tunnel packet's IPv6 header.

Usage Guidelines

Use this command to specify the value to be written in the hoplimit field of the IPv6 header for packets traversing a tunnel interface. The hoplimit field of a packet's IPv6 header used to limit the lifetime of an IPv6 packet and to prevent indefinite packet looping.

Use the **set** form of this command to specify the hoplimit value to write into a tunnel packet's IPv6 header.

Use the **delete** form of this command to restore the default behavior for the hoplimit field.

Use the **show** form of this command to view the hoplimit field configuration.

interfaces tunnel <tunx> parameters ipv6 tclass <class>

Specifies the value to be written into the tclass byte of the tunnel packet's IPv6 header.

Syntax

set interfaces tunnel *tunx* **parameters ipv6 tclass** *class*

delete interfaces tunnel *tunx* **parameters ipv6 tclass**

show interfaces tunnel *tunx* **parameters ipv6 tclass**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      paramaters
         ipv6 {
            tclass class
         }
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
class	Optional. The value to be written into the tclass byte in tunnel packet IPv6 headers (the carrier packet). The range is 0 to 0xff, where 0 means tunnel packets copy the tclass byte from the packet being encapsulated (the passenger packet). The default is inherit.

Default

The tclass byte of the encapsulated packet is copied into the tclass byte of the tunnel packet's IPv6 header.

Usage Guidelines

Use this command to specify the value to be written in the tclass byte of the IPv6 header for packets traversing a tunnel interface. The tclass byte of a packet's IPv6 header specifies the forwarding behavior to be applied to the packet.

Use the **set** form of this command to specify the tclass value to write into a tunnel packet's IPv6 header.

Use the **delete** form of this command to restore the default behavior for the tclass field.

Use the **show** form of this command to view the tclass field configuration.

interfaces tunnel <tunx> remote-ip <ipv4>

Sets the IP address for the remote endpoint of a tunnel.

Syntax

set interfaces tunnel *tunx* **remote-ip** *ipv4*

delete interfaces tunnel *tunx* **remote-ip**

show interfaces tunnel *tunx* **remote-ip**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tunx {
      remote-ip ipv4
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun*x*,** where *x* is a non-negative integer.
ipv4	Mandatory. The IPv4 address to be used as the tunnel endpoint on the remote router. The IP address must already be configured for the interface.

Default

None.

Usage Guidelines

Use this command to specify the IP address to use as the remote endpoint of the tunnel.

Use the **set** form of this command to set address of the remote endpoint of the tunnel.

Use the **delete** form of this command to remove the remote endpoint of the tunnel. Note that the tunnel cannot be established without both endpoints configured.

Use the **show** form of this command to view remote tunnel endpoint configuration.

show interfaces tunnel

Displays information about tunnel interfaces.

Syntax

show interfaces tunnel [*tunx* [**brief**] | **detail**]

Command Mode

Operational mode.

Parameters

tunx	Optional. Displays information for the specified tunnel interface. The range is **tun0** to **tun*x***, where *x* is a non-negative integer.
brief	Optional. Displays a brief status of the specified tunnel.
detail	Optional. Displays a detailed status of the tunnel interfaces.

Default

Information is displayed for all tunnel interfaces.

Usage Guidelines

Use this command to view the operational status of tunnel interfaces.

Examples

Example 3-1 shows operational status for the GRE tunnel interface tun0.

Example 3-1 "show interfaces tunnel": Displaying tunnel configuration

```
vyatta@vyatta:~$ show interfaces tunnel
   tun0@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue
   link/gre 192.168.20.2 peer 192.168.20.3
   inet 192.168.20.1/24 brd 192.168.20.255 scope global tun0
   RX:  bytes    packets    errors    dropped    overrunmcast
          0          0         0          0          0   0
   TX:  bytes    packets    errors    dropped    carriercollisions
          0          0         0          0          0   0
```

Glossary of Acronyms

ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6

DLCI	data-link connection identifier
DMI	desktop management interface
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
I/O	Input/Ouput
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM

IPsec	IP security
IPv4	IP Version 4
IPv6	IP Version 6
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
MIB	Management Information Base
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
ND	Neighbor Discovery
NIC	network interface card
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect

PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SSH	Secure Shell
SSID	Service Set Identifier
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol

ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
vif	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access