

## Chapter 31

# Configure 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 PIC installed in your router, 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 “Configure Encryption Interfaces” on page 235 and the *JUNOS Internet Software System Basics Configuration Guide*. For information about VPNs, see the *JUNOS Internet Software VPNs Configuration Guide*. For information about MPLS, see the *JUNOS Internet Software MPLS Applications Configuration Guide*.

The JUNOS software supports the following tunnel encapsulations:

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

This chapter discusses the following tunnel interface configuration tasks:

- Configure a Unicast Tunnel on page 418
- Configure a Multicast Tunnel on page 419
- Configure a Logical Tunnel on page 419
- Configure a Tunnel Interface for Routing Table Lookup on page 420
- Configure a Tunnel Interface for VRF Table Lookup on page 421
- Configure PIM Tunnels on page 423
- Configure an IPv6-over-IPv4 Tunnel on page 423

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

Example: Configure Unicast Tunnels on page 424

Example: Configure a Virtual Loopback Tunnel Interface for VRF Table Lookup on page 425

Example: Configure an IPv6-over-IPv4 Tunnel on page 426

Example: Configure a Logical Tunnel on page 427

## Configure 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:

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

You can configure 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.

To set the TTL field that is included in the encapsulating header, include the `tll` 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: Configure Unicast Tunnels” on page 424 and the *JUNOS Internet Software MPLS Applications Configuration Guide*.

## Configure a Multicast Tunnel

---

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

```
multicasts-only;
```

You can configure 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 MT tunnels only.

## Configure a Logical Tunnel

---

If your router is equipped with a Tunnel Services PIC, you can configure logical tunnel interfaces. Logical tunnel interfaces allow you to connect logical routers. 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.

You can configure each logical tunnel interface with either Frame Relay or Frame Relay CCC encapsulation, and with the IP, IPv6, ISO, or MPLS protocol family.

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 (frame-relay | frame-relay-ccc);
    peer-unit unit-number; #peering logical router unit number
    dlcI dlcI-number;
    family (inet | inet6 | iso | mpls);
  }
}
```

You can configure these statements at the following hierarchy levels:

[edit interfaces]

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

When configuring logical tunnel interfaces, note the following:

You must specify either Frame Relay or Frame Relay circuit-cross connect (CCC) as the logical interface encapsulation.

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 or Link Services PICs.

Logical tunnel interfaces require an enhanced FPC.

For more information about configuring logical routers, see the *JUNOS Internet Software Routing Protocols Configuration Guide*.

## Configure a Tunnel Interface for Routing Table Lookup

---

To configure tunnel interfaces to facilitate routing table lookups for VPNs, you specify a tunnel's end point 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. To configure the destination VPN, include the routing-instance statement:

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

You can configure 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*. By default, the tunnel route prefixes are assumed to be in the default Internet routing table *inet.0*.



**NOTE:** If you configure a virtual loopback tunnel interface and the `vrf-table-label` statement on the same routing instance, the `vrf-table-label` statement takes precedence over the virtual loopback tunnel interface. For more information, see “Configure a Tunnel Interface for VRF Table Lookup” on page 421.

For more information about VPNs, see the *JUNOS Internet Software VPNs Configuration Guide*.

## Configure 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 routers equipped with a Tunnel PIC. Table 16 describes each method.

**Table 16: Methods for Configuring Egress Filtering**

Method	Interface Type	Configuration Guidelines	Comments
Filter traffic based on the IP header	Non-channelized PPP/HDLC core-facing SONET interfaces	Include the <code>vrf-table-label</code> statement at the [edit routing-instances <i>instance-name</i> ] hierarchy level.  For more information, see the <i>JUNOS Internet Software VPNs Configuration Guide</i> .	You cannot include the <code>vrf-table-label</code> statement when configuring the 10-port E1 PIC, aggregated interfaces, Fast Ethernet 12-port and 48-port PIC, Gigabit Ethernet 4-port PIC, or Gigabit Ethernet intelligent queuing (IQ) PIC.  There is no restriction on CE router-to-PE router interfaces.
Configure the virtual loopback tunnel interface on routers equipped with a Tunnel PIC	All interfaces	See the guidelines in this section.	Router must be equipped with a Tunnel PIC.  There is no restriction on the type of core-facing interface used or CE router-to-PE router interface used.  You cannot configure a virtual loopback tunnel interface and the <code>vrf-table-label</code> statement at the same time.

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.

---

## Configure PIM Tunnels

---

PIM tunnels are enabled automatically on routers 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 router encapsulates packets destined for the Rendezvous Point (RP) router. The packets are encapsulated with a unicast header and are forwarded through a unicast tunnel to the RP. The RP then de-encapsulates the packets and transmits them through its multicast tree. To perform the encapsulation and de-encapsulation, the first-hop and RP routers must be equipped with Tunnel PICs.

The JUNOS software creates two interfaces to handle PIM tunnels:

pe—Encapsulates packets destined for the RP. This interface is present on the first-hop router.

pd—De-encapsulates packets at the RP. This interface is present on the RP.

## Configure an IPv6-over-IPv4 Tunnel

---

If you have a Tunnel PIC installed in your router, 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 end point.

IPv6-over-IPv4 tunnels are defined in RFC 2893, *Transition Mechanisms for IPv6 Hosts and Routers*. For information about configuring a unicast tunnel, see “Configure a Unicast Tunnel” on page 418. For an IPv6-over-IPv4 tunnel configuration example, see “Example: Configure an IPv6-over-IPv4 Tunnel” on page 426.

## Example: Configure 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;
      family iso;
    }
    unit 1 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.254;
      }
      family inet;
      family iso;
    }
  }
}
```

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 5.5.5.1/30;
      }
      family iso;
    }
    unit 1 {
      tunnel {
        source 192.168.4.18;
        destination 192.168.4.254;
      }
      family inet {
        address 6.6.6.100/30;
      }
      family iso;
    }
  }
}
```

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 5.1.1.1/30;
      }
      family iso;
      family mpls;
    }
  }
}

```

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

---

Configure a tunnel on both sides of the connection.

On Router 1:

```

[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 2:

```

[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;
      }
    }
  }
}

```

## Example: Configure 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;
    }
  }
}
```

