

## Chapter 32

# Configuring Tunnel Interfaces

By encapsulating arbitrary packets inside a transport protocol, tunneling provides a private, secure path through an otherwise public network. Tunnels connect discontinuous subnetworks and enable encryption interfaces, virtual private networks (VPNs), and Multiprotocol Label Switching (MPLS). If you have a Tunnel Physical Interface Card (PIC) installed in your routing platform, you can configure unicast, multicast, and logical tunnels.

You can configure two types of tunnels for VPNs: one to facilitate routing table lookups and another to facilitate VPN routing and forwarding instance (VRF) table lookups.

For information about encryption interfaces, see “Configuring Encryption Interfaces” on page 337 and the *JUNOS System Basics Configuration Guide*. For information about VPNs, see the *JUNOS VPNs Configuration Guide*. For information about MPLS, see the *JUNOS MPLS Applications Configuration Guide*.

The JUNOS software supports the following tunnel encapsulations:

- Generic routing encapsulation (GRE)
- IP over IP (IP-IP)
- Virtual Private Network (VPN)
- PIM encapsulation

This chapter discusses the following tunnel interface configuration tasks:

- Configuring a Unicast Tunnel on page 570
- Configuring a Multicast Tunnel on page 573
- Configuring a Logical Tunnel on page 574
- Configuring a Tunnel Interface for Routing Table Lookup on page 576
- Configuring a Tunnel Interface for VRF Table Lookup on page 576
- Configuring PIM Tunnels on page 577
- Configuring an IPv6-over-IPv4 Tunnel on page 578

For examples of tunnel interface configuration, see the following sections:

Example: Configuring Unicast Tunnels on page 578

Example: Configuring a Virtual Loopback Tunnel Interface for VRF Table Lookup on page 580

Example: Configuring an IPv6-over-IPv4 Tunnel on page 581

## Configuring a Unicast Tunnel

---

To configure a unicast tunnel, you configure the `gr` interface (to use GRE encapsulation) or the `ip` interface (to use IP-IP encapsulation) and include the `tunnel` and `family` statements:

```
gr-fpc/pic/port or ip-fpc/pic/port {
  unit logical-unit-number {
    tunnel {
      backup-destination address;
      destination address;
      key number;
      routing-instance {
        destination routing-instance-name;
      }
      source source-address;
      ttl number;
    }
    family family {
      address address {
        destination address;
      }
    }
  }
}
```

You can include these statements at the following hierarchy levels:

[edit interfaces]

[edit logical-routers *logical-router-name* interfaces]

You can configure multiple logical units for each GRE or IP-IP interface, and you can configure only one tunnel per unit.

Each tunnel interface must be a point-to-point interface. Point to point is the default interface connection type, so you do not need to include the point-to-point statement in the logical interface configuration.

You must specify the tunnel's destination and source addresses. The remaining statements are optional.



**NOTE:** For transit packets exiting the tunnel, forwarding path features, such as RPF, forwarding table filtering, source class usage, destination class usage, and stateless firewall filtering, are not supported on the interfaces you configure as tunnel sources.

To set the time-to-live (TTL) field that is included in the encapsulating header, include the `ttl` statement. If you explicitly configure a TTL value for the tunnel, you must configure it to be one larger than the number of hops in the tunnel. For example, if the tunnel has seven hops, you must configure a TTL value of 8.

You must configure at least one family on the logical interface. To enable MPLS over GRE tunnel interfaces, you must include the `family mpls` statement in the GRE interface configuration. In addition, you must configure the `protocols` statements to enable RSVP, MPLS, and LSPs over GRE tunnels. Unicast tunnels are bidirectional.

A configured tunnel cannot go through Network Address Translation (NAT) at any point along the way to the destination. For more information, see “Example: Configuring Unicast Tunnels” on page 578 and the *JUNOS MPLS Applications Configuration Guide*.

For GRE tunnel interfaces on Adaptive Services PICs, you can configure additional tunnel attributes, as described in the following sections:

Configuring a Key Number on GRE Tunnels on page 571

Enabling Fragmentation on GRE Tunnels on page 572

Specifying an MTU Setting for the Tunnel on page 573

### **Configuring a Key Number on GRE Tunnels**

For Adaptive Services PICs on M-series platforms (except the M320), you can assign a key value to identify an individual traffic flow within a GRE tunnel, as defined in RFC 2890.

Each IPv4 packet entering the tunnel is encapsulated with the GRE tunnel key value. Each IPv4 packet exiting the tunnel is verified by the GRE tunnel key value and de-encapsulated. The Adaptive Services PIC drops packets that do not match the configured key value.

To assign a key value to a GRE tunnel interface, include the `key` statement:

```
key number;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number tunnel]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number tunnel]
```

The key number can be 0 through 4,294,967,295. You must configure the same GRE tunnel key value on tunnel endpoints.

### **Enabling Fragmentation on GRE Tunnels**

For GRE tunnel interfaces on Adaptive Services PICs only, you can enable fragmentation of IPv4 packets in GRE tunnels.

By default, IPv4 traffic transmitted over GRE tunnels is not fragmented. To enable fragmentation of IPv4 packets in GRE tunnels, include the `clear-dont-fragment-bit` statement:

```
clear-dont-fragment-bit;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number]
```

When you include the `clear-dont-fragment-bit` statement in the configuration, the don't-fragment (DF) bit is cleared on all packets, even packets that do not exceed the tunnel maximum transmission unit (MTU). If the packet's size exceeds the tunnel's MTU value, the packet is fragmented before encapsulation. If the packet's size does not exceed the tunnel's MTU value, the packet is not fragmented.

You can also clear the DF bit in packets transmitted over IPSec tunnels. For more information, see the *JUNOS Services Interfaces Configuration Guide*.

### Specifying an MTU Setting for the Tunnel

To enable key numbers and fragmentation on GRE tunnels (as described in “Configuring a Key Number on GRE Tunnels” on page 571 and “Enabling Fragmentation on GRE Tunnels” on page 572), you must also specify an MTU setting for the tunnel.

To specify an MTU setting for the tunnel, include the `mtu` statement:

```
mtu bytes;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces gr-fpc/pic/port unit logical-unit-number family inet]
```

```
[edit logical-router logical-router-name interfaces gr-fpc/pic/port unit
logical-unit-number family inet]
```

For more information about tunnels and adaptive services interfaces, see the *JUNOS Services Interfaces Configuration Guide*.

### Configuring a Multicast Tunnel

---

For interfaces that carry IPv4 or IPv6 traffic, you can configure multicast tunnels. To configure a multicast tunnel, include the `multicast-only` statement:

```
multicast-only;
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces interface-name unit logical-unit-number family family]
```

```
[edit logical-routers logical-router-name interfaces interface-name unit
logical-unit-number family family]
```

Multicast tunnels filter all unicast packets; if an incoming packet is not destined for a 224/8 or greater prefix, the packet is dropped and a counter is incremented.

You can configure this property on GRE, IP-IP, PIM, and multicast tunnel interfaces only.

## Configuring a Logical Tunnel

---

If your routing platform is equipped with a Tunnel Services PIC, you can configure logical tunnel interfaces. Logical tunnel interfaces allow you to connect logical router. To connect two logical routers, you configure a logical tunnel interface on both logical routers. Then you configure a peer relationship between the logical tunnel interfaces, thus creating a point-to-point connection.

To configure a point-to-point connection between two logical routers, you configure the logical tunnel interface:

```
lt-fpc/pic/port {
  unit logical-unit-number {
    encapsulation type;
    peer-unit unit-number; # Peering logical router unit number
    dlcI dlcI-number;
    family family;
  }
}
```

You can include these statements at the following hierarchy levels:

[edit interfaces]

[edit logical-routers *logical-router-name* interfaces]

The encapsulation type can be ethernet, ethernet-ccc, ethernet-vpls, frame-relay, frame-relay-ccc, vlan, vlan-ccc, or vlan-vpls.

The protocol family can be ccc, inet, inet6, iso, mpls, or tcc.

When configuring logical tunnel interfaces, note the following:

The peering logical interfaces must belong to the same logical tunnel interface derived from the Tunnel Services PIC.

You can configure only one peer unit for each logical interface. For example, unit 0 cannot peer with both unit 1 and unit 2.

To enable the logical tunnel interface, you must configure at least one physical interface statement.

Logical tunnels are not supported with Adaptive Services (AS) or Link Services PICs.

Logical tunnel interfaces require an enhanced FPC.

**Example: Configuring a Logical Tunnel**

Configure three logical tunnels:

```
[edit interfaces]
lt-4/2/0 {
    description "Logical tunnel interface connects three logical routers"
}

[edit logical-routers]
lr1 {
    interfaces lt-4/2/0 {
        unit 12 {
            peer-unit 21; # Peering with lr2
            encapsulation frame-relay;
            dlci 612;
            family inet;
        }
        unit 13 {
            peer-unit 31; # Peering with lr3
            encapsulation frame-relay-ccc;
            dlci 613;
        }
    }
}
lr2 {
    interfaces lt-4/2/0 {
        unit 21 {
            peer-unit 12; # Peering with lr1
            encapsulation frame-relay-ccc;
            dlci 612;
            family inet;
        }
        unit 23 {
            peer-unit 32; # Peering with lr3
            encapsulation frame-relay;
            dlci 623;
        }
    }
}
lr3 {
    interfaces lt-4/2/0 {
        unit 31 {
            peer-unit 13; # Peering with lr1
            encapsulation frame-relay;
            dlci 613;
            family inet;
        }
        unit 32 {
            peer-unit 23; # Peering with lr2
            encapsulation frame-relay-ccc;
            dlci 623;
        }
    }
}
```

## Configuring a Tunnel Interface for Routing Table Lookup

---

To configure tunnel interfaces to facilitate routing table lookups for VPNs, you specify a tunnel's endpoint IP addresses and associate them with a routing instance that belongs to a particular routing table. This enables the JUNOS software to search in the appropriate routing table for the route prefix, because the same prefix can appear in multiple routing tables.

By default, the tunnel route prefixes are assumed to be in the default Internet routing table `inet.0`. If the tunnel destination address is not in `inet.0`, you need to specify which routing table to search for the tunnel destination address.

To configure the destination VPN, include the `routing-instance` statement:

```
routing-instance {
    destination routing-instance-name;
}
```

You can include this statement at the following hierarchy levels:

```
[edit interfaces gr-fpc/pic/port unit logical-unit-number tunnel]
```

```
[edit logical-routers logical-router-name interfaces gr-fpc/pic/port unit
logical-unit-number tunnel]
```

This configuration indicates that the tunnel's destination address is in routing instance *routing-instance-name*. For more information about VPNs, see the *JUNOS VPNs Configuration Guide*.

## Configuring a Tunnel Interface for VRF Table Lookup

---

To enable egress filtering, you can either configure filtering based on the IP header, or you can configure the virtual loopback tunnel interface on routing platforms equipped with a Tunnel PIC. For more information about filtering traffic based on the IP header, see the *JUNOS VPNs Configuration Guide*.

You can configure a virtual loopback tunnel interface to facilitate VPN routing and forwarding instance (VRF) table lookup based on MPLS labels. You might want to enable this functionality so you can do either of the following:

Forward traffic on a PE-router-to-CE-device interface, in a shared medium, where the CE device is a Layer 2 switch without IP capabilities (for example, a metro Ethernet switch).

The first lookup is done based on the VPN label to determine which VRF table to refer to, and the second lookup is done on the IP header to determine how to forward packets to the correct end hosts on the shared medium.

Perform egress filtering at the egress PE router.

The first lookup on the VPN label is done to determine which VRF table to refer to, and the second lookup is done on the IP header to determine how to filter and forward packets. You can enable this functionality by configuring output filters on the VRF interfaces.

To configure a virtual loopback tunnel interface to facilitate VRF table lookup based on MPLS labels, you specify a virtual loopback tunnel interface name and associate it with a routing instance that belongs to a particular routing table. The packet loops back through the virtual loopback tunnel interface for route lookup.

To specify a virtual loopback tunnel interface name, you configure the virtual loopback tunnel interface and include the `family inet` and `family mpls` statements:

```
[edit interfaces]
vt-fpc/pic/port {
  unit 0 {
    family inet;
    family mpls;
  }
  unit 1 {
    family inet;
  }
}
```

To associate the virtual loopback tunnel interface with a routing instance, include the virtual loopback tunnel interface name at the `[edit routing-instances]` hierarchy level:

```
[edit routing-instances] {
interface vt-fpc/pic/port;
```



**NOTE:** For the virtual loopback tunnel interface, none of the logical interface statements are valid.

---

## Configuring PIM Tunnels

---

PIM tunnels are enabled automatically on routing platforms that have a tunnel PIC and on which you enable PIM sparse mode. You do not need to configure the tunnel interface.

PIM tunnels are unidirectional.

In PIM sparse mode, the first-hop routing platform encapsulates packets destined for the rendezvous point routing platform. The packets are encapsulated with a unicast header and are forwarded through a unicast tunnel to the rendezvous point. The rendezvous point then de-encapsulates the packets and transmits them through its multicast tree. To perform the encapsulation and de-encapsulation, the first-hop and rendezvous point routing platforms must be equipped with Tunnel PICs.

The JUNOS software creates two interfaces to handle PIM tunnels:

`pe`—Encapsulates packets destined for the rendezvous point. This interface is present on the first-hop routing platform.

`pd`—De-encapsulates packets at the rendezvous point. This interface is present on the rendezvous point.

## Configuring an IPv6-over-IPv4 Tunnel

---

If you have a Tunnel PIC installed in your routing platform, you can configure IPv6-over-IPv4 tunnels. To do this, you configure a unicast tunnel across an existing IPv4 network infrastructure. IPv6 packets are encapsulated in IPv4 headers and sent across the IPv4 infrastructure through the configured tunnel. You manually configure configured tunnels on each endpoint.

IPv6-over-IPv4 tunnels are defined in RFC 2893, *Transition Mechanisms for IPv6 Hosts and Routers*. For information about configuring a unicast tunnel, see “Configuring a Unicast Tunnel” on page 570. For an IPv6-over-IPv4 tunnel configuration example, see “Example: Configuring an IPv6-over-IPv4 Tunnel” on page 581.

### Example: Configuring Unicast Tunnels

---

Configure two unnumbered IP-IP tunnels:

```
[edit]
interfaces {
  ip-0/3/0 {
    unit 0 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.253;
      }
      family inet;
    }
    unit 1 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.254;
      }
      family inet;
    }
  }
}
```

To configure a numbered tunnel interface, include an address under family inet:

```
[edit]
interfaces
  ip-0/3/0 {
    unit 0 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.253;
      }
      family inet {
        address 10.5.5.1/30;
      }
    }
    unit 1 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.254;
      }
      family inet {
        address 10.6.6.100/30;
      }
    }
  }
}
```

To configure MPLS over GRE tunnels, include the family mpls statement:

```
interfaces {
  gr-1/2/0 {
    unit 0 {
      tunnel {
        source 192.168.1.1;
        destination 192.168.1.2;
      }
      family inet {
        address 10.1.1.1/30;
      }
      family iso;
      family mpls;
    }
  }
}
```

## Example: Configuring a Virtual Loopback Tunnel Interface for VRF Table Lookup

---

Configure a virtual loopback tunnel interface for VRF table lookup:

```
[edit routing-instances]
routing-instance-1 {
  instance-type vrf;
  interface vt-1/0/0.0;
  interface so-0/2/2.0;
  route-distinguisher 2:3;
  vrf-import VPN-A-import;
  vrf-export VPN-A-export;
  routing-options {
    static {
      route 10.0.0.0/8 next-hop so-0/2/2.0;
    }
  }
}
routing-instance-2 {
  instance-type vrf;
  interface vt-1/0/0.1;
  interface so-0/3/2.0;
  route-distinguisher 4:5;
  vrf-import VPN-B-import;
  vrf-export VPN-B-export;
  routing-options {
    static {
      route 10.0.0.0/8 next-hop so-0/3/2.0;
    }
  }
}
}

[edit interfaces]
vt-1/0/0 {
  unit 0 {
    family inet;
    family mpls;
  }
  unit 1 {
    family inet;
  }
}
```

## Example: Configuring an IPv6-over-IPv4 Tunnel

---

Configure a tunnel on both sides of the connection.

```
On Router A    [edit]
                 interfaces {
                   gr-1/0/0 {
                     unit 0 {
                       tunnel {
                         source 10.19.2.1;
                         destination 10.19.3.1;
                       }
                       family inet6 {
                         address 7019::1/126;
                       }
                     }
                   }
                 }
```

```
On Router B    [edit]
                 interfaces {
                   gr-1/0/0 {
                     unit 0 {
                       tunnel {
                         source 10.19.3.1;
                         destination 10.19.2.1;
                       }
                       family inet6 {
                         address 7019::2/126;
                       }
                     }
                   }
                 }
```

