VYATTA, INC. | Vyatta System

Tunnels

Reference Guide

GRE Tunnels
IP-in-IP Tunnels

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

ISSUE DATE: Apil 2010

DOCUMENT REVISION. R6.0 v03

RELEASED WITH: R6.0

PART NO. A0-0229-10-0004

Table of 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 guide describes commands for configuring and monitoring Generic Routing Encapsulation and IP-in-IP routable tunnel interfaces.

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

- Intended Audience
- Organization of This Guide
- Document Conventions
- Vyatta Publications

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

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 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.

This guide has the following chapters and appendixes:

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 configuration examples for GRE and IP-in-IP tunnels.	5
Chapter 3: Tunnel Commands	This chapter lists the commands for configuring GRE and IP-in-IP tunnels.	13
Glossary of Acronyms		34

Document Conventions

This guide contains advisory paragraphs and uses typographic conventions.

Advisory Paragraphs

This guide uses the following advisory paragraphs:

Warnings alert you to situations that may pose a threat to personal safety, as in the following example:

WARNING *Switch off power at the main breaker before attempting to connect the remote cable to the service power at the utility box.*

Cautions alert you to situations that might cause harm to your system or damage to equipment, or that may affect service, as in the following example:

CAUTION *Restarting a running system will interrupt service.*

Notes provide information you might need to avoid problems or configuration errors:

NOTE *You must create and configure network interfaces before enabling them for routing protocols.*

Typographic Conventions

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.
[*arg1* | *arg2*]	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 in the first case and a comma-separated list in the second case).

Vyatta Publications

More information about the Vyatta system is available in the Vyatta technical library, and on www.vyatta.com and www.vyatta.org.

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.

Chapter 1: Tunneling Overview

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

This chapter presents the following topics:

- Supported Tunnel Types
- Tunnel Interfaces and IPsec
- Supported Standards

Supported Tunnel Types

The Vyatta system supports Generic Routing Encapsulation (GRE) tunnels and IP-in-IP tunnels.

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, Banyen 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. Where security is required, GRE should be used in conjunction with IPsec.

GRE uses IP protocol number 47.

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 IP-in-IP tunnels are not secure. For secure communications, IP-in-IP tunnels should be used together with IPsec.

Tunnel Interfaces and IPsec

GRE, IP-in-IP, and SIT tunnels are not encrypted, and provide no security outside of a simple password-like key that is exchanged in clear text in each packet. This means that GRE, IP-in-IP, and SIT tunnels, on their own, do not provide adequate security for production environments.

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 should 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

Chapter 2: Tunnel Configuration Examples

This chapter provides configuration examples for GRE and IP-in-IP tunnels.

This chapter presents the following topics:

- Before You Begin
- Configuring a Basic GRE Tunnel
- Configuring a More Complex GRE Tunnel

Before You Begin

- In this set of examples, 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 interface must be configured with the IP address you want to use as the source IP for packets sent to the other tunnel endpoint. In this example, IP address 192.0.2.1 is defined on eth1 of WEST, and 192.0.2.33 is defined on eth0 of EAST.

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

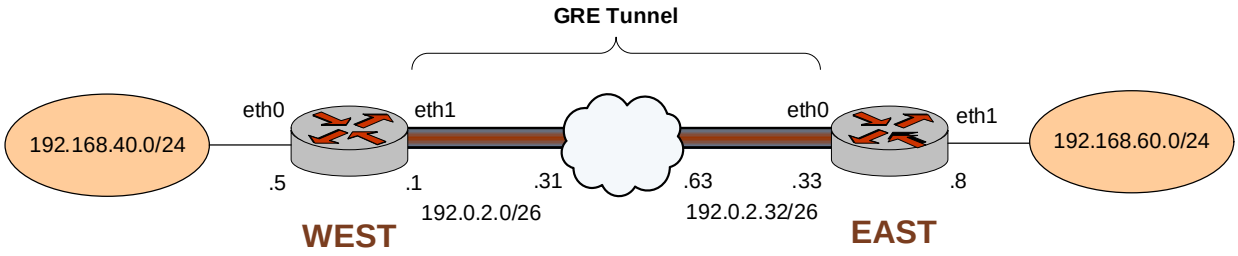
Configuring a Basic GRE Tunnel

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 eth1 on WEST through the wide-area network to eth0 on 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 192.0.2.1 on network 192.0.2.0/26.

- The source IP address of the tunnel endpoint (the **local-ip**) is the same as the address associated with the interface in this example.
- The IP address of the other end of the tunnel is 192.0.2.33 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 192.0.2.1/26` `[edit]`
Specify the source IP address for the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 local-ip 192.0.2.1` `[edit]`
Specify the IP address of the other end of the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 remote-ip 192.0.2.33` `[edit]`
Specify the encapsulation mode for the tunnel.	`vyatta@WEST# set interfaces tunnel tun0 encapsulation gre` `[edit]`
Assign a brief description for the tunnel interface.	`vyatta@WEST# set interfaces tunnel tun0 description “GRE tunnel to EAST”` `[edit]`
Commit the configuration.	`vyatta@WEST# commit` `[edit]`

View the configuration.	`vyatta@WEST# show interfaces tunnel tun0`
	`address 192.0.2.1/26`
	`description “Tunnel to EAST”`
	`encapsulation gre`
	`local-ip 192.0.2.1`
	`remote-ip 192.0.2.33`
	`[edit]`

Configure EAST

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

- The tunnel interface tun0 on EAST is assigned the IP address 192.0.2.33 on network 192.0.2.32/26.

- The source IP address of the tunnel endpoint (the **local-ip**) is the same as the address associated with the interface in this example.
- The IP address of the other end of the tunnel is 192.0.2.1 on WEST. By assumption, this IP address was previously configured for interface eth0 on WEST.

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 192.0.2.33/26** [edit]
Specify the source IP address for the tunnel.	vyatta@EAST# **set interfaces tunnel tun0 local-ip 192.0.2.33** [edit]
Specify the IP address of the other end of the tunnel.	vyatta@EAST# **set interfaces tunnel tun0 remote-ip 192.0.2.1** [edit]
Specify the encapsulation mode for the tunnel.	vyatta@EAST# **set interfaces tunnel tun0 encapsulation gre** [edit]
Assign a brief description for the tunnel interface.	vyatta@EAST# **set interfaces tunnel tun0 description “GRE tunnel to WEST”** [edit]
Commit the configuration.	vyatta@EAST# **commit** [edit]
View the configuration.	vyatta@EAST# **show interfaces tunnel tun0** address 192.0.2.33/26 description “Tunnel to WEST” encapsulation gre local-ip 192.0.2.33 remote-ip 192.0.2.1 [edit]

Configuring 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` `[edit]`
Set the time-to-live.	`vyatta@WEST# set interfaces tunnel tun0 ttl 220` `[edit]`
Set the Type of Service.	`vyatta@WEST# set interfaces tunnel tun0 tos 55` `[edit]`

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

Set the MTU.	`vyatta@WEST# set interfaces tunnel tun0 mtu 1460` `[edit]`
Apply the firewall rule set for incoming packets.	`vyatta@WEST# set interfaces tunnel tun0 firewall in name tun0-fw-in` `[edit]`
Apply the firewall rule set for outgoing packets.	`vyatta@WEST# set interfaces tunnel tun0 firewall out name tun0-fw-out` `[edit]`
Commit the configuration.	`vyatta@WEST# commit` `[edit]`

View the configuration.	`vyatta@WEST# show interfaces tunnel tun0`
	`address 192.0.2.1/26`
	`description “Tunnel to EAST”`
	`encapsulation gre`
	`firewall`
	`in {`
	`name tun0-fw-in`
	`}`
	`out {`
	`name tun0-fw-out`
	`}`
	`}`
	`key 101088`
	`local-ip 192.0.2.1`
	`mtu 1460`
	`remote-ip 192.0.2.33`
	`tos 55`
	`ttl 220`
	`[edit]`

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** [edit]
Set the time-to-live.	vyatta@EAST# **set interfaces tunnel tun0 ttl 220** [edit]
Set the Type of Service.	vyatta@EAST# **set interfaces tunnel tun0 tos 55** [edit]
Set the MTU.	vyatta@EAST# **set interfaces tunnel tun0 mtu 1460** [edit]
Apply the firewall rule set for incoming packets.	vyatta@EAST# **set interfaces tunnel tun0 firewall in name tun0-fw-in** [edit]
Apply the firewall rule set for outgoing packets.	vyatta@EAST# **set interfaces tunnel tun0 firewall out name tun0-fw-out** [edit]
Commit the configuration.	vyatta@EAST# **commit** [edit]

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

View the configuration.

```
vyatta@EAST# show interfaces tunnel tun0
   address 192.0.2.33/26
   description “Tunnel to WEST”
   encapsulation gre
   firewall
      in {
         name tun0-fw-in
      }
      out {
         name tun0-fw-out
      }
   }
   key 101088
   local-ip 192.0.2.33
   mtu 1460
   remote-ip 192.0.2.1
   tos 55
   ttl 220
[edit]
```

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> key <key>	Defines an authentication key 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> remote-ip <ipv4>	Sets the IP address for the remote endpoint of a tunnel.
interfaces tunnel <tunx> tos <tos>	Specifies the value to be written into the ToS byte of the tunnel packet's IP header.
interfaces tunnel <tunx> ttl <ttl>	Defines the time-to-live (TTL) value to be written into the tunnel packet's IP 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*.

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 **tun23**.

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 tun0..tun23 {
    }
}
```

Parameters

tunx	Mandatory. Multi-node. An identifier for the tunnel interface you are defining. The range is **tun0** to **tun23**. You can define multiple tunnel interfaces by creating multiple **tunnel** configuration nodes.

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 tun0..tun23 {
      address ipv4net
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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 tun0..tun23 {
      description text
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
descr	A mnemonic name or 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 tun0..tun23 {
      disable
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.

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 | ipip | sit**}

delete interfaces tunnel *tunx* **encapsulation**

show interfaces tunnel *tunx* **encapsulation**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tun0..tun23 {
      encapsulation [gre|ipip|sit]
      }
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun9**.
gre	Uses Generic Routing Encapsulation (GRE) to encapsulate transported packets.
ipip	Uses IP-in-IP to encapsulate transported packets.
sit	Uses Simple Internet Transition (SIT) encapsulation.

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.

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 typically 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> key <key>

Defines an authentication key for a tunnel interface.

Syntax

set interfaces tunnel *tunx* **key** *key*

delete interfaces tunnel *tunx* **key**

show interfaces tunnel *tunx* **key**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tun0..tun23 {
      key 0-999999
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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> 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 tun0..tun23 {
      local-ip ipv4
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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 tun0..tun23 {
      mtu mtu
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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> 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 tun0..tun23 {
      remote-ip ipv4
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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.

interfaces tunnel <tunx> tos <tos>

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

Syntax

set interfaces tunnel *tunx* **tos** *tos*

delete interfaces tunnel *tunx* **tos**

show interfaces tunnel *tunx* **tos**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tun0..tun23 {
      tos tos
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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> ttl <ttl>

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

Syntax

set interfaces tunnel *tunx* **ttl** *ttl*

delete interfaces tunnel *tunx* **ttl**

show interfaces tunnel *tunx* **ttl**

Command Mode

Configuration mode.

Configuration Statement

```
interfaces {
   tunnel tun0..tun23 {
      ttl 0-255
   }
}
```

Parameters

tunx	Mandatory. The name of the tunnel interface you are configuring. The range is **tun0** to **tun23**.
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 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 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.

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 **tun23**.
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   overrun   mcast
          0     0         0         0         0         0
   TX:  bytes   packets   errors    dropped   carrier   collisions
          0     0         0         0         0         0
```

Glossary of Acronyms

ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
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
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
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access 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
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RA	router advertisement
RIB	Routing Information Base
RIP	Routing Information Protocol

RIPng	RIP next generation
RS	router solicitation
Rx	receive
SLAAC	Stateless address auto-configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SSH	Secure Shell
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TCP	Transmission Control Protocol
ToS	Type of Service
Tx	transmit
UDP	User Datagram Protocol
vif	virtual interface
VLAN	virtual LAN
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network