

Junos® OS

Tunnel and Encryption Services Interfaces User Guide for Routing Devices

Published
2025-06-22

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Junos® OS Tunnel and Encryption Services Interfaces User Guide for Routing Devices
Copyright © 2025 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | viii

1

Tunnel Services

Tunnel Services Overview | 2

Tunnel Services Overview | 2

Tunnel Interfaces on MX Series Routers with Line Cards (MPC7E through MPC11E) | 6

Dynamic Tunnels Overview | 14

Configuring Tunnel Interfaces | 15

Tunnel Interface Configuration on MX Series Routers Overview | 16

Configuring Tunnel Interfaces on an MX Series Router with a 16x10GE 3D MPC | 17

Configuring Tunnel Interfaces on MX Series Routers with the MPC3E | 19

Example: Configuring Tunnel Interfaces on the MPC3E | 20

Requirements for Configuration of Tunnel Interfaces on the MPC3E | 20

Ethernet Tunnel Configuration Overview | 20

Configuring a 20-Gigabit Ethernet Tunnel | 21

Configuring a Tunnel With Unspecified Bandwidth | 21

Configuring Tunnel Interfaces on MX Series Routers with MPC4E | 22

Configuring Tunnel Interfaces on MX Series Routers with MPC7E-MRATE/MPC7E-10G | 23

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC8E | 24

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC9E | 25

Configuring Tunnel Interfaces on MX Series Routers with MPC10E-10C and MPC10E-15C | 26

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC11E | 28

Configuring Tunnel Interfaces on MX Series Routers with MX10K-LC9600 | 29

Configuring Tunnel Interfaces on MX304 Devices | 30

Example: Configuring Tunnel Interfaces on a Gigabit Ethernet 40-Port DPC | 31

Example: Configuring Tunnel Interfaces on a 10-Gigabit Ethernet 4-Port DPC | 32

Configuring Tunnel Interfaces on MX 204 Routers | 33

Configuring Tunnel Interfaces on T4000 Routers | 34

Configuring Tunnel Interfaces on MX Series Routers and PTX Series Routers | 35

Understanding Tunnel Interfaces on MX Series Routers | 35

Understanding Tunnel Interfaces on PTX Series Routers | 37

Use Cases Implemented by Configuring Tunnels on MX Series and PTX Series Routers | 41

Configuring Tunnel Interfaces on ACX Series Routers | 43

Guidelines for Configuring Logical Tunnel Interfaces on ACX Series Routers | 43

Configuring Logical Tunnel Interfaces on ACX7K Series Routers | 46

Configuring Flexible Tunnel Interfaces | 49

Flexible Tunnel Interfaces Overview | 49

Configuring Flexible Tunnel Interfaces | 63

Configuring FTI on PE1 | 64

Verification | 66

Configuring a Flexible Tunnel Interface on an SRX Firewall | 68

| 70

Example: Configuring Flexible Tunnel Interfaces on MX Series Routers | 70

Requirements | 70

Overview | 71

Configuration | 71

Verification | 77

Configuring IP-IP Decapsulation by Tunnel Termination on FTI | 77

Configuring GRE Tunnel Interfaces | 79

Understanding Generic Routing Encapsulation | 80

Configuring Generic Routing Encapsulation Tunneling | 83

Configuring a GRE Tunnel Port | 84

Configuring Tunnels to Use Generic Routing Encapsulation | 84

GRE Keepalive Time Overview | 85

Configuring GRE Keepalive Time | 87

- Configuring Keepalive Time and Hold time for a GRE Tunnel Interface | 87

- Display GRE Keepalive Time Configuration | 88

- Display Keepalive Time Information on a GRE Tunnel Interface | 89

Enabling Fragmentation on GRE Tunnels | 91

Soft GRE Capability | 92

- Soft GRE Capability Overview | 92

- Configure Tunnel Attributes for GRE or UDP Dynamic Tunnels | 94

Configuring IP Tunnel Interfaces | 95

- Configuring IPv6-over-IPv4 Tunnels | 95

- Example: Configuring an IPv6-over-IPv4 Tunnel | 96

Filtering Unicast Packets Through Multicast Tunnel Interfaces | 97

- Configuring Unicast Tunnels | 97

- Examples: Configuring Unicast Tunnels | 103

- Restricting Tunnels to Multicast Traffic | 105

Connecting Logical Systems Using Logical Tunnel Interfaces | 106

- Configuring Logical Tunnel Interfaces | 107

- Guidelines for Configuring Logical Tunnels on MX Series Routers | 108

- Configuring Bridge Domain over Logical Tunnel Physical Interface | 110

- Configuring Recycle Bandwidth for Logical Tunnel Physical Interface | 111

- Layer 3 VPN Support over Logical Tunnel Interfaces | 112

- Example: Configuring Logical Tunnels | 114

- Configuring an Interface in the VRF Domain to Receive Multicast Traffic | 115

- Redundant Logical Tunnels Overview | 118

- Configuring Redundant Logical Tunnels | 121

- Configuring Single Link Targeting for Redundant Logical Tunnels | 122

- Configuring Minimum Active Links for Redundant Logical Tunnels | 122

Example: Configuring Redundant Logical Tunnels | 123

Requirements | 123

Overview | 123

Configuration | 125

Verification | 133

Configuring Layer 2 Ethernet Services over GRE Tunnel Interfaces | 137

Layer 2 Services over GRE Tunnel Interfaces on MX Series with MPCs | 138

Format of GRE Frames and Processing of GRE Interfaces for Layer 2 Ethernet Packets | 138

Guidelines for Configuring Layer 2 Ethernet Traffic Over GRE Tunnels | 139

Sample Scenarios of Configuring Layer 2 Ethernet Traffic Over GRE Tunnels | 141

Configuring Layer 2 Services over GRE Logical Interfaces in Bridge Domains | 142

Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains | 143

Requirements | 144

Overview | 144

Configuration | 144

Verification | 148

Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains with IPv6 Transport | 150

Requirements | 151

Overview | 151

Configuration | 152

Verification | 158

Configuring PIM Tunnels | 159

Facilitating VRF Table Lookup Using Virtual Loopback Tunnel Interfaces | 160

Configuring Virtual Loopback Tunnels for VRF Table Lookup | 160

Configuring Tunnel Interfaces for Routing Table Lookup | 162

Example: Configuring a Virtual Loopback Tunnel for VRF Table Lookup | 163

Example: Virtual Routing and Forwarding (VRF) and Service Configuration | 164

BGP Layer 3 VPN over IP-IP Tunnels Overview | 166

2

Encryption Services**Encryption Services Overview | 169****Configuring Encryption Interfaces | 169**

Configuring Encryption Interfaces | 170

Configuring Filters for Traffic Transiting the ES PIC | 172

Configuring an ES Tunnel Interface for a Layer 3 VPN | 179

Configuring ES PIC Redundancy | 179

Configuring IPsec Tunnel Redundancy | 180

3

Configuration Statements and Operational Commands**Junos CLI Reference Overview | 183**

About This Guide

Use this guide to configure and monitor tunneling, which encapsulates packets inside a transport protocol, providing a private, secure path through an otherwise public network.

1

CHAPTER

Tunnel Services

IN THIS CHAPTER

- [Tunnel Services Overview | 2](#)
 - [Configuring Tunnel Interfaces | 15](#)
 - [Configuring Tunnel Interfaces on MX Series Routers and PTX Series Routers | 35](#)
 - [Configuring Tunnel Interfaces on ACX Series Routers | 43](#)
 - [Configuring Flexible Tunnel Interfaces | 49](#)
 - [Configuring GRE Tunnel Interfaces | 79](#)
 - [Soft GRE Capability | 92](#)
 - [Configuring IP Tunnel Interfaces | 95](#)
 - [Filtering Unicast Packets Through Multicast Tunnel Interfaces | 97](#)
 - [Connecting Logical Systems Using Logical Tunnel Interfaces | 106](#)
 - [Configuring Layer 2 Ethernet Services over GRE Tunnel Interfaces | 137](#)
 - [Configuring PIM Tunnels | 159](#)
 - [Facilitating VRF Table Lookup Using Virtual Loopback Tunnel Interfaces | 160](#)
 - [BGP Layer 3 VPN over IP-IP Tunnels Overview | 166](#)
-

Tunnel Services Overview

IN THIS SECTION

- [Tunnel Services Overview | 2](#)
- [Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\) | 6](#)
- [Dynamic Tunnels Overview | 14](#)

Tunnel Services Overview

IN THIS SECTION

- [Platform-Specific Tunnel Interfaces Behavior | 6](#)

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 MPLS. If you have a Tunnel *Physical Interface Card* (PIC) installed in your M Series or T Series router, you can configure unicast, multicast, and logical tunnels.

For details on platform and Junos version support, see [Feature Explorer](#).

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 170](#). For information about VPNs, see the [Junos OS VPNs Library for Routing Devices](#). For information about MPLS, see the [MPLS Applications User Guide](#).

On SRX Series Firewalls, Generic Routing Encapsulation (GRE) and IP-IP tunnels use internal interfaces, gr-0/0/0 and ip-0/0/0, respectively. The Junos OS creates these interfaces at system bootup; they are not associated with physical interfaces.

The Juniper Networks Junos OS supports the tunnel types shown in the following table.

Table 1: Tunnel Interface Types

Interface	Description
gr-0/0/0	<p>Configurable generic routing encapsulation (GRE) interface. GRE allows the encapsulation of one routing protocol over another routing protocol.</p> <p>Within a router, packets are routed to this internal interface, where they are first encapsulated with a GRE packet and then re-encapsulated with another protocol packet to complete the GRE. The GRE interface is an internal interface only and is not associated with a physical interface. You must configure the interface for it to perform GRE.</p>
gre	Internally generated GRE interface. This interface is generated by the Junos OS to handle GRE. You cannot configure this interface.
ip-0/0/0	<p>Configurable IP-over-IP encapsulation (also called IP tunneling) interface. IP tunneling allows the encapsulation of one IP packet over another IP packet.</p> <p>Packets are routed to an internal interface where they are encapsulated with an IP packet and then forwarded to the encapsulating packet's destination address. The IP-IP interface is an internal interface only and is not associated with a physical interface. You must configure the interface for it to perform IP tunneling.</p>
ipip	Internally generated IP-over-IP interface. This interface is generated by the Junos OS to handle IP-over-IP encapsulation. It is not a configurable interface.
lt-0/0/0	<p>The lt interface on M Series and T Series routers supports configuration of logical systems—the capability to partition a single physical router into multiple logical devices that perform independent routing tasks.</p> <p>On SRX Series Firewalls, the lt interface is a configurable logical tunnel interface that interconnects logical systems. See the <i>Junos OS Logical Systems Configuration Guide for Security Devices</i>.</p>

Table 1: Tunnel Interface Types (Continued)

Interface	Description
mt-0/0/0	<p>Internally generated multicast tunnel interface. 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.</p> <p>Within a router, packets are routed to this internal interface for multicast filtering. The multicast tunnel interface is an internal interface only and is not associated with a physical interface. If your router has a Tunnel Services PIC, the Junos OS automatically configures one multicast tunnel interface (mt-) for each virtual private network (VPN) you configure. You do not need to configure multicast tunnel interfaces. However, you can configure properties on mt- interfaces, such as the <code>multicast-only</code> statement.</p>
mtun	<p>Internally generated multicast tunnel interface. This interface is generated by the Junos OS to handle multicast tunnel services. It is not a configurable interface.</p>
pd-0/0/0	<p>Configurable Protocol Independent Multicast (PIM) de-encapsulation interface. In PIM sparse mode, the first-hop router encapsulates packets destined for the rendezvous point router. 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.</p> <p>Within a router, packets are routed to this internal interface for de-encapsulation. The PIM de-encapsulation interface is an internal interface only and is not associated with a physical interface. You must configure the interface for it to perform PIM de-encapsulation.</p> <p>NOTE: On SRX Series Firewalls, this interface type is <code>ppd0</code>.</p>
pe-0/0/0	<p>Configurable PIM encapsulation interface. In PIM sparse mode, the first-hop router encapsulates packets destined for the rendezvous point router. 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.</p> <p>Within a router, packets are routed to this internal interface for encapsulation. The PIM encapsulation interface is an internal interface only and is not associated with a physical interface. You must configure the interface for it to perform PIM encapsulation.</p> <p>NOTE: On SRX Series Firewalls, this interface type is <code>ppe0</code>.</p>

Table 1: Tunnel Interface Types (Continued)

Interface	Description
pimd	Internally generated PIM de-encapsulation interface. This interface is generated by the Junos OS to handle PIM de-encapsulation. It is not a configurable interface.
pime	Internally generated PIM encapsulation interface. This interface is generated by the Junos OS to handle PIM encapsulation. It is not a configurable interface.
vt-0/0/0	<p>Configurable virtual loopback tunnel interface. Facilitates VRF table lookup based on MPLS labels. This interface type is supported on M Series and T Series routers, but not on SRX Series Firewalls.</p> <p>To configure a virtual loopback tunnel 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 for route lookup.</p>

Starting in Junos OS Release 15.1, you can configure Layer 2 Ethernet services over GRE interfaces (*gr-fpc/pic/port* to use GRE encapsulation). To enable Layer 2 Ethernet packets to be terminated on GRE tunnels, you must configure the bridge domain protocol family on the *gr-* interfaces and associate the *gr-* interfaces with the bridge domain. You must configure the GRE interfaces as core-facing interfaces, and they must be access or trunk interfaces. To configure the bridge domain family on *gr-* interfaces, include the family bridge statement at the [edit interfaces *gr-fpc/pic/port* unit *logical-unit-number*] hierarchy level. To associate the *gr-* interface with a bridge domain, include the interface *gr-fpc/pic/port* statement at the [edit routing-instances *routing-instance-name* bridge-domains *bridge-domain-name*] hierarchy level. You can associate GRE interfaces in a bridge domain with the corresponding VLAN ID or list of VLAN IDs in a bridge domain by including the *vlan-id* (all | none | number) statement or the *vlan-id-list* [*vlan-id-numbers*] statement at the [edit bridge-domains *bridge-domain-name*] hierarchy level. The VLAN IDs configured for the bridge domain must match with the VLAN IDs that you configure for GRE interfaces by using the *vlan-id* (all | none | number) statement or the *vlan-id-list* [*vlan-id-numbers*] statement at the [edit interfaces *gr-fpc/pic/port* unit *logical-unit-number*] hierarchy level. You can also configure GRE interfaces within a bridge domain associated with a virtual switch instance. Layer 2 Ethernet packets over GRE tunnels are also supported with the GRE key option. The *gre-key* match condition allows a user to match against the GRE key field, which is an optional field in GRE encapsulated packets. The key can be matched as a single key value, a range of key values, or both.



NOTE: Starting in Junos OS Release 16.1, Layer 2 Port mirroring to a remote collector over a GRE interface is supported.

Platform-Specific Tunnel Interfaces Behavior

Platform	Difference
MX304	Starting with Junos OS Release 24.4R1, tunnel interfaces are deleted or created upon PFE offline or online respectively. When the anchor PFE associated with IPv4-over-IPv6 or IPv6-over-IPv4 dynamic tunnels goes offline then the tunnel is also deleted.

SEE ALSO

[GRE Keepalive Time Overview | 85](#)

[Configuring Unicast Tunnels | 97](#)

[Restricting Tunnels to Multicast Traffic | 105](#)

[Configuring Tunnel Interfaces on T4000 Routers | 34](#)

Tunnel Interfaces on MX Series Routers with Line Cards (MPC7E through MPC11E)

IN THIS SECTION

- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-MRATE | 7](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-10G | 8](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC8E | 8](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC9E | 9](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-10C | 10](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-15C | 10](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC11E | 11](#)
- [Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX10K-LC9600 | 11](#)

MPC7E-10G, MPC7E-MRATE, MX2K-MPC8E, and MX2K-MPC9E support a total of four inline tunnel interfaces per MPC, one per PIC. You can create a set of tunnel interfaces per PIC slot up to a maximum of four slots (from 0 through 3) on MX Series routers with these MPCs.

MPC10E-15C supports three inline tunnel interfaces per MPC, one per PIC, whereas MPC10E-10C supports two inline tunnel interfaces per MPC, one per PIC. On MX Series routers with MPC10E-15C, you can

create a set of tunnel interfaces per PIC slot up to a maximum of three slots (from 0 through 2). And, on MX Series routers with MPC10E-10C, you can create a set of tunnel interfaces per PIC slot up to a maximum of two slots (0 and 1).

MX2K-MPC11E supports 8 inline tunnel interfaces per MPC, one per PIC. On MX Series routers with MX2K-MPC11E, you can create a set of tunnel interfaces per PIC slot up to a maximum of eight slots (from 0 through 7). These PICs are referred to as pseudo tunnel PICs. You create tunnel interfaces on MX Series routers with MPC7E-10G, MPC7E-MRATE, MX2K-MPC8E, MX2K-MPC9E, MPC10E-15C, MPC10E-10C, and MX2K-MPC11E by including the following statements at the **[edit chassis]** hierarchy level:

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-MRATE

The tunnel bandwidth for MPC7E-MRATE is 1–120Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 120Gbps.

[Table 2 on page 7](#) shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MPC7-MRATE .

Table 2: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-MRATE

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	120Gbps	PFE0	120Gbps	240Gbps

Table 2: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-MRATE (Continued)

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC1	120Gbps			
PIC2	120Gbps	PFE1	120Gbps	240Gbps
PIC3	120Gbps			

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-10G

The tunnel bandwidth for MPC7E-10G is 1–120Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 120Gbps.

[Table 3 on page 8](#) shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MPC7E-10G.

Table 3: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC7E-10G

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	120Gbps	PFE0	120Gbps	200Gbps
PIC1	120Gbps			
PIC2	120Gbps	PFE1	120Gbps	200Gbps
PIC3	120Gbps			

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC8E

The tunnel bandwidth for MX2K-MPC8E is 1–120Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 120Gbps.

Table 4 on page 9 shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MX2K-MPC8E.

Table 4: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC8E

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	Packet Forwarding Engine Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	120Gbps	PFE0	120Gbps	240Gbps
PIC1	120Gbps	PFE1	120Gbps	240Gbps
PIC2	120Gbps	PFE2	120Gbps	240Gbps
PIC3	120Gbps	PFE3	120Gbps	240Gbps

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC9E

The tunnel bandwidth for MX2K-MPC9E is 1–200Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 200Gbps.

Table 5 on page 9 shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MX2K-MPC9E.

Table 5: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC9E

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	200Gbps	PFE0	200Gbps	400Gbps
PIC1	200Gbps	PFE1	200Gbps	400Gbps
PIC2	200Gbps	PFE2	200Gbps	400Gbps
PIC3	200Gbps	PFE3	200Gbps	400Gbps

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-10C

The tunnel bandwidth for MPC10E-10C is 1–400Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 400Gbps.

[Table 6 on page 10](#) shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MPC10E-10C.

Table 6: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-10C.

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	Packet Forwarding Engine Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	250Gbps	PFE0	250Gbps	500Gbps
PIC1	250Gbps	PFE1	250Gbps	500Gbps

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-15C

The tunnel bandwidth for MPC10E-15C is 1–400Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 400Gbps.

[Table 7 on page 10](#) shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MPC10E-15C.

Table 7: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MPC10E-15C.

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	Packet Forwarding Engine Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	250Gbps	PFE0	250Gbps	500Gbps
PIC1	250Gbps	PFE1	250Gbps	500Gbps
PIC2	250Gbps	PFE2	250Gbps	500Gbps

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC11E

The tunnel bandwidth for MX2K-MPC11E is 1–400Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 400Gbps.

Table 8 on page 11 shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MX2K-MPC11E .

Table 8: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX2K-MPC11E

Pseudo Tunnel PIC	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	250Gbps	PFE0	250Gbps	500Gbps
PIC1	250Gbps	PFE1	250Gbps	500Gbps
PIC2	250Gbps	PFE2	250Gbps	500Gbps
PIC3	250Gbps	PFE3	250Gbps	500Gbps
PIC4	250Gbps	PFE4	250Gbps	500Gbps
PIC5	250Gbps	PFE5	250Gbps	500Gbps
PIC6	250Gbps	PFE6	250Gbps	500Gbps
PIC7	250Gbps	PFE7	250Gbps	500Gbps



NOTE: An unspecified tunnel services bandwidth value in the configuration for MPC10E-10C, MPC10E-15C, and MX2K-MPC11E results in a value larger than the maximum tunnel bandwidth per PFE in certain traffic conditions.

Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX10K-LC9600

The tunnel bandwidth for MX10K-LC9600 is 1–400Gbps with an increment of 1Gbps. However, if you do not specify the bandwidth in the configuration, it is set to 400Gbps.

Table 9 on page 12 shows the mapping between the tunnel bandwidth and the Packet Forwarding Engines for MX10K-LC9600.

Table 9: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX10K-LC9600

Pseudo Tunnel PIC	Tunnel Port	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC0	0	200Gbps	PFE0	200Gbps	800Gbps
	1	200Gbps	PFE0	200Gbps	
	2	200Gbps	PFE1	200Gbps	
	3	200Gbps	PFE1	200Gbps	
PIC1	0	200Gbps	PFE2	200Gbps	800Gbps
	1	200Gbps	PFE2	200Gbps	
	2	200Gbps	PFE3	200Gbps	
	3	200Gbps	PFE3	200Gbps	
PIC2	0	200Gbps	PFE4	200Gbps	800Gbps
	1	200Gbps	PFE4	200Gbps	
	2	200Gbps	PFE5	200Gbps	
	3	200Gbps	PFE5	200Gbps	
PIC3	0	200Gbps	PFE6	200Gbps	800Gbps
	1	200Gbps	PFE6	200Gbps	

Table 9: Packet Forwarding Engine Mapping and Tunnel Bandwidth for MX10K-LC9600 (Continued)

Pseudo Tunnel PIC	Tunnel Port	Maximum Bandwidth per Tunnel PIC	PFE Mapping	Maximum Tunnel Bandwidth per PFE	Maximum PFE Bandwidth
PIC4	2	200Gbps	PFE7	200Gbps	800Gbps
	3	200Gbps	PFE7	200Gbps	
	0	200Gbps	PFE8	200Gbps	
	1	200Gbps	PFE8	200Gbps	
PIC5	2	200Gbps	PFE9	200Gbps	800Gbps
	3	200Gbps	PFE9	200Gbps	
	0	200Gbps	PFE10	200Gbps	
	1	200Gbps	PFE10	200Gbps	
PIC5	2	200Gbps	PFE11	200Gbps	800Gbps
	3	200Gbps	PFE11	200Gbps	
	0	200Gbps	PFE10	200Gbps	
	1	200Gbps	PFE10	200Gbps	

SEE ALSO*tunnel-services**bandwidth*

Dynamic Tunnels Overview

A VPN that travels through a non-MPLS network requires a GRE tunnel. This tunnel can be either a static tunnel or a dynamic tunnel. A static tunnel is configured manually between two PE routers. A dynamic tunnel is configured using BGP route resolution.

When a router receives a VPN route that resolves over a BGP next hop that does not have an MPLS path, a GRE tunnel can be created dynamically, allowing the VPN traffic to be forwarded to that route. Only GRE IPv4 tunnels are supported.

To configure a dynamic tunnel between two PE routers, include the `dynamic-tunnels` statement:

```
dynamic-tunnels tunnel-name {
    destination-networks prefix;
    source-address address;
}
```

You can configure this statement at the following hierarchy levels:

- [edit routing-options]
- [edit routing-instances *routing-instance-name* routing-options]
- [edit logical-systems *logical-system-name* routing-options]
- [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options]

SEE ALSO

[*dynamic-tunnels*](#)

[Junos OS Routing Protocols Library](#)

[Junos OS VPNs Library for Routing Devices](#)

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
16.1	Starting in Junos OS Release 16.1, Layer 2 Port mirroring to a remote collector over a GRE interface is supported.

15.1

Starting in Junos OS Release 15.1, you can configure Layer 2 Ethernet services over GRE interfaces (*gr-fpc/pic/port* to use GRE encapsulation).

Configuring Tunnel Interfaces

IN THIS SECTION

- [Tunnel Interface Configuration on MX Series Routers Overview | 16](#)
- [Configuring Tunnel Interfaces on an MX Series Router with a 16x10GE 3D MPC | 17](#)
- [Configuring Tunnel Interfaces on MX Series Routers with the MPC3E | 19](#)
- [Example: Configuring Tunnel Interfaces on the MPC3E | 20](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MPC4E | 22](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MPC7E-MRATE/MPC7E-10G | 23](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC8E | 24](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC9E | 25](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MPC10E-10C and MPC10E-15C | 26](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC11E | 28](#)
- [Configuring Tunnel Interfaces on MX Series Routers with MX10K-LC9600 | 29](#)
- [Configuring Tunnel Interfaces on MX304 Devices | 30](#)
- [Example: Configuring Tunnel Interfaces on a Gigabit Ethernet 40-Port DPC | 31](#)
- [Example: Configuring Tunnel Interfaces on a 10-Gigabit Ethernet 4-Port DPC | 32](#)
- [Configuring Tunnel Interfaces on MX 204 Routers | 33](#)
- [Configuring Tunnel Interfaces on T4000 Routers | 34](#)

Tunnel Interface Configuration on MX Series Routers Overview

Because MX Series routers do not support Tunnel Services PICs, you create tunnel interfaces on MX Series routers by including the following statements at the [edit chassis] hierarchy level:

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth (1g | 10g | 20g | 30g | 40g | 50g | 60g | 70g | 80g | 90g | 100g);
    }
  }
}
```

Where:

`fpc slot-number` is the slot number of the DPC, MPC, or MIC. On the MX80 router, possible values are 0 and 1. On other MX Series routers, if two SCBs are installed, the range is 0 through 11. If three SCBs are installed, the range is 0 through 5 and 7 through 11.

`pic number` is the slot number of the PIC. On MX80 routers, if the FPC is 0, the PIC number can only be 0. If the FPC is 1, the PIC range is 0 through 3. For all other MX Series routers, the range is 0 through 3.

`bandwidth (1g | 10g | 20g | 30g | 40g | 50g | 60g | 70g | 80g | 90g | 100g)` is the maximum amount of bandwidth, in gigabits, that is available for tunnel traffic on each Packet Forwarding Engine. For MPCs and MICs, this bandwidth is not reserved for tunnel traffic and can be shared by the network interfaces. For DPCs, this bandwidth is reserved and cannot be shared by the network interfaces.



NOTE: When you use MPCs and MICs, tunnel interfaces are soft interfaces and allow as much traffic as the forwarding-path allows, so it is advantageous to set up tunnel services without artificially limiting traffic by use of the bandwidth option. However, you *must* specify bandwidth when configuring tunnel services for MX Series routers with DPCs or FPCs. The GRE key option is not supported on the tunnel interfaces for DPCs on MX960 routers.

If you specify a bandwidth that is not compatible, tunnel services are not activated. For example, you cannot specify a bandwidth of 1 Gbps for a Packet Forwarding Engine on a 10-Gigabit Ethernet 4-port DPC.

When you configure tunnel interfaces on the Packet Forwarding Engine of a 10-Gigabit Ethernet 4-port DPC, the Ethernet interfaces for that port are removed from service and are no longer visible in the command-line interface (CLI). The Packet Forwarding Engine of a 10-Gigabit Ethernet 4-port DPC

supports either tunnel interfaces or Ethernet interfaces, but not both. Each port on the 10-Gigabit Ethernet 4-port DPC includes two LEDs, one for tunnel services and one for Ethernet services, to indicate which type of service is being used. On the Gigabit Ethernet 40-port DPC, you can configure both tunnel and Ethernet interfaces at the same time.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#). The bandwidth that you specify determines the port number of the tunnel interfaces that are created. When you specify a bandwidth of 1g, the port number is always 10. When you specify any other bandwidth, the port number is always 0.



NOTE: When the tunnel bandwidth is unspecified in the Routing Engine CLI, the maximum tunnel bandwidth for an MPC3E is 60G.



NOTE: You cannot configure ingress queueing and tunnel services on the same MPC because doing so causes PFE forwarding to stop. You can configure and use each feature separately.

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Configuring Tunnel Interfaces on an MX Series Router with a 16x10GE 3D MPC

MX960, MX480, and M240 routers support the 16-port 10-Gigabit Ethernet MPC (16x10GE 3D MPC) fixed configuration Field Replaceable Unit (FRU). Each Packet Forwarding Engine on a 16x10GE MPC can support a full-duplex 10Gbps tunnel without losing line-rate capacity. For example, a full-duplex 10Gbps tunnel can be hosted on a 10-Gigabit-Ethernet port, while two other 10-Gigabit-Ethernet ports on the same PFE can concurrently forward line-rate traffic.

To configure an MPC and its corresponding Packet Forwarding Engine to use tunneling services, include the `tunnel-services` statement at the `[edit chassis fpc slot-number pic pic-number]` hierarchy level. The Junos

OS creates tunnel interfaces **gr-*fpc/pic/port*0**, **vt-*fpc/pic/port*0**, and so on. You also configure the amount of bandwidth reserved for tunnel services.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth 10g;
    }
  }
}
```

fpc slot-number is the slot number of the MPC. If two SCBs are installed, the range is 0 through 11. If three SCBs are installed, the range is 0 through 5 and 7 through 11.

pic number is the number of the Packet Forwarding Engine on the MPC. The range is 0 through 3.

bandwidth 10g is the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine.

In the following example, you create tunnel interfaces on Packet Forwarding Engine 0 of MPC 4 with 10 Gbps of bandwidth reserved for tunnel traffic. With this configuration, the tunnel interfaces created are **gr-4/0/0**, **pe-4/0/0**, **pd-4/0/0**, **vt-4/0/0**, and so on.

```
[edit chassis]
fpc 4 pic 0 {
  tunnel-services {
    bandwidth 10g;
  }
}
```

SEE ALSO

[Configuring Junos OS to Run a Specific Network Services Mode in MX Series Routers](#)

Configuring Tunnel Interfaces on MX Series Routers with the MPC3E

Because the MX Series routers do not support Tunnel Services PICs, you create tunnel interfaces on MX Series routers by including the following statements at the [edit chassis] hierarchy level:

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth (1g | 10g | 20g | 40g);
    }
  }
}
```

`fpc slot-number` is the slot number of the DPC, MPC, or MIC. On the MX80 router, the range is 0 through 1. On other MX series routers, if two SCBs are installed, the range is 0 through 11. If three SCBs are installed, the range is 0 through 5 and 7 through 11.

The `pic number` On MX80 routers, if the FPC is 0, the PIC number can only be 0. If the FPC is 1, the PIC range is 0 through 3. For all other MX series routers, the range is 0 through 3.

`bandwidth (1g | 10g | 20g | 40g)` is the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine.



NOTE: When you use MPCs and MICs, tunnel interfaces are soft interfaces and allow as much traffic as the forwarding-path allows, so it is advantageous to setup tunnel services without artificially limiting traffic by use of the `bandwidth` option. However, you *must* specify `bandwidth` when configuring tunnel services for MX Series routers with DPCs or FPCs.

1g indicates that 1 gigabit per second of bandwidth is reserved for tunnel traffic.

10g indicates that 10 gigabits per second of bandwidth is reserved for tunnel traffic.

20g indicates that 20 gigabits per second of bandwidth is reserved for tunnel traffic.

40g indicates that 40 gigabits per second of bandwidth is reserved for tunnel traffic.

If you specify a bandwidth that is not compatible, tunnel services are not activated. For example, you cannot specify a bandwidth of 1 Gbps for a Packet Forwarding Engine on a 10-Gigabit Ethernet 4-port DPC.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#). The bandwidth that you specify determines the port number of the tunnel interfaces that are created. When you specify a bandwidth of 1g, the port number is always 10. When you specify any other bandwidth, the port number is always 0.

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Example: Configuring Tunnel Interfaces on the MPC3E

IN THIS SECTION

- [Requirements for Configuration of Tunnel Interfaces on the MPC3E | 20](#)
- [Ethernet Tunnel Configuration Overview | 20](#)
- [Configuring a 20-Gigabit Ethernet Tunnel | 21](#)
- [Configuring a Tunnel With Unspecified Bandwidth | 21](#)

Requirements for Configuration of Tunnel Interfaces on the MPC3E

This example requires MX Series routers with the MPC3E.

Ethernet Tunnel Configuration Overview

MX Series routers do not support Tunnel Services PICs. However, you can create one set of tunnel interfaces per pic slot up to a maximum of 4 slots from 0-3 on MX Series routers with the MPC3E.

To configure the tunnels, include the **tunnel-services** statement and an optional bandwidth of (**1g | 10g | 20g | 30g | 40g**) at the **[edit chassis]** hierarchy level.



NOTE: When no tunnel bandwidth is specified, the tunnel interface can have a maximum bandwidth of up to 60Gbps.



NOTE: A MIC need not be plugged in to the MPC3E to configure a tunnel interface.

Configuring a 20-Gigabit Ethernet Tunnel

IN THIS SECTION

- [Procedure | 21](#)

Procedure

Step-by-Step Procedure

In the following example, you create tunnel interfaces on PIC-slot 1 of MPC 0 with 20 gigabit per second of bandwidth reserved for tunnel traffic. With this configuration, the tunnel interfaces created are **gr-0/1/0**, **pe-0/1/0**, **pd-0/1/0**, **vt-0/1/0**, and so on.

1. To create a 20 gigabit per second tunnel interface, use the following configuration:

```
[edit chassis]
fpc 0 pic 1 {
  tunnel-services {
    bandwidth 20g;
  }
}
```

Configuring a Tunnel With Unspecified Bandwidth

IN THIS SECTION

- [Procedure | 22](#)

Procedure

Step-by-Step Procedure

In the following example, you create a tunnel interface on PIC-slot 3 of MPC 0 with no bandwidth specified. The tunnel traffic can carry up to a maximum of 60Gbps depending on other traffic through the packet forwarding engine. With this configuration, the tunnel interfaces created are **gr-0/3/0**, **pe-0/3/0**, **pd-0/3/0**, **vt-0/3/0**, and so on.

1. To create a tunnel interface with no bandwidth specification, use the following configuration:

```
[edit chassis]
fpc 0 pic 3 {
    tunnel-services;
}
```

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Configuring Tunnel Interfaces on MX Series Routers with MPC4E

MX Series routers do not support Tunnel Services PICs. However, you can create a set of tunnel interfaces per PIC slot up to a maximum of four slots from 0 through 3 on MX Series routers with MPC4E.

To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth of (1g | 10g | 20g | 30g | 40g) at the `[edit chassis]` hierarchy level. When no tunnel bandwidth is specified, the tunnel interface can have a maximum bandwidth of up to 60 Gbps.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#). The bandwidth that you specify determines the port number of the tunnel interfaces that are created. When you specify a bandwidth of 1g, the port number is always 10. When you specify any other bandwidth, the port number is always 0.

In the following example, you create tunnel interfaces on **PIC 1** of **MPC 4** with 40 Gbps of bandwidth reserved for tunnel traffic. `fpc slot-number` is the slot number of the MPC. In this configuration, the tunnel interfaces created are **gr-4/1/1**, **pe-4/1/1**, **pd-4/1/1**, **vt-4/1/1**, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 4 pic 1 {
  tunnel-services {
    bandwidth 40g;
  }
}
```

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Configuring Tunnel Interfaces on MX Series Routers with MPC7E-MRATE/MPC7E-10G

MPCs support a total of four inline tunnels per MPC, one per PIC. You can create a set of tunnel interfaces per PIC slot up to a maximum of four slots from 0 through 3

To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth of 1 Gbps through 120Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 120 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MPC 5 with 40 Gbps of bandwidth reserved for tunnel traffic. *fpc slot-number* is the slot number of the MPC. In this configuration, the tunnel interfaces created are gr-5/1/1, pe-5/1/1, pd-5/1/1, vt-5/1/1, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

SEE ALSO

[Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\) | 6](#)

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC8E

MX2K-MPC8E support a total of four inline tunnels per MPC, one per PIC. You can create a set of tunnel interfaces per PIC slot up to a maximum of four slots from 0 through 3.

To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth of 1–120Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 120 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth;
    }
  }
}
```


To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MPC 5 with 40 Gbps of bandwidth reserved for tunnel traffic. *fpc slot-number* is the slot number of the MPC. In this configuration, the tunnel interfaces created are gr-5/1/1, pe-5/1/1, pd-5/1/1, vt-5/1/1, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

SEE ALSO

[Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\) | 6](#)

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC9E

MX2K-MPC9E supports a total of four inline tunnels per MPC, one per PIC. You can create a set of tunnel interfaces per PIC slot up to a maximum of four slots from 0 through 3.

To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth in the range 1–200Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 200 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

```
}
}
```

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MPC 5 with 40 Gbps of bandwidth reserved for tunnel traffic. *fpc slot-number* is the slot number of the MPC. In this configuration, the tunnel interfaces created are gr-5/1/1, pe-5/1/1, pd-5/1/1, vt-5/1/1, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

SEE ALSO

[Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\) | 6](#)

Configuring Tunnel Interfaces on MX Series Routers with MPC10E-10C and MPC10E-15C

SUMMARY

MPC10E-10C supports two inline tunnel interfaces per MPC. You can create a set of tunnel interfaces per PIC slot up to a maximum of two slots- 0 and 1. MPC10E-10C MPC10E-15C supports three inline per PIC slot up to a maximum of two slots- 0 and 1. MPC10E-10C MPC10E-15C supports three inline

tunnel interfaces per MPC. You can create a set of tunnel interfaces per PIC slot up to a maximum of three slots from 0 through 2.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MPC with 40 Gbps of bandwidth reserved for tunnel traffic. `fpc slot-number` is the slot number of the MPC. In this configuration, the tunnel interfaces created are gr-5/1/0, pe-5/1/0, pd-5/1/1, lt-5/1/0, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

SEE ALSO

| [Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\)](#) | 6

Configuring Tunnel Interfaces on MX Series Routers with MX2K-MPC11E

SUMMARY

MX2K-MPC11E supports a total of eight inline tunnels per MPC, one per PIC. You can create a set of tunnel interfaces per PIC slot up to a maximum of eight slots from 0 through 7.

To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth in the range 1–400Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 400 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MPC 5 with 40 Gbps of bandwidth reserved for tunnel traffic. `fpc slot-number` is the slot number of the MPC. In this configuration, the tunnel interfaces created are `gr-5/1/0`, `pe-5/1/0`, `pd-5/1/0`, `lt-5/1/0`, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

```
}
}
```

SEE ALSO

[Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\)](#) | 6

Configuring Tunnel Interfaces on MX Series Routers with MX10K-LC9600

MX10K-LC9600 MPC supports a total of six inline tunnels per MPC. Each tunnel PIC can have up to 4 tunnel interfaces.

To configure the tunnel interfaces, include the `tunnel-services` statement and bandwidth upto 400Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 400 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      tunnel-port {
        bandwidth ;
      }
    }
  }
}
```



NOTE: When you configure tunnel bandwidth of 1G, there will not be any configuration commit failures. If you newly commit an inline tunnel services configuration with the bandwidth 1G as part of MX10K-LC9600 reboot, an error message appears and no tunnel IFDs are created.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MX10K-LC9600 and tunnel port number 1 with 40 Gbps of bandwidth reserved for tunnel traffic. *fpc slot-number* is the slot number of the MPC. In this configuration, the tunnel interfaces created are gr-5/1/1, pe-5/1/1, pd-5/1/1, vt-5/1/1, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      tunnel-port 1 {
        bandwidth 40g;
      }
    }
  }
}
```

Configuring Tunnel Interfaces on MX304 Devices

MX304 supports a total of three inline tunnels per MPC.

To configure the tunnel interfaces, include the `tunnel-services` statement and bandwidth upto 400Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 400 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      tunnel-port {
        bandwidth ;
      }
    }
  }
}
```



NOTE: When you configure tunnel bandwidth of 1Gbps, the configuration commit does not fail. If you newly commit an inline tunnel service configuration with the bandwidth of 1Gbps or apply it as part of MX304 reboot, tunnel interfaces are not created. However, no error message is displayed.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 1 of MX304 and tunnel port number 1 with 40 Gbps of bandwidth reserved for tunnel traffic. `fpc slot-number` is the slot number of the MPC. In this configuration, the tunnel interfaces created are `gr-5/1/1`, `pe-5/1/1`, `pd-5/1/1`, `vt-5/1/1`, and so on.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 5 {
  pic 1 {
    tunnel-services {
      tunnel-port 1 {
        bandwidth 40g;
      }
    }
  }
}
```

Example: Configuring Tunnel Interfaces on a Gigabit Ethernet 40-Port DPC

The following example shows how to create tunnel interfaces on Packet Forwarding Engine 1 of DPC 4 with 1 Gbps of bandwidth reserved for tunnel services. On a Gigabit Ethernet 40-port DPC, tunnel interfaces coexist with Ethernet interfaces. With this configuration, the Gigabit Ethernet interfaces are **ge-4/1/0** through **ge-4/1/9**. The tunnel interfaces created are **gr-4/1/10**, **pe-4/1/10**, **pd-4/1/10**, **vt-4/1/10** and so on.

```
[edit chassis]
fpc 4 pic 1 {
  tunnel-services {
```

```

        bandwidth 1g;
    }
}

```

SEE ALSO

[Configuring the Junos OS to Support ILMI for Cell Relay Encapsulation on an ATM2 IQ PIC](#)

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Example: Configuring Tunnel Interfaces on a 10-Gigabit Ethernet 4-Port DPC

In this example, you create tunnel interfaces on Packet Forwarding Engine 0 of DPC 4 with 10 Gbps of bandwidth reserved for tunnel traffic. Ethernet and tunnel interfaces cannot coexist on the same Packet Forwarding Engine of a 10-Gigabit Ethernet 4-port DPC. With this configuration, the tunnel interfaces created are **gr-4/0/0**, **pe-4/0/0**, **pd-4/0/0**, **vt-4/0/0** and so on.

```

[edit chassis]
fpc 4 pic 0 {
    tunnel-services {
        bandwidth 10g;
    }
}

```

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Configuring Tunnel Interfaces on MX 204 Routers

The MX204 router is a fixed-configuration router, and supports one fixed Routing Engine. It has two PICs and contains a total of twelve fixed ports, in two groups of four and eight, respectively. The set of four ports (referred to as the PIC 0 ports) are rate selectable and can be configured at 10-Gbps (by using a breakout cable), 40-Gbps, or 100-Gbps speed. However, not all the ports support all the three speeds. The set of eight ports (referred to as PIC 1 ports) operate at a fixed speed of 10-Gbps.

The MX204 router supports two inline tunnels - one per PIC. To configure the tunnel interfaces, include the `tunnel-services` statement and an optional bandwidth of 1 Gbps through 200 Gbps at the `[edit chassis fpc fpc-slot pic number]` hierarchy level. If you do not specify the tunnel bandwidth then, the tunnel interface can have a maximum bandwidth of up to 200 Gbps.

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth ;
    }
  }
}
```

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the [CLI Explorer](#).

In the following example, you create tunnel interfaces on PIC 0 of MPC 0 with 40 Gbps of bandwidth reserved for tunnel traffic. `fpc slot-number` is the slot number of the MPC.

To create a 40-Gbps tunnel interface, use the following configuration:

```
[edit chassis]
fpc 0 {
  pic 0 {
    tunnel-services {
      bandwidth 40g;
    }
  }
}
```

SEE ALSO

[Tunnel Interfaces on MX Series Routers with Line Cards \(MPC7E through MPC11E\) | 6](#)

Configuring Tunnel Interfaces on T4000 Routers

To create tunnel interfaces on a T4000 Core Router, include the following statements at the [edit chassis] hierarchy level:

```
[edit chassis]
fpc slot-number {
  pic number {
    tunnel-services {
      bandwidth bandwidth-value;
    }
  }
}
```

`fpc slot-number` denotes the slot number of the FPC. On the T4000 router, the range is 0 through 7.


NOTE:

- This applies only to the T4000 Type 5 FPC. If any other type of FPC is configured in this slot, this configuration is ignored and no tunnel physical interface is created.
- When you use Type 5 FPCs, the tunnel interfaces are soft interfaces and allow as much traffic as the forwarding-path allows. So, it is advantageous to setup tunnel services without artificially limiting traffic by setting the `bandwidth` statement.

`pic number` on the T4000 router is 0 or 1.

`bandwidth bandwidth-value` is the amount of bandwidth to reserve for the tunnel traffic on each Packet Forwarding Engine. The bandwidth value accepted includes every multiple of 10g up to 100g.

If you specify a bandwidth that is not compatible, tunnel services are not activated. For example, you cannot specify a bandwidth of 1 Gbps for a Packet Forwarding Engine on a 100-Gigabit Ethernet PIC with CFP.

To verify that the tunnel interfaces have been created, issue the `show interfaces terse operational mode` command. For more information, see the *Junos Interfaces Command Reference*.

SEE ALSO

bandwidth (Tunnel Services)

tunnel-services (Chassis)

Configuring Tunnel Interfaces on MX Series Routers and PTX Series Routers

IN THIS SECTION

- [Understanding Tunnel Interfaces on MX Series Routers | 35](#)
- [Understanding Tunnel Interfaces on PTX Series Routers | 37](#)
- [Use Cases Implemented by Configuring Tunnels on MX Series and PTX Series Routers | 41](#)

Understanding Tunnel Interfaces on MX Series Routers

- Interface (gr-, lt-, and ip-)-based tunnels —You can configure interface-based tunnels when the bandwidth profile enforcement is required. The GRE tunnels support IPv4, IPv6, MPLS, ISO, and Ethernet payload, whereas the IP-IP tunnels support IPv4 and IPv6 payload.

You can configure interface-based tunnels to implement:

- Tunneling over an IP network with bandwidth enforcement per tunnel. For example, toward the remote site.
- Steering for DDoS cleaning.
- Mirroring to remote destinations.

During the GRE process, after the GRE header is added to the packet, the packet is looped back to the same Packet Forwarding Engine for a second lookup. The packet enters the ingress pipeline back through the loopback interface, which has a limited bandwidth of 400G. The encapsulated packet is then forwarded to the destination.

GRE tunnel de-encapsulation is implemented by inline tunnel termination and does not use the loopback stream. However, the overall packet performance of the Packet Forwarding Engine is

impacted due to extra fabric hops as the traffic flow switches from one Packet Forwarding Engine to another.

- Flexible tunnel interfaces (FTIs)—You can configure FTIs when the bandwidth profile enforcement is not required. FTIs support both IPv4 and IPv6 payload. You can configure FTIs to implement:
 - Mirroring to remote destinations.
 - IP fabrics with IP-IP overlays.
 - Steering for DDoS cleaning.

The packet performance is reduced due to the extra lookup after encapsulation and de-encapsulation.

- Dynamic tunnels—You can configure dynamic tunnels to design data center gateways.

In the dynamic tunnels implementation with loopback avoidance, the packet performance is reduced due to the extra lookup after an encapsulation or de-encapsulation.

- Firewall filter-based tunnels— These tunnels support:
 - GRE and GRE in UDP encapsulation.
 - GRE, IP-IP, and GRE in UDP de-encapsulation.

You can configure firewall filter-based tunnels to design data center gateways.

You can configure GRE-based encapsulation and de-encapsulation using a firewall filter action without using a tunnel interface. Encapsulation and de-encapsulation happens at the Packet Forwarding Engine which processes the filter. MX Series routers support firewall filters at:

- Interface level on input (executed on the ingress Packet Forwarding Engine)
- Interface level on output (executed on the egress Packet Forwarding Engine)
- Forwarding table level (either before route lookup or after route lookup). In both cases, the filter is executed on the ingress PFE

In this scenario, the packet performance is reduced due to the extra lookup after encapsulation. In case of de-encapsulation, the packet performance is reduced due to the filter configuration.

Understanding Tunnel Interfaces on PTX Series Routers

IN THIS SECTION

- [Tunnel Interfaces on PTX10001-36MR, PTX10004, PTX10008, and PTX10016-Overview | 37](#)
- [Tunnel Interfaces on PTX1000, PTX5000, and PTX10002-50C-Overview | 40](#)

You can configuring tunnel interfaces to implement different features on the PTX Series routers. The following sections provide an overview of the features implemented on different PTX Series routers.

Tunnel Interfaces on PTX10001-36MR, PTX10004, PTX10008, and PTX10016-Overview

This section provides information about configuring tunnel interfaces to implement different features on PTX10001-36MR, PTX10004, PTX10008, and PTX10016 routers with Junos OS Evolved.

- Flexible tunnel interfaces (FTIs)—You can configure FTI-based tunnels to implement:
 - Steering for DDoS cleaning.
 - De-encapsulation for all dynamic tunnels use cases.

These tunnels support GRE, UDP, and IP-IP encapsulation and de-encapsulation options. The reduction in the packet performance depends on the encapsulation option. Encapsulation supports flattened next-hop topology..

You can configure GRE tunnels on flexible tunnel interfaces. When you enable the `tunnel-termination` statement at the `[edit interfaces fti0 unit unit-number family (inet | inet6)]` hierarchy, tunnels are terminated on the WAN interface before any other actions, such as sampling, port mirroring, or filtering, are applied.

- Flexible tunnel interfaces (FTIs) through a loopback—You can configure FTIs to implement mirroring to remote destinations.

On an FTI, after tunnel encapsulation, traffic is sent into the loopback interface (on the ingress Packet Forwarding Engine) and later to the ultimate destination. The destinations could include those behind segment routing-traffic engineered (SR-TE) next hops. The loopback interface has a limited bandwidth of 400G.

You can configure GRE/IP-in-IP/UDP tunnel encapsulation on FTIs using the loopback interface. You can configure encapsulation by using the command `tunnel encapsulation (gre|ipip|udp) source address destination address` at the `[edit interfaces fti0 unit unit-number hierarchy`. You must consider the following points while configuring this feature:

- Adding `tunnel-termination` makes the tunnel decap-only tunnel and encapsulation is disabled.
- Specifying both the source and destination address is mandatory when you do not configure the command.
- Configuring a variable prefix mask on the source address is not allowed

When you configure a firewall filter for de-encapsulation, the firewall filter de-encapsulates the packets based on the configured match conditions and action. Then de-encapsulated packets are re-circulated back to ingress block to do inner header lookup and forwarded accordingly. However, tunnel termination is completed in a single pass of packet processing, thus providing performance improvement over filter based process.

To enable termination only mode, where the tunnel is unidirectional, you can configure `tunnel-termination` at the `[edit interfaces fti0 unit unit-number tunnel encapsulation (gre|ipip|udp)]` hierarchy on the FTI.

```
[edit interfaces fti0]
    unit unit-number {
        tunnel {
            encapsulation < gre | ipip | udp > {
                tunnel-termination;
                key key;
                source {
                    address source_ip;
                }
                destination {
                    address dst_ip;
                }
            }
            tunnel-routing-instance {
                routing-instance instance;
            }
        }
    }
family < inet | inet6 | mpls >;
```

```
}
}
```

```
[edit interfaces fti0]
    unit unit-number {
        tunnel {
            encapsulation < gre | ipip | udp > {
                tunnel-termination;
                source {
                    address device-loopback;
                }
            }
        }
    }
    family < inet | inet6 | mpls >;
}
```

The source address can be excluded which indicates that the tunnel terminates on the configured destination address. The optional `tunnel-routing-instance` indicates that after de-encapsulation, the inner IP lookup is done with VRF ID corresponding to routing-instance.

Packets to be de-encapsulated are processed in the source lookup unit. The search key contains the IPv4 or IPv6 destination address and optionally L3VPN address if the tunnel need not be terminated on all interfaces. If the first lookup is successful, no more source lookup is needed and tunnel is terminated. If second lookup is needed on the source address, the source lookup unit does another lookup using source address and first lookup result as the key. If the second lookup is successful, the tunnel is terminated.

When you enable the `tunnel-termination` statement at the `[edit interfaces fti0 unit unit-number]` hierarchy, tunnels are terminated on the WAN interface before any other actions, such as sampling, port mirroring, or filtering, are applied.

To enable tunnel termination on the incoming interface configure `tunnel-termination` at the `[edit interfaces et fpc/pic/port unit unit-number]`

```
{edit interfaces et-fpc/pic/port}
    unit unit-number {
        family inet {
            tunnel-termination;
        }
        family inet6 {
            tunnel-termination;
        }
    }
}
```

```

    }
  }
}

```

- Dynamic tunnels— You can configure dynamic tunnels to design:
 - IP fabrics.
 - IP overlays.
 - Data center gateways.

These tunnels support IP-IP encapsulation.

PTX Series routers with Junos OS Evolved, do not support dynamic tunnels for de-encapsulation.. Instead, you can use static FTI tunnels for de-encapsulation, without specifying the destination address. The tunnels are configured with the de-encapsulation-only option.

You can configure next-hop-based dynamic UDP tunnels, also known as MPLS-over-UDP tunnels. Junos OS dynamically creates next hops to resolve the tunnel destination route. You can also use policy control to resolve MPLS-over-UDP tunnels over select IP prefixes. Because when the next-hops are enabled by default, the MPLS-over-UDP feature provides a scaling advantage for the number of IP tunnels supported on the router.

Tunnel Interfaces on PTX1000, PTX5000, and PTX10002-50C-Overview

This section provides information about configuring tunnel interfaces to implement different features on PTX1000, PTX5000, and PTX10002-50C routers with Junos OS Evolved.

- Flexible tunnel interfaces (FTIs)—You can configure FTIs to implement IP fabric. We prefer this option when we need to reach tunnel destinations through IP. These tunnels support:
 - GRE, UDP, and IP-IP encapsulation options.
 - UDP and IP-IP de-encapsulation options.

The packet performance is reduced due to the extra lookup after encapsulation and de-encapsulation.

- Dynamic tunnels—You can configure dynamic tunnels to design data center gateway. The packet performance is reduced due to extra lookup after encapsulation and de-encapsulation.
- Firewall filter-based tunnels—You can configure firewall filter-based tunnels to implement egress peering engineering (EPE). The packet performance is reduced during encapsulation due to the extra lookup.

- Mirroring to remote destination— You can implement tunneling of mirrored packets to remote destinations. The packet performance is reduced during encapsulation due to the extra lookup.

Use Cases Implemented by Configuring Tunnels on MX Series and PTX Series Routers

IN THIS SECTION

- [Port Mirroring to Remote Destinations | 41](#)
- [Data Center Gateways | 42](#)

This section provides information about some of the use cases of configuring tunnel interfaces to implement different features (use cases) on the MX series and PTX series routers.

Port Mirroring to Remote Destinations

You can use port mirroring for traffic analysis on routers and switches that, unlike hubs, do not broadcast packets to every port on the destination device. Port mirroring sends copies of all packets or policy-based sample packets to local or remote analyzers where you can monitor and analyze the data.

For an MX Series router configured as a provider edge (PE) router on the customer-facing edge of a service provider network, you can apply a Layer 2 port-mirroring firewall filter at the ingress and egress points to mirror the traffic between the MX Series router and customer edge (CE) devices, such as routers and Ethernet switches.

On MX Series routers, you can mirror traffic arriving at tunnel interfaces to multiple destinations. You specify two or more destinations in a next-hop group, define a firewall filter that references the next-hop group as the filter action, and then apply the filter to a logical tunnel interface (lt-) or virtual tunnel interfaces (vt-) on the MX Series router.

See [Configuring Port Mirroring](#) and [Configuring Port Mirroring for Remote Destinations](#).

When the data path traverses a flexible tunnel interface (FTI)-based tunnel, the router sends the output packet with tunnel encapsulation. You can set up a configuration that mirrors the original packet as well as the packet with all encapsulations as it exits the interface out.

To enable mirroring based on a filter installed on the FTI:

- You mark packets for mirroring using the policy action at the FTI. The router typically uses the policy action to select the egress rewrite rule, but in this case, it uses the policy action to mark interesting packets with an internal policy attribute, without any special rewrite rule configured.
- You have the software intercept packets that match the specific policy on the egress WAN side and initiate the `l2-mirror` action. Packets are reported with Layer 2 header information, including tunnel encapsulation.

See *Example: Configuring Local Port Mirroring on PTX Routers* No Link Title and No Link Title.

Data Center Gateways

Data center gateways interconnect the Internet or enterprise VPNs on one side and virtual machines hosted on servers on the other side. Overlay transport technologies such as MPLS-over-GRE or MPLS-over-UDP are part of a data center design. Host routes are communicated to the data center gateway from the SDN controller and imported into virtual routing and forwarding (VRF) or Internet routing contexts.. The data center servers are reachable through next-hop-based dynamic tunnels. Tunnels are established on servers in the BGP route protocol next-hop resolution process.

As described in, *RFC 5549*, IPv4 traffic is tunneled from CPE devices to IPv4-over-IPv6 gateways. These gateways are announced to CPE devices through anycast addresses. The gateway devices then create dynamic IPv4-over-IPv6 tunnels to remote CPE devices and advertise IPv4 aggregate routes to steer traffic. Route reflectors with programmable interfaces inject the tunnel information into the network. The route reflectors are connected through internal BGP (IBGP) to gateway routers, which advertise the IPv4 addresses of host routes with IPv6 addresses as the next hop.

The MPLS-over-UDP tunnel is handled as follows:

1. After an MPLS-over-UDP tunnel is configured, a tunnel destination mask route with a tunnel composite next hop is created for the tunnel in the `inet.3` routing table. This IP tunnel route is withdrawn only when the dynamic tunnel configuration is deleted.

The tunnel composite next-hop attributes include the following:

- When the Layer 3 VPN composite next hop is disabled—Source and destination address, encapsulation string, and VPN label.
- When the Layer 3 VPN composite next hop and per-prefix VPN label allocation are enabled—Source address, destination address, and encapsulation string.
- When the Layer 3 VPN composite next hop is enabled and the per-prefix VPN label allocation is disabled—Source address, destination address, and encapsulation string. The route in this case is added to the other VRF instance table with a secondary route.

2. The provider edge (PE) devices are interconnected using an IBGP session. The IBGP route next hop to a remote BGP neighbor is the protocol next hop, which is resolved using the tunnel mask route with the tunnel next hop.
3. After the protocol next hop is resolved over the tunnel composite next hop, indirect next hops with forwarding next hops are created.
4. The tunnel composite next hop is used to forward the next hops of the indirect next hops.

See [Example: Configuring Next-Hop-Based MPLS-Over-UDP Dynamic Tunnels](#) and [Example: Configuring Next-Hop-Based IP-Over-IP Dynamic Tunnels](#)

Configuring Tunnel Interfaces on ACX Series Routers

IN THIS SECTION

- [Guidelines for Configuring Logical Tunnel Interfaces on ACX Series Routers | 43](#)
- [Configuring Logical Tunnel Interfaces on ACX7K Series Routers | 46](#)

Guidelines for Configuring Logical Tunnel Interfaces on ACX Series Routers

Observe the following guidelines while configuring logical tunnel (lt-) interfaces on ACX Series routers:

- You can use a logical tunnel interface to connect only bridge domains and pseudowires.
- Logical tunnel interfaces cannot interconnect the following links:
 - Pseudowire and a routing instance (Pseudowire terminating on a VRF)
 - Two routing instances
 - VPLS instance and a routing instance
 - Two VPLS instances
 - Two Bridge domains

- Bridge domain and a VPLS instance
- Only one logical tunnel (physical interface) per bandwidth type (1 Gbps or 10 Gbps) can be configured on ACX routers. However, you can specify up to two logical tunnel interfaces (one with 1 Gb bandwidth and another with 10 Gb bandwidth) on ACX routes.
- Guaranteed bandwidth for logical tunnels is 1 Gbps and certain platforms support up to an additional 10 Gbps bandwidth. All the services configured using logical tunnel interfaces share this bandwidth.

The bandwidth configured on the logical tunnel interface is shared between upstream and downstream traffic on that interface. The effective bandwidth available for the service is half the configured bandwidth.

- Multiple logical tunnel interfaces to enable configuration of separate services on each logical interface to obtain increased bandwidth for each individual interface separately or the bundling of individual logical tunnel interfaces is not supported.
- You can configure Ethernet VLAN, Ethernet CCC, VLAN bridge on Ethernet interfaces, and VLAN on circuit cross-connects (CCC) as encapsulation types on logical tunnel interfaces. Other encapsulation types such as Ethernet, VLAN, Ethernet VPLS, or VLAN VPLS are not supported.
- When the encapsulation configured on the logical interface units is one of the supported types such as Ethernet VLAN or VLAN bridge, you can enable only bridge domains or CCC protocols on logical tunnel interfaces. Other address families or protocols such as IPv4, IPv6, MPLS, or OSPF are not supported.
- Classifier, rewrite and ingress policer configuration are supported on logical tunnel interfaces. Fixed, BA-based, and multifield classifiers are supported on the Lt- interfaces at the physical interface-level.

802.1p, 802.1ad, TOS and DSCP based BA classifiers are supported. Remarking rules can be configured at the port level on the LT interface. 802.1p, 802.1ad, TOS and DSCP fields in the packet can be rewritten in the LT interface. Ingress policers are supported.

Simple, Single-rate tricolor marking (srTCM), two-rate tricolor marking (trTCM) policers are supported. Egress policers are not supported.

- Default classifiers do not work properly when Lt- interfaces are configured on non-Ethernet PICs.
- Port-level queuing is supported; up to eight queues per Lt- interface are supported. These eight queues are shared between the upstream and downstream traffic traversing through the Lt- interface. If the configured bandwidth on the Lt- interface is not adequate for the upstream and downstream traffic of the services configured on the interface, a failure occurs with traffic propagation because multiple Lt- interfaces are not supported.
- Eight forwarding classes (0-7) are mapped to the eight queues based on the global system configuration. The remainder of the scheduler configuration, buffer-size, transmit-rate, shaping-rate, priority and WRED or drop profiles maps can be configured on the Lt- interface queues.

- The following firewall filter types are supported on lt- interfaces:

- Logical interface-level filters
- Bridge family filters
- CCC family filters

All firewall configurations are supported. The scaling limitation with such filters is the same as the existing firewall filter restrictions.

- OAM is not supported on lt- interfaces.
- Similar to other physical interfaces, the number of logical interfaces that can be supported on logical tunnel physical interfaces is 30.
- When a bridge domain is configured with a VLAN ID (bridge domain has normalized VLANs), the difference in behavior between MX and ACX Series routers is that the MX router does not match the user-vlan-id in output filter, whereas the ACX router matches the user-vlan-id specified in the output filter.
- If the logical tunnel interface is created using non Ethernet PICs, then default classifier is not bound to the interface.

To create logical tunnel interfaces and the bandwidth in gigabits per second to reserve for tunnel services, include the `tunnel-services bandwidth (1g | 10g)` statement at the `[edit chassis fpc slot-number pfe pfe-number core core-number channel channel-number]` hierarchy level:

```
[edit interfaces]
  lt-fpc/pic/port {
    unit logical-unit-number {
      encapsulation encapsulation;
      peer-unit unit-number;    # peering logical system unit number
      dlci dlci-number;
      family (inet | inet6 | iso | mpls);
    }
  }
```

The ACX5048 and ACX5096 routers support ethernet-vpls and vlan-vpls encapsulations. These encapsulations are supported only on logical tunnel interface and are required for configuring hierarchical VPLS.

You can use any unused physical port on the ACX5048 and ACX5096 routers to create a logical tunnel interface as shown below:

```
user@host# edit chassis
fpc 0 {
  pic 0 {
    tunnel-services {
      port port-number;
    }
  }
}
```

The following sample configuration allows you to encapsulate vlan-ccc to vlan-vpls using LT interface in ACX5048 and ACX5096 routers:

```
user@host# edit interfaces
lt-0/0/1 {
  unit 0 {
    encapsulation vlan-ccc;
    vlan-id 1;
    peer-unit 1;
  }
  unit 1 {
    encapsulation vlan-vpls;
    vlan-id 1;
    peer-unit 0;
  }
}
```

Configuring Logical Tunnel Interfaces on ACX7K Series Routers

IN THIS SECTION

- [Configuring Logical Tunnel Physical Interface on ACX7K Series Routers | 47](#)

Starting with Junos Evolved OS Release 24.2R1, ACX7K Series routers support logical tunnel physical interface (IFD) configuration for Layer 2 services (BD).

- Support for logical tunnel physical interface which includes:
 - Logical tunnel interface physical interface level configuration
 - Support stitchings of two disjoint services through the logical tunnel interface
 - Support SNMP on logical tunnel interface
- Support logical tunnel interface (LT ifl) and Bridge Domain which includes:
 - Creation of logical tunnel interface, each unit of logical tunnel interface with a peer-unit configuration as a mandatory parameter. If unit X is configured unit Y as peer-unit, then unit Y must have unit X as a peer-unit.
 - Support encapsulation vlan-bridge on logical tunnel interface
 - Support encapsulation ethernet-bridge on logical tunnel interface
 - Support receiver and transmitter statistics on logical tunnel interface. The statistics of the receiver and transmitter of logical tunnel interface must work same as other logical interface statistics.
 - Support Layer 2 flooding on logical tunnel interface
 - Support MAC learning. This support includes addition of static MAC on logical tunnel interface, dynamic MAC learning on logical tunnel interface, and all MAC events and notifications handling.

Configuring Logical Tunnel Physical Interface on ACX7K Series Routers

To create logical tunnel interfaces and the bandwidth in Gbps to reserve for tunnel services, include the tunnel-services bandwidth *value* statement at the [edit chassis fpc *slot-number* | feb slot *slot-number* pfe *pfe-number* core *core-number* channel *channel-number*] hierarchy level.

The following sample configuration allows you to configure the logical tunnel on FPC based systems:

```
user@host# edit chassis
fpc 0 {
  pfe 0 {
    core 0 {
      channel 0 {
        tunnel-services {
          bandwidth 10g;
        }
      }
    }
  }
}
```

```
    }
  }
}
```

For example, you can use the following sample configuration to create lt-0/0/0:2 with a bandwidth of 10Gbps on an FPC-based system with a bandwidth of 10Gbps:

```
set chassis fpc 0 pfe 0 core 0 channel 2 tunnel-services bandwidth 10g
```

The following sample configuration allows you to configure the logical tunnel on FEB based systems:

```
user@host# edit chassis
feb slot 0 {
  pfe 0 {
    core 0 {
      channel 0 {
        tunnel-services {
          bandwidth 10g;
        }
      }
    }
  }
}
```

For example, to create lt-0/0/0:3 on an FEB-based system with a bandwidth of 10 Gbps, you can use this configuration:

```
set chassis feb 0 pfe 0 core 0 channel 3 tunnel-services bandwidth 10g
```

Create logical tunnel interface and encapsulate the logical interface for service provider style bridging configuration.

```
set chassis fpc 0 pfe 0 core 0 channel 3 tunnel-services bandwidth 10G
set interfaces lt-0/0/0:3 flexible-vlan-tagging
set interfaces lt-0/0/0:3 unit 0 peer-unit 1 encapsulation vlan-bridge vlan-id 100
set interfaces lt-0/0/0:3 unit 1 peer-unit 0 encapsulation vlan-bridge vlan-id 100
```


Configuring Flexible Tunnel Interfaces

IN THIS SECTION

- [Flexible Tunnel Interfaces Overview | 49](#)
- [Configuring Flexible Tunnel Interfaces | 63](#)
- [Configuring a Flexible Tunnel Interface on an SRX Firewall | 68](#)
- [| 70](#)
- [Example: Configuring Flexible Tunnel Interfaces on MX Series Routers | 70](#)
- [Configuring IP-IP Decapsulation by Tunnel Termination on FTI | 77](#)

Flexible Tunnel Interfaces Overview

IN THIS SECTION

- [Flexible Tunnel Interfaces on MX Series Routers and SRX Series Firewalls | 49](#)
- [Flexible Tunnel Interfaces on PTX Series Routers and QFX Series Switches | 50](#)
- [Platform-Specific FTI Behavior | 52](#)
- [MPLS Support for FTI tunnels on PTX Series Routers | 53](#)
- [Flexible Tunnel Interfaces on ACX Series Routers | 54](#)
- [MPLS Support for FTI tunnels on ACX Series Routers | 59](#)
- [Benefits of Flexible Tunnel Interfaces | 61](#)
- [Limitations of Flexible Tunnel Interfaces | 62](#)

A flexible tunnel interface (FTI) is a type of logical tunnel interface that uses static routing and BGP protocols to exchange routes over a tunnel that connects endpoints to routers.

Flexible Tunnel Interfaces on MX Series Routers and SRX Series Firewalls

FTIs have the following features on MX Series routers and SRX Series firewalls:

- FTI supports only VXLAN encapsulation with Layer 2 pseudo-headers.
- FTI is used between a router and a server hosting multiple virtual machines, or between routers in two different data centers.
- FTI can be configured as port-mirror destinations.
- FTI support logical interface statistics streaming.

In the VXLAN encapsulation process, the Layer 2 address is populated with “pseudo” source (source MAC: 00-00-5E-00-52-00) and destination (destination MAC: 00-00-5E-00-52-01) MAC addresses without VLAN tagging; however, these addresses are ignored when the packets reach the remote endpoint. The remote endpoint is identified by the destination IP address and a specified destination UDP port number. The corresponding FTI on the remote endpoint is identified by the virtual network identifier (VNI) value, the source IP address of the tunnel, and the destination UDP port number. All of these values can be configured on an FTI with VXLAN encapsulation.

Figure 1: FTIs Connecting Remote Devices to a Virtual Private Cloud

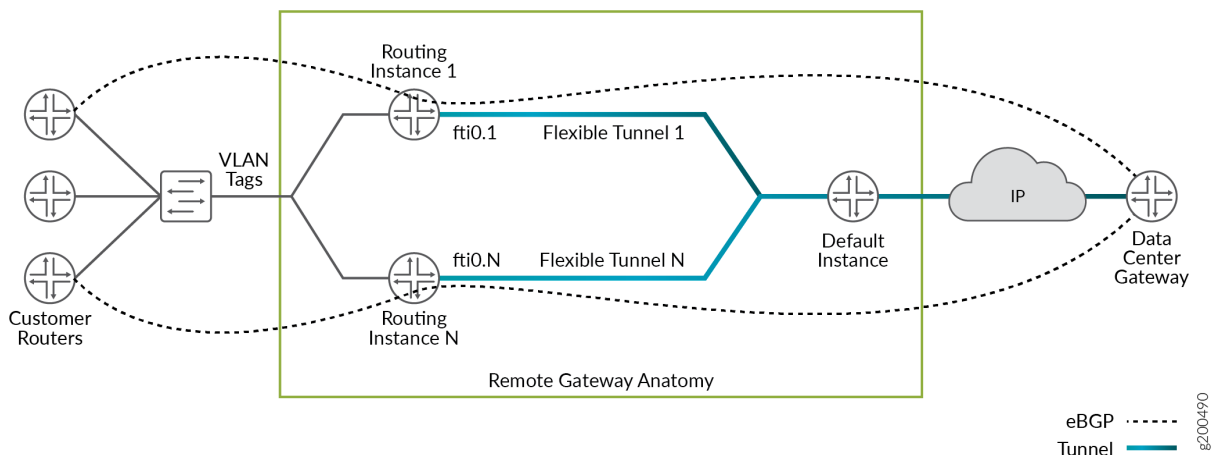


Figure 1 on page 50 illustrates how an FTI works to provide connectivity into a virtual private cloud from a remote location. Individual flexible tunnels (1 through M) are provisioned for every customer. The customer-facing logical interface and the corresponding FTIs are configured to operate in one routing instance. The FTI uses BGP protocols (eBGP and iBGP) to carry packets from the customer device to the remote gateway and vice versa.

Flexible Tunnel Interfaces on PTX Series Routers and QFX Series Switches

Some PTX Series routers and QFX Series switches support FTIs. For details on platform and Junos version support, see [Feature Explorer](#). FTI support on PTX and QFX switches include the following features:

- FTI is supported in releases starting Junos OS Release 19.3R1.
- FTI supports only UDP encapsulation.
- FTI can be initiated at any place in the MPLS tunnel: MPLS transit, ingress, egress, and PHP.
- FTI with UDP encapsulation supports the following payloads:
 - IPV4 inside IPV4 UDP packet
 - IPV6 inside IPV4 UDP packet
 - MPLS inside IPV4 UDP packet
 - ISO inside IPV4 UDP packet

FTI with UDP encapsulation supports the following features and functionality:

- MPLS link protection and node-link protection.
- Manual configuration of RSVP bandwidth.
- BFD support for liveliness detection, excluding BFD over LDP and RSVP.
- Support for the following protocols:
 - BGP
 - RSVP
 - LDP
 - OSPF
 - ISIS
- Static routes.
- FTI logical interface statistics.
- MTU configuration on FTI and fragmentation of payload before entering the tunnel.
- Underlay can be Aggregated Ethernet or regular interface, and can be tagged sub-interface or regular Layer 3 interfaces.
- Overlay and underlay ECMP.

Platform-Specific FTI Behavior

Platform	Difference
PTX Routers with Junos OS Evolved	Output firewall filter on FTI is not supported.

To configure an FTI interface with UDP encapsulation, include the `udp` statement at the `[edit interfaces fti0 unit unit tunnel encapsulation]` hierarchy level.

For example:

```
[edit interfaces]
fti0 {
  unit unit_number {
    tunnel {
      encapsulation udp {
        source {
          address ipv4_address;
        }
        destination {
          address ipv4_address;
        }
      }
    }
    family inet {
      destination-udp-port udp port ;
    }
    family inet6 {
      destination-udp-port udp port ;
    }
    family mpls {
      destination-udp-port udp port ;
    }
    family iso {
      destination-udp-port udp port ;
    }
  }
}
```

MPLS Support for FTI tunnels on PTX Series Routers

Starting In Junos OS Evolved Release 21.4R1, you can configure MPLS protocols over FTI tunnels, thereby transporting MPLS packets over IP networks which does not support MPLS.

In Junos OS Evolved Release 21.4R1, generic routing encapsulation (GRE) and UDP tunnels support MPLS protocol for IPv4 and IPv6 traffic. You can configure encapsulation and decapsulation for the GRE and UDP tunnels.

The following features are supported :

- Encapsulation and decapsulation for IPv4 and IPv6 traffic
- UDP port number configuration
- MPLS node-link protection
- Ingress, egress, PHP, and transit roles for LSP
- Ping and traceroute support in ingress, egress, PHP, and transit roles for LSP
- Overlay and underlay ECMP
- Manual configuration of RSVP bandwidth.
- MPLS services
 - L3VPN
 - 6VPE
 - L2 circuit
 - BGP-LU with per nexhop or prefix label
- Routing instance
- Class-of-service (CoS) including the configuration of rewrite rules and classifiers
- MTU configuration and fragmentation of payload
- BFD support for liveliness detection.
- Jvision

The following features and functionality are not supported:

- MPLS link protection
- RSVP bandwidth Inheritance based on next hop to tunnel destination for FTI interfaces

- TTL propagation.
- Class-of-service on tunnel endpoints .
- FT-over-FT resolution .
- FT destination IP should be reachable through IGP and not BGP (no indirect next hop). The reachability should be through an IPV4 route and not through an LSP.
- Path MTU discovery .

To allow the MPLS traffic on the UDP tunnels include the `mpls port-number` statement at the `[edit forwarding-options tunnels udp port-profile profile-name]` hierarchy level. To allow the MPLS traffic on the GRE tunnels, include the `mpls` statement at the `[edit interfaces fti0 unit unit family]` hierarchy.

For example:

```
[edit forwarding-options]
  tunnels {
    udp {
      port-profile p1 {
        inet <port num>
        inet6 <port num>
        mpls <port num>
        iso <port num>
      }
    }
  }
```

Flexible Tunnel Interfaces on ACX Series Routers

Starting in Junos OS Evolved Release 24.1R1, you can configure encapsulation by using the `tunnel encapsulation gre source address destination address` command at the `[edit interfaces fti unit unit]` hierarchy level. For details on platform and Junos version support, see [Feature Explorer](#).

The following features are supported:

- FTI interface based GRE encapsulation and de-encapsulation mode
- inet, inte6, iso payloads
- Both IPv4 and IPv6 as overlay
- Both IPv4 and IPv6 as underlay

- BFD, OSPF, ISIS, and static route
- Underlay and overlay ECMP
- FTI IFL statistics
- MTU configuration on FTI.
- TTL configuration on FTI.
- Host ping
- FTI link up or down status based on tunnel end point reachability
- ECMP of FTI and regular interfaces
- Input Filter support on FTI at de-encapsulation side.

The following features and functionality are not supported:

- Tunnel-termination only mode
- Path MTU discovery for both IPv4 and IPv6 encapsulation
- Anti-spoofing at tunnel decapsulation for the inner source IP
- Flexible-vlan-tagging
- Tunnel destination reachability over other tunnel
- MTU exception generation when FTI IFF MTU value is high and underlay IFF MTU is low
- Output filter applied on tunnel underlay interface for tunnelled packet as well as payload due to data path limitation
- Input filter applied on NNI interface at tunnel de-encapsulation node for tunnelled packet as well as payload due to data path limitation
- FTI tunnel along with dynamic next hop tunnel for the same tunnel configuration.
- FTI IFF disable and enable
- IP Fragmentation at tunnel start point and end point
- Tunnel encapsulation stats is not supported for MPLS encapsulated packet sent over a tunnel with both tunnel and MPLS stats enabled. Only MPLS stats is supported.
- You can configure tunnel encapsulation or de-encapsulation stats using the `set system packet-forwarding-options tunnel encap-stats-enable` and `set system packet-forwarding-options tunnel decap-stats-enable` statements. When you use the CLI (set/delete/deactivate), datapath restart occurs to associate or disassociate counters with the tunnel.

To configure an FTI interface with GRE encapsulation, include the `gre` statement at the `[edit interfaces fti0 unit unit-number tunnel encapsulation]` hierarchy level.

For example:

```
[edit interfaces]
fti0 {
  unit unit-number{
    tunnel {
      encapsulation gre {

        source {
          address ipv4/ipv6_address;
        }
        destination {
          address ipv4/ipv6_address;
        }
        tunnel-routing-instance {
          routing-instance instance name;
        }

        bypass-loopback;
        ttl ttl-value;

      }
    }
    family inet {
      address ip_address ;
    }
    family inet6 {
      address ip_address;
    }
    family mpls;
    family iso;
  }
}
```

Starting in Junos OS Evolved Release 24.2R1, ACX series support FTI with UDP encapsulation and de-encapsulation. For details on platform and Junos version support, see [Feature Explorer](#).

The following features are supported:

- FTI with UDP supports the following payloads:
 - IPv4 inside IPv4 UDP packet

- IPv6 inside IPv4 UDP packet
- MPLS inside IPv4 UDP packet
- ISO inside IPv4 UDP packet
- IPv4 inside IPv6 UDP packet
- IPv6 inside IPv6 UDP packet
- MPLS inside IPv6 UDP packet
- ISO inside IPv6 UDP packet
- Support for the following protocols:
 - BGP
 - BFD
 - OSPF
 - ISIS
- Static routes.
- FTI logical interface statistics.
- MTU configuration on FTI.
- TTL configuration on FTI.
- Overlay and underlay ECMP.
- `bypass-loopback` configuration and `payload-port-profile profile name` configuration is mandatory.

The following features and functionality are not supported:

- Tunnel-termination only mode
- Path MTU discovery for both IPv4 and IPv6 encapsulation
- Anti-spoofing at tunnel decapsulation for the inner source IP
- Flexible-vlan-tagging
- Tunnel destination reachability over other tunnel
- MTU exception generation when FTI IFF MTU value is high and underlay IFF MTU is low
- Output filter applied on tunnel underlay interface for tunnelled packet as well as payload due to data path limitation

- Input filter applied on NNI interface at tunnel de-encapsulation node for tunnelled packet as well as payload due to data path limitation
- FTI tunnel along with dynamic next hop tunnel for the same tunnel configuration.
- FTI IFF disable and enable
- IP Fragmentation at tunnel start point and end point
- Tunnel encapsulation stats is not supported for MPLS encapsulated packet sent over a tunnel with both tunnel and MPLS stats enabled. Only MPLS stats is supported.
- You can configure tunnel encapsulation or de-encapsulation stats using the `set system packet-forwarding-options tunnel encap-stats-enable` and `set system packet-forwarding-options tunnel decap-stats-enable` statements. When you use the CLI (`set/delete/deactivate`), datapath restart occurs to associate or disassociate counters with the tunnel.

To configure an FTI interface with UDP encapsulation, include the `udp` statement at the `[edit interfaces fti0 unit unit tunnel encapsulation]` hierarchy level.

For example:

```
[edit interfaces]
fti0 {
  unit unit-number{
    tunnel {
      encapsulation udp {

        source {
          address ipv4/ipv6_address;
        }
        destination {
          address ipv4/ipv6_address;
        }
        tunnel-routing-instance {
          routing-instance instance name;
        }
        bypass-loopback;
        payload-port-profile profile name{
          inet port num;
          inet6 port num;
          mpls port num;
          iso port num;
        }
      }
    }
  }
}
```

```

        ttl t1l-value;

    }
}
family inet {
    address ip_address ;
}
family inet6 {
    address ip_address;
}
family mpls;
family iso;
}
}

```

MPLS Support for FTI tunnels on ACX Series Routers

Starting In Junos OS Evolved Release 24.2R1, you can configure MPLS protocols over FTI tunnels on ACX series routers, thereby transporting MPLS packets over IP networks which does not support MPLS.

In Junos OS Evolved Release 24.2R1, generic routing encapsulation (GRE) and UDP tunnels support MPLS protocol for IPv4 and IPv6 traffic on ACX series routers. You can configure encapsulation and decapsulation for the GRE and UDP tunnels. For details on platform and Junos version support, see [Feature Explorer](#).

The following features are supported :

- Encapsulation and decapsulation for IPv4 and IPv6 traffic
- UDP port number configuration
- Ping and traceroute support in ingress, egress, PHP, and transit roles for LSP
- Overlay and underlay ECMP
- LDP, RSVP, static LSP, and BGP protocols with encapsulation and decapsulation
- Support different tunnel termination scenarios, MPLSoGRE or MPLSoUDP tunnel can start at any of the following cases:
 - MPLS Label Edge Router (LER)
 - MPLS Label Switch Router (LSR)
 - MPLS PHP

- MPLS Egress PE
- MPLS services
 - L3VPN
 - 6VPE
 - 6PE

The following features and functionality are not supported:

- Tunnel-termination only mode
- Path MTU discovery for both IPv4 and IPv6 encapsulation
- Anti-spoofing at tunnel decapsulation for the inner source IP
- Flexible-vlan-tagging
- Tunnel destination reachability over other tunnel
- MTU exception generation when FTI IFF MTU value is high and underlay IFF MTU is low
- Output filter applied on tunnel underlay interface for tunnelled packet as well as payload due to data path limitation
- Input filter applied on NNI interface at tunnel de-encapsulation node for tunnelled packet as well as payload due to data path limitation
- FTI tunnel along with dynamic next hop tunnel for the same tunnel configuration.
- FTI IFF disable and enable
- IP Fragmentation at tunnel start point and end point
- Tunnel encapsulation stats is not supported for MPLS encapsulated packet sent over a tunnel with both tunnel and MPLS stats enabled. Only MPLS stats is supported.
- You can configure tunnel encapsulation or de-encapsulation stats using the `set system packet-forwarding-options tunnel encap-stats-enable` and `set system packet-forwarding-options tunnel decap-stats-enable` statements. When you use the CLI (`set/delete/deactivate`), datapath restart occurs to associate or disassociate counters with the tunnel.

To configure MPLS traffic on GRE or UDP tunnel include the `mpls` statement at the `[edit interfaces fti0 unit unit family]` hierarchy.

```
[edit interfaces]
fti0 {
```

```

unit unit-number{
    tunnel {
        encapsulation (gre | udp) {

source {
            address ipv4/ipv6_address;
        }
        destination {
            address ipv4/ipv6_address;
        }
        tunnel-routing-instance {
            routing-instance instance name;
        }
        bypass-loopback;
        payload-port-profile profile name{
            inet port num;
            inet6 port num;
            mpls port num;
            iso port num;
        }

        ttlttl-value;

    }
}
family inet {
    address ip_address;
}
family inet6 {
    address ip_address;
}
family mpls;
family iso;
}
}

```

Benefits of Flexible Tunnel Interfaces

- Entropy and load balancing occur in transit. Unlike over tunnel encapsulations, such as IP in IP or generic routing encapsulation (GRE), VXLAN encapsulation supports passing of the hash computation result in the source port of the UDP datagram. This enables you to load-balance traffic efficiently in transit.

- FTIs have an extensible design that enables them to support multiple encapsulations.
- The `vni` attribute of the VXLAN encapsulation in FTIs helps in customer isolation.
- FTIs with UDP encapsulation use the source and destination port in the UDP header. Because the UDP source port is derived from the hash value of the inner payload, you can benefit from better traffic distribution over ECMP.

Limitations of Flexible Tunnel Interfaces

- Policing follows the distributed forwarding model of the FTIs; therefore provisioned bandwidth limits are enforced at an individual Packet Forwarding Engine level. As a result, more traffic might be admitted.
- Currently, FTI-tunneled traffic is strictly routed in the `inet.0` instance. Therefore, FTIs support only IPv4 traffic.
- The MX80 does not support FTIs.
- Class-of-service (CoS) configuration, including the configuration of rewrite rules and classifiers is not supported on FTIs.
- Time-to-live (TTL) on the tunnel header is set to the default value 64.
- Differentiated Services code point (DSCP) value is set to the default value 0, but internal forwarding class and loss priority fields are retained and can be used to rewrite DSCP in the egress interface rewrite rules.
- IP fragmentation is not supported on FTIs.

FTI with UDP encapsulation do not support the following features and functionality:

- BFD over LDP and RSVP is not supported.
- Aggregate Ethernet member statistics on QFX1000 device is not supported.
- 10,000 routes per FTI logical interface is not supported.
- Routing instance is not supported.
- Logical systems is not supported.
- Path MTU discovery is not supported.
- Policing and firewall is not supported.
- BGP signaling for UDP tunnels is not supported.

- Class-of-service on tunnel endpoints is not supported.
- TTL propagation is not supported.
- Multicast traffic is not supported.
- Plain IPV6 UDP tunnel is not supported.
- Anti-spoofing check for tunneled traffic is not supported.
- MPLS FRR is not supported.
- FT-over-FT resolution is not supported.
- FT destination IP should be reachable through IGP and not BGP (no indirect next hop). The reachability should be through an IPV4 route and not through an LSP.
- FT physical interface level statistics is not supported.
- All the interfaces under FTI except for fti0 are not supported.
- Un-numbered address is not supported.

SEE ALSO

show interfaces fti

vxlان-gpe (FTI)

vni(Interfaces)

destination-udp-port (FTI)

Configuring Flexible Tunnel Interfaces

IN THIS SECTION

- [Configuring FTI on PE1 | 64](#)
- [Verification | 66](#)

You can configure flexible tunnel Interfaces (FTIs) that support the Virtual Extensible LAN (VXLAN) encapsulation with Layer 2 pseudo-headers on MX Series routers, or UDP encapsulation on PTX Series routers and QFX Series switches. A flexible tunnel interface (FTI) is a point-to-point Layer 3 interface that can be used to create IPv4 and IPv6 overlays over an IPv4 transport network. A BGP protocol session can be configured to run over FTIs in order to distribute routing information.

The following sections describe how to configure FTIs on your device and to enable multiple encapsulations using the `udp` or `vxlan-gpe` parameter under the mandatory `tunnel-endpoint vxlan` encapsulation identified with the `vni` and `destination-udp-port` values:

Configuring FTI on PE1

You can configure an FTI by including the `tunnel-endpoint vxlan` statement at the `[edit interfaces]` hierarchy level.

To configure an FTI and define its attributes for an IPv4 network:

1. In configuration mode, go to the `[edit interfaces]` hierarchy level.

```
[edit]
user@host# edit interfaces
```

2. On MX Series routers, configure a logical unit for the interface and the encapsulation `vxlan-gpe`. The unit is a logical interface configured on the physical device. Specify the value of the unit from 0 through 8191. VXLAN is defined as an encapsulation format that encapsulates Ethernet frames in an outer UDP/IP transport.



NOTE: The capabilities of VXLAN-GPE are a super-set of what VXLAN without protocol extension offers. Therefore generic `vxlan-gpe` hierarchy is introduced to configure VXLAN tunnel encapsulation attributes; however, only regular VXLAN encapsulation without protocol extensions and pseudo Layer 2 MAC is used. The pseudo Layer 2 address is populated with “pseudo” source (source MAC: 00-00-5E-00-52-00) and destination (destination MAC: 00-00-5E-00-52-01) MAC addresses without VLAN tagging.

On MX Series routers:

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe
```


Starting in Junos OS Release 19.3R1, you can configure flexible tunnel interfaces (FTIs) with UDP encapsulation on the PTX Series routers and the QFX Series switches, which provide support for static UDP tunnels only.

FTIs with UDP encapsulation provides the benefit of better traffic distribution over ECMP, that is achieved by the UDP source port derived from the hash value of the inner payload. In addition to this, the other benefits of this feature include, shortened interface hop counts, smooth IGP domain separation, and reduced operational complexity.

On PTX Series routers and QFX Series switches:

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation udp
```

3. Configure the source address for the interface. The source address is the IPv4 address or address range of the encapsulator (the local ingress PE router).

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe source
address
```



NOTE: The source address can be a global WAN address and loopback address (lo0) is not mandatory.

4. Configure the destination address for the interface. The destination address is the IPv4 address of the tunnel endpoint destination.

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe
destination-address address
```

5. Configure tunnel-endpoint with the encapsulation vxlan. This step is mandatory to enable Layer 2 pseudo-header with VXLAN encapsulation.

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe tunnel-
endpoint vxlan
```

6. Specify the UDP port value of the destination to be used in the UDP header for the generated frames. The numeric value for `destination-udp-port` identifies the endpoint. Specify the value of `destination-udp-port` from 1 through 65,535.

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe
destination-udp-port port-number
```

7. Specify the virtual network identifier (VNI) value to be used to identify the encapsulation, `vxlan-gpe`. Specify the value of the `vni` from 0 through 16,777,214.

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number tunnel encapsulation vxlan-gpe vni vni-
number
```

8. Configure an IPv4 address of an interface (signified by family `inet`). For IPv6 address configuration, use the `inet6` family.

```
[edit interfaces]
user@host# set interface-name unit logical-unit-number family inet address address
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Purpose | 66
- Action | 67
- Meaning | 67

Purpose

Verify that the FTI is configured and verify its status.

Action

In configuration mode, you can verify if FTI on MX Series router has been configured by executing the `show interfaces fti number` command.

```
user@host# show interfaces fti0
Physical interface: fti0, Enabled, Physical link is Up
  Interface index: 136, SNMP ifIndex: 504
  Type: FTI, Link-level type: Flexible-tunnel-Interface, MTU: Unlimited, Speed: Unlimited
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

Logical interface fti0.0 (Index 340) (SNMP ifIndex 581)
  Flags: Up Point-To-Point SNMP-Traps Encapsulation: VXLAN-GPEv4
  Destination UDP port: 4789, VNI: 1000, Source address: 5.5.5.5, Destination address: 6.6.6.6
  Input packets : 0
  Output packets: 0
  Protocol inet, MTU: Unlimited
  Max nh cache: 0, New hold nh limit: 0, Curr nh cnt: 0, Curr new hold cnt: 0, NH drop cnt: 0
  Flags: Sendbroadcast-pkt-to-re
  Addresses, Flags: Is-Preferred Is-Primary
    Destination: 3.3.3/24, Local: 3.3.3.5, Broadcast: 3.3.3.255
```

Similarly you can execute the `show interfaces fti0 detail`, `show interfaces fti0 extensive`, `show interfaces fti0 terse`, and `show interfaces fti0 statistics` commands to get more details FTIs. See *show interfaces fti*.

Meaning

The `show interfaces fti0` command displays the status of the FTIs that have been configured with the new encapsulation `vxlan-gpe`. The output verifies that the FTI is configured and the physical link is up.

RELATED DOCUMENTATION

show interfaces fti

vxlan-gpe (FTI)

vni(Interfaces)

destination-udp-port (FTI)

Configuring a Flexible Tunnel Interface on an SRX Firewall

IN THIS SECTION

- [Verify Flexible Tunnel Creation | 69](#)

When you configure a Flexible Tunnel Interface (FTI) on an SRX firewall, you must also configure the zone and security policy for the interface.

A security zone is a collection of one or more network segments that requires the regulation of inbound and outbound traffic through policies. You assign the FTI to one of the security zone and the FTI functions as a security doorway from one security zone to another. Security policies control the traffic flow through the FTI. You can configure the security policies on the SRX firewall to permit or deny traffic pass through the FTI. The following sample configuration shows how to configure an FTI on the SRX firewall.

1. Configure the FTI with vxlan-gpe encapsulation.

```
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe source address 198.51.100.1
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination address 198.51.100.2
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe tunnel-endpoint vxlan
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination-udp-port 4789
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe vni 22701
set interfaces fti0 unit 0 family inet address 198.51.100.1/24
set interfaces lo0 unit 0 family inet address 192.168.100.1
```

2. Configure and assign the FTI to a security zone. For more information on security zones, see <https://www.juniper.net/documentation/us/en/software/junos/security-policies/topics/topic-map/security-zone-configuration.html>.

```
set security zones security-zone FTI-ZONE host-inbound-traffic system-services all
set security zones security-zone FTI-ZONE host-inbound-traffic protocols all
set security zones security-zone FTI-ZONE interfaces fti0.0
```

3. Create a policy for traffic being sent to the FTI and the actions that need to take place as the traffic passes through the interface. In this example, we permit all traffic to pass through. For more information on configuring security policies, see <https://www.juniper.net/documentation/us/en/software/junos/security-policies/topics/topic-map/security-policy-configuration.html>.

```
set security policies from-zone FTI-ZONE to-zone trust policy fti-out match source-address any
set security policies from-zone FTI-ZONE to-zone trust policy fti-out match destination-address any
set security policies from-zone FTI-ZONE to-zone trust policy fti-out match application any
set security policies from-zone FTI-ZONE to-zone trust policy fti-out then permit
set security policies from-zone FTI-ZONE to-zone trust policy fti-in match source-address any
set security policies from-zone FTI-ZONE to-zone trust policy fti-in match destination-address any
set security policies from-zone FTI-ZONE to-zone trust policy fti-in match application any
set security policies from-zone FTI-ZONE to-zone trust policy fti-in then permit
set routing-options static route 198.51.100.2/32 next-hop 10.100.12.2
```

Verify Flexible Tunnel Creation

Use the **show interfaces fti0.0** command to display information about the flexible tunnel interface:

```
user@device1>show interfaces fti0.0
Logical interface fti0.0 (Index 72) (SNMP ifIndex 520)
  Flags: Up Point-To-Point SNMP-Traps Encapsulation: VXLAN-GPEv4
  Destination UDP port: 4789, Source UDP port range: [49160 - 65535],
  VNI: 22701, Source address: 10.0.0.2, Destination address: 10.0.0.1
  Input packets : 0
  Output packets: 5
  Security: Zone: FTI-ZONE
  Allowed host-inbound traffic : bootp bfd bgp dns dvmrp igmp ldp msdp nhrp ospf ospf3 pgm pim
  rip ripng router-discovery
```

```

    rsvp sap vrrp dhcp finger ftp tftp ident-reset http https ike netconf ping reverse-telnet
reverse-ssh rlogin rpm rsh snmp
    snmp-trap ssh telnet traceroute xnm-clear-text xnm-ssl lsping lsselfping ntp sip dhcpv6 r2cp
webapi-clear-text webapi-ssl
    tcp-encap sdwan-appqoe high-availability
Protocol inet, MTU: 1450
Max nh cache: 0, New hold nh limit: 0, Curr nh cnt: 0, Curr new hold cnt: 0, NH drop cnt: 0
    Flags: Sendbcast-pkt-to-re
Addresses, Flags: Is-Preferred Is-Primary
    Destination: 10.18.1/24, Local: 10.18.1.2, Broadcast: 10.18.1.255

```

Example: Configuring Flexible Tunnel Interfaces on MX Series Routers

IN THIS SECTION

- [Requirements | 70](#)
- [Overview | 71](#)
- [Configuration | 71](#)
- [Verification | 77](#)

Requirements

This example uses the following hardware and software components:

- An MX10003 and an MX Series 5G Universal Routing Platform.
- Junos OS Release 18.3 or later.

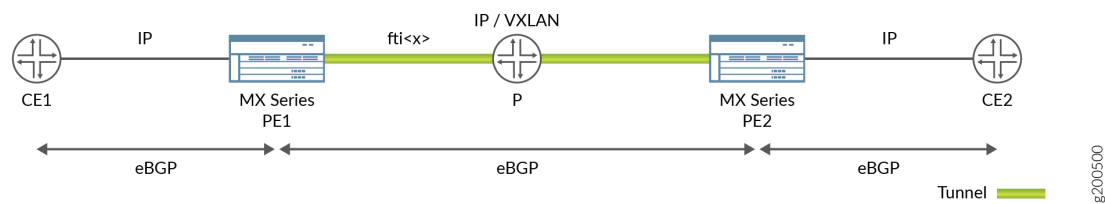
Overview

In this example, flexible tunnel interfaces are used to create a Layer 3 VPN overlay network between two routers. In the actual deployment, one of the endpoints can be the server in a data center or a data center gateway.

Consider a sample topology in which a gateway device, PE1, functions as a link between the enterprise customers to represent the customer side for an FTI tunnel. eBGP is used to distribute routes between customer edge (CE1) and provider edge (PE1) devices. IPv4 is used for transmission of test frames over the Layer 3 network. This test is used to transfer the traffic between CE1 and CE2. Logical interfaces on both the routers are configured with IPv4 addresses to create an FTI to transfer the traffic of network devices for the IPv4 service.

Figure 2 on page 71 shows the sample topology of how an FTI performs for a Layer 3 IPv4 service.

Figure 2: Flexible Tunnel Interfaces Topology



Configuration

IN THIS SECTION

- [CLI Quick Configuration | 72](#)
- [Configuring on PE1 | 72](#)
- [Configuring on PE2 | 74](#)
- [Results | 75](#)

In this example, you configure FTI for a Layer 3 IPv4 service that is between interface fti0 on PE1 and interface fti0 on PE2 to form a tunnel interface of the interconnecting routers.

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them in a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level:

To Configure Parameters on PE1

```
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe source address 198.51.100.1
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination address 198.51.100.2
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe tunnel-endpoint vxlan
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination-udp-port 4789
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe vni 22701
set interfaces fti0 unit 0 family inet address 198.51.100.1/24
```

To Configure Parameters on PE2

```
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe source address 198.51.100.2
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination address 198.51.100.1
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe tunnel-endpoint vxlan
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination-udp-port 4789
set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe vni 22701
set interfaces fti0 unit 0 family inet address 198.51.100.2/24
```

Configuring on PE1

Step-by-Step Procedure

The following steps require you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the [Junos OS CLI User Guide](#).

To configure the parameters on PE1:

1. In configuration mode, go to the [edit interfaces] hierarchy level:

```
[edit]
user@host# edit interfaces
```

2. Configure the FTI and a logical unit and specify the protocol family.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe
```

3. Specify the source address for the logical interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe source address 198.51.100.1
```

4. Specify the destination address for the logical interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination address 198.51.100.2
```

5. Set tunnel-endpoint with the encapsulation vxlan.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe tunnel-endpoint vxlan
```

6. Specify the UDP port value of the destination to be used in the UDP header for the generated frames.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination-udp-port 4789
```

7. Specify the vni value to be used to identify the encapsulation `vxlان-gpe` on the interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlان-gpe vni 22701
```

8. Specify the address type family for the interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 family inet address 198.51.100.1/24
```

Configuring on PE2

Step-by-Step Procedure

The following steps require you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the [Junos OS CLI User Guide](#).

To configure the parameters on PE2:

1. In configuration mode, go to the `[edit interfaces]` hierarchy level:

```
[edit]
user@host# edit interfaces
```

2. Configure the FTI and a logical unit and specify the protocol family.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlان-gpe
```

3. Specify the source address for the logical interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlان-gpe source address
198.51.100.2
```

4. Specify the destination address for the logical interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination address
198.51.100.1
```

5. Set tunnel-endpoint with the encapsulation vxlan.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe tunnel-endpoint vxlan
```

6. Specify the UDP port value of the destination to be used in the UDP header for the generated frames.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe destination-udp-port
4789
```

7. Specify the vni value to be used to identify the encapsulation vxlan-gpe on the interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 tunnel encapsulation vxlan-gpe vni 22701
```

8. Specify the address type family for the interface.

```
[set interfaces]
user@host# set interfaces fti0 unit 0 family inet address 198.51.100.2/24
```

After the configuration is successfully completed, you can view the parameters by entering the `show fti0` command.

Results

In configuration mode, confirm your configuration on PE1 and PE2 by entering the `show` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

Parameters on PE1:

```
[edit interfaces]
fti0{
  unit 0 {
  tunnel {
    encapsulation vxlan-gpe {
      source {
        address 198.51.100.1;
      }
      destination {
        address 198.51.100.2;
      }
      tunnel-endpoint vxlan;
      destination-udp-port 4789;
      vni 22701;
    }
  }
  family inet {
    address 198.51.100.1/24;
  }
}
```

Parameters on PE2:

```
[edit interfaces]
fti0{
  unit 0 {
  tunnel {
    encapsulation vxlan-gpe {
      source {
        address 198.51.100.2;
      }
      destination {
        address 198.51.100.1;
      }
      tunnel-endpoint vxlan;
      destination-udp-port 4789;
      vni 22701;
    }
  }
}
```

```
family inet {
    address 198.51.100.2/24;
}
}
```

After you have configured the interface, enter the `commit` command in configuration mode.

Verification

IN THIS SECTION

- [Verifying the Results | 77](#)

Verifying the Results

Purpose

Verify that the necessary and desired tunnel displays the values configured for the FTI test that is run on the flexible tunnel between PE1 and PE2.

Action

In operational mode, enter the `show interfaces fti0` command to display status of the FTIs that have been configured with the new encapsulation `vxlan-gpe`. The output verifies that the FTI is configured and the physical link is up.

SEE ALSO

`show interfaces fti`
`vni(Interfaces)`

Configuring IP-IP Decapsulation by Tunnel Termination on FTI

In filter based decapsulation, the decapsulated packets are re-circulated for inner header lookup and forwarded accordingly. However, tunnel termination is completed in a single pass of packet processing, thus providing performance improvement over filter based process. Starting in Junos OS Evolved

Release 20.1R2, you can configure IP-IP decapsulation on a flexible tunnel interface on PTX series routers by configuring tunnel termination. You can configure IP-IP decapsulation on a flexible tunnel interface by configuring tunnel termination at the [edit interfaces fti0 unit *number* tunnel encapsulation IPIP] hierarchy level.



NOTE: For the Junos OS Evolved Release 20.1R2, FTI does not support encapsulation.

To configure IP-IP decapsulation by tunnel termination:

1. On PTX Series routers, configure the FTI, logical unit for the interface, and the encapsulation IPIP. The unit is a logical interface configured on the physical device. Specify the value of the unit from 0 through 4096.

```
[edit interfaces]
user@host# set fti0 unit logical-unit-number tunnel encapsulation IPIP
```

2. For IP-IP decapsulation, configure the tunnel termination and specify the address family. For IPv6 address configuration, use the inet6 family.



NOTE: For the Junos OS Evolved Release 20.1R2, this step is mandatory.

```
[edit interfaces fti0 unit logical-unit-number tunnel encapsulation IPIP]
user@host# set tunnel-termination
user@host# set famlly inet
```

3. Configure the source address and destination address for the interface.

```
[edit interfaces fti0 unit logical-unit-number tunnel encapsulation IPIP]
user@host# set source address 192.168.1.1
user@host# set destination address 192.168.2.1
```

4. Configure the routing instance for the FTI to facilitate routing table lookups. Create a virtual-router instance and associate the interface.

```
[edit routing-instances routing-instance-name]
user@host# set instance-type virtual-router
```

5. Verify the tunnel termination.

```
[edit]
user@host# show interfaces fti0 detail
```

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
20.1R2 Evo	Starting in Junos OS Evolved Release 20.1R2, you can configure IP-IP decapsulation on a flexible tunnel interface on PTX series routers by configuring tunnel termination.
19.3R1	FTI is supported in releases starting Junos OS Release 19.3R1.
19.3R1	Starting in Junos OS Release 19.3R1, you can configure flexible tunnel interfaces (FTIs) with UDP encapsulation on the PTX Series routers and the QFX Series switches, which provide support for static UDP tunnels only.

Configuring GRE Tunnel Interfaces

IN THIS SECTION

- [Understanding Generic Routing Encapsulation | 80](#)
- [Configuring Generic Routing Encapsulation Tunneling | 83](#)
- [GRE Keepalive Time Overview | 85](#)
- [Configuring GRE Keepalive Time | 87](#)
- [Enabling Fragmentation on GRE Tunnels | 91](#)

Understanding Generic Routing Encapsulation

IN THIS SECTION

- [Overview of GRE | 80](#)
- [GRE Tunneling | 81](#)
- [Configuration Limitations | 82](#)
- [Platform-Specific GRE Encapsulation or Decapsulation Behavior | 83](#)

Generic routing encapsulation (GRE) provides a private, secure path for transporting packets through an otherwise public network by encapsulating (or tunneling) the packets.

This topic describes:

Overview of GRE

GRE encapsulates data packets and redirects them to a device that de-encapsulates them and routes them to their final destination. This allows the source and destination routers to operate as if they have a virtual point-to-point connection with each other (because the outer header applied by GRE is transparent to the encapsulated payload packet). For example, GRE tunnels allow routing protocols such as RIP and OSPF to forward data packets from one router to another router across the Internet. In addition, GRE tunnels can encapsulate multicast data streams for transmission over the Internet.

GRE is described in RFC 2784 (obsoletes earlier RFCs 1701 and 1702). The routers support RFC 2784, but not completely. (For a list of limitations, see ["Configuration Limitations" on page 82.](#))

As a *tunnel source router*, the router encapsulates a payload packet for transport through the tunnel to a destination network. The payload packet is first encapsulated in a GRE packet, and then the GRE packet is encapsulated in a delivery protocol. The router performing the role of a *tunnel remote router* extracts the tunneled packet and forwards the packet to its destination.



NOTE: Service chaining for GRE, NAT, and IPSec services on ACX1100-AC and ACX500 routers is not supported.



NOTE: Layer 2 over GRE is not supported in ACX2200 router.

ACX routers support OSPF routing protocol when a GRE tunnel is configured on a WAN interface.

For details on platform and Junos version support, see [Feature Explorer](#).

GRE Tunneling

Data is routed by the system to the GRE endpoint over routes established in the route table. (These routes can be statically configured or dynamically learned by routing protocols such as RIP or OSPF.) When a data packet is received by the GRE endpoint, it is de-encapsulated and routed again to its destination address.

GRE tunnels are *stateless*—that is, the endpoint of the tunnel contains no information about the state or availability of the remote tunnel endpoint. Therefore, the router operating as a tunnel source router cannot change the state of the GRE tunnel interface to down if the remote endpoint is unreachable.

For details about GRE tunneling, see:

Encapsulation and De-Encapsulation on the Router

Encapsulation—A router operating as a tunnel source router encapsulates and forwards GRE packets as follows:

1. When a router receives a data packet (payload) to be tunneled, it sends the packet to the tunnel interface.
2. The tunnel interface encapsulates the data in a GRE packet and adds an outer IP header.
3. The IP packet is forwarded on the basis of the destination address in the outer IP header.

De-encapsulation—A router operating as a tunnel remote router handles GRE packets as follows:

1. When the destination router receives the IP packet from the tunnel interface, the outer IP header and GRE header are removed.
2. The packet is routed based on the inner IP header.

Number of Source and Destination Tunnels Allowed on a Router

ACX routers support as many as 64 GRE tunnels between routers transmitting IPv4 or IPv6 payload packets over GRE.

Configuration Limitations


Some GRE tunneling features are not currently available on ACX Series routers. Be aware of the following limitations when you are configuring GRE on an ACX router:

- **Unsupported features**—GRE on the ACX routers *does not support* the following features:
 - Virtual routing over GRE
 - Bidirectional Forwarding Detection (BFD) protocol over GRE distributed mode
 - MPLS over GRE tunnels
 - GRE keepalives
 - GRE keys, payload packet fragmentation, and sequence numbers for fragmented packets
 - BGP dynamic tunnels
 - RFC 1701 and RFC 1702
 - RFC 2890—Key and sequence number extensions to GRE
 - IPv6 as delivery header
 - GRE path MTU discovery
 - Load balancing when NNI is ECMP
 - Interface statistics on GRE interfaces
 - Class of service and firewall on GRE tunnel
- **Routing Protocol**—ACX routers do not support routing protocols on GRE interfaces. You need to disable routing on GRE interfaces under the [edit protocols] hierarchy. For example,

```
[edit]
user@host# show protocols
ospf {
  area 0.0.0.0 {
    interface all;
    interface gr-0/0/10.0 {
      disable;
    }
  }
}
```

Platform-Specific GRE Encapsulation or Decapsulation Behavior

Platform	Difference
ACX7000	Supports GRE encapsulation and decapsulation through Flexible Tunnel Interfaces (FTIs)

-  **NOTE:** This limitation is applicable for all routing protocols (such as OSPF, ISIS).

SEE ALSO

| [Configuring Unicast Tunnels](#) | 97

Configuring Generic Routing Encapsulation Tunneling

IN THIS SECTION

-  [Configuring a GRE Tunnel Port](#) | 84
-  [Configuring Tunnels to Use Generic Routing Encapsulation](#) | 84

Tunneling provides a private, secure path for transporting packets through an otherwise public network by encapsulating packets inside a transport protocol known as an *IP encapsulation protocol*. Generic routing encapsulation (GRE) is an IP encapsulation protocol that is used to transport packets over a network. Information is sent from one network to the other through a GRE tunnel.

GRE tunneling is accomplished through routable tunnel endpoints that operate on top of existing physical and other logical endpoints. GRE tunnels connect one endpoint to another and provide a clear data path between them.

This topic describes:

Configuring a GRE Tunnel Port

To configure GRE tunnels on a router, you convert a network port or uplink port on the router to a GRE tunnel port for tunnel services. Each physical tunnel port, named *gr-fpc/pic/port*, can have one or more logical interfaces, each of which is a GRE tunnel.

After conversion to a GRE tunnel port, the physical port cannot be used for network traffic.

To configure a GRE tunnel port on an router, you need to create logical tunnel interfaces and the bandwidth in gigabits per second to reserve for tunnel services. Include the `tunnel-services bandwidth (1g / 10g)` statement at the `[edit chassis fpc slot-number pic number]` hierarchy level.

To configure a GRE tunnel port , use any unused physical port on the router to create a logical tunnel interface as shown below:

```
user@host# edit chassis
fpc 0 {
  pic 0 {
    tunnel-services {
      port port-number;
    }
  }
}
```

This also creates a `gr-` interface.

Configuring Tunnels to Use Generic Routing Encapsulation

Normally, a GRE tunnel port comes up as soon as it is configured and stays up as long as a valid tunnel source address exists or an interface is operational. Each logical interface you configure on the port can be configured as the source or as the endpoint of a GRE tunnel.

To configure a tunnel port to use GRE:

1. Configure a physical GRE port with a logical interface name and address:

- For IPv4 over GRE, specify the protocol family `inet`:

```
[edit interfaces]
user@host# set gr-fpc/pic/port unit number family inet address
```

- For IPv6 over GRE, specify the protocol family `inet6`:

```
[edit interfaces]
user@host# set gr-fpc/pic/port unit number family inet6 address
```

2. Specify the tunnel source address for the logical interface:

```
[edit interfaces]
user@host# set gr-fpc/pic/port unit number tunnel source source-address
```

3. Specify the destination address:

```
[edit interfaces]
user@host# set gr-fpc/pic/port unit number tunnel destination destination-address
```

RELATED DOCUMENTATION

| [Configuring Unicast Tunnels](#) | 97

GRE Keepalive Time Overview

Generic routing encapsulation (GRE) tunnel interfaces do not have a built-in mechanism for detecting when a tunnel is down. You can enable keepalive messages to serve as the detection mechanism.

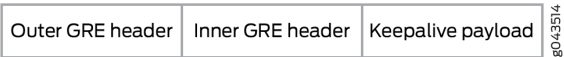
When you enable a GRE tunnel interface for keepalive messages, the interface sends out keepalive request packets to the remote endpoint at regular intervals. If the data path forwarding for the GRE tunnel works correctly at all points, keepalive response packets are returned to the originator. These keepalive messages are processed by the Routing Engine.

You can configure keepalive messages on the physical or logical GRE tunnel interface. If configured on the physical interface, keepalive messages are sent on all logical interfaces that are part of the physical interface. If configured on an individual logical interface, keepalives are sent only on that logical interface.

You configure how often keepalive messages are sent and the length of time that the interface waits for a keepalive response before marking the tunnel as operationally down.

The keepalive request packet is shown in [Figure 3 on page 86](#).

Figure 3: Keepalive Request Packet



The keepalive payload includes information to ensure the keepalive response is correctly delivered to the application responsible for the GRE keepalive process.

The outer GRE header includes:

- Source IP Address—IP address of the endpoint that initiates the keepalive request
- Destination IP Address—IP address of the endpoint that receives the keepalive request
- GRE Protocol ID—IP

The inner GRE header includes:

- Source IP Address—IP address of the endpoint that receives the keepalive request
- Destination IP Address—IP address of the endpoint that initiates the keepalive request
- GRE Protocol ID—A value that the packet forwarding engine recognizes as a GRE keepalive packet



NOTE: Starting in Junos OS Release 17.3R1, you can configure IPv6 generic routing encapsulation (GRE) tunnel interfaces on MX Series routers. This lets you run a GRE tunnel over an IPv6 network. Packet payload families that can be encapsulated within the IPv6 GRE tunnels include IPv4, IPv6, MPLS, and ISO. Fragmentation and reassembly of the IPv6 delivery packets is not supported.

To configure an IPv6 GRE tunnel interface, specify IPv6 addresses for source and destination at the [interfaces gr-0/0/0 unit 0 tunnel] hierarchy level.

Keepalive is not supported for GRE IPv6.

SEE ALSO

keepalive-time
hold-time

Configuring GRE Keepalive Time

IN THIS SECTION

- [Configuring Keepalive Time and Hold time for a GRE Tunnel Interface | 87](#)
- [Display GRE Keepalive Time Configuration | 88](#)
- [Display Keepalive Time Information on a GRE Tunnel Interface | 89](#)

Configuring Keepalive Time and Hold time for a GRE Tunnel Interface

You can configure the keepalives on a generic routing encapsulation (GRE) tunnel interface by including both the `keepalive-time` statement and the `hold-time` statement at the `[edit protocols oam gre-tunnel interface interface-name]` hierarchy level.



NOTE: For proper operation of keepalives on a GRE interface, you must also include the `family inet` statement at the `[edit interfaces interface-name unit unit]` hierarchy level. If you do not include this statement, the interface is marked as down.

To configure a GRE tunnel interface:

1. Configure the GRE tunnel interface at `[edit interfaces interface-name unit unit-number]` hierarchy level, where the interface name is `gr-x/y/z`, and the family is set as `inet`.

```
user@host# set interfaces interface-name unit unit-number family family-name
```

2. Configure the rest of the GRE tunnel interface options as explained in *Configuring a GRE Tunnel Interface Between a PE and CE Router* or *Configuring a GRE Tunnel Interface Between PE Routers* based on requirement.

To configure keepalive time for a GRE tunnel interface:

1. Configure the Operation, Administration, and Maintenance (OAM) protocol at the `[edit protocols]` hierarchy level for the GRE tunnel interface.

```
[edit]
user@host# edit protocols oam
```

2. Configure the GRE tunnel interface option for OAM protocol.

```
[edit protocols oam]
user@host# edit gre-tunnel interface interface-name
```

3. Configure the keepalive time from 1 through 50 seconds for the GRE tunnel interface.

```
[edit protocols oam gre-tunnel interface interface-name]
user@host# set keepalive-time seconds
```

4. Configure the hold time from 5 through 250 seconds. Note that the hold time must be at least twice the keepalive time.

```
[edit protocols oam gre-tunnel interface interface-name]
user@host# set hold-time seconds
```

Display GRE Keepalive Time Configuration

IN THIS SECTION

- [Purpose | 88](#)
- [Action | 88](#)

Purpose

Display the configured keepalive time value as 10 and hold time value as 30 on a GRE tunnel interface (for example, gr-1/1/10.1).

Action

To display the configured values on the GRE tunnel interface, run the `show oam gre-tunnel` command at the `[edit protocols]` hierarchy level:

```
[edit protocols]
user@host# show oam gre-tunnel
```



```
interface gr-1/1/10.1 {
    keepalive-time 10;
    hold-time 30;
}
```

Display Keepalive Time Information on a GRE Tunnel Interface

IN THIS SECTION

Purpose | 89

Action | 89

Meaning | 90

Purpose

Display the current status information of a GRE tunnel interface when keepalive time and hold time parameters are configured on it and when the hold time expires.

Action

To verify the current status information on a GRE tunnel interface (for example, gr-3/3/0.3), run the `show interfaces gr-3/3/0.3 terse` and `show interfaces gr-3/3/0.3 extensive` operational commands.

`show interfaces gr-3/3/0.3 terse`

```
user@host> show interfaces gr-3/3/0.3 terse
```

Interface	Admin	Link	Proto	Local	Remote
gr-3/3/0.3	up	up	inet	192.0.2.1/24	
			mpls		

`show interfaces gr-3/3/0.3 extensive`

```
user@host> show interfaces gr-3/3/0.3 extensive
Logical interface gr-3/3/0.3 (Index 73) (SNMP ifIndex 594) (Generation 900)
  Flags: Point-To-Point SNMP-Traps 0x4000 IP-Header
10.1.19.11:10.1.19.12:47:df:64:0000000000000000 Encapsulation: GRE-NULL
  Gre keepalives configured: On, Gre keepalives adjacency state: down
```

```

*****
Traffic statistics:
  Input bytes :          15629992
  Output bytes :         15912273
  Input packets:          243813
  Output packets:         179476
Local statistics:
  Input bytes :          15322586
  Output bytes :         15621359
  Input packets:          238890
  Output packets:         174767
Transit statistics:
  Input bytes :          307406           0 bps
  Output bytes :         290914           0 bps
  Input packets:           4923           0 pps
  Output packets:          4709           0 pps
Protocol inet, MTU: 1476, Generation: 1564, Route table: 0
  Flags: Sendbroadcast-pkt-to-re
  Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
*****

  Destination: 192.0.2/24, Local: 192.0.2.1, Broadcast: 192.0.2.255, Generation: 1366
Protocol mpls, MTU: 1464, Maximum labels: 3, Generation: 1565, Route table: 0

```



NOTE: When the hold time expires:

- The GRE tunnel will stay up even though the interface cannot send or receive traffic.
- The Link status will be Up and the Gre keepalives adjacency state will be Down.

Meaning

The current status information of a GRE tunnel interface with keepalive time and hold time parameters is displayed as expected when the hold time expires.

RELATED DOCUMENTATION

keepalive-time

hold-time

Enabling Fragmentation on GRE Tunnels

To enable fragmentation of IPv4 packets in generic routing encapsulation (GRE) tunnels, include the `clear-dont-fragment-bit` statement and a maximum transmission unit (MTU) setting for the tunnel as part of an existing GRE configuration at the `[edit interfaces]` hierarchy level:

```
[edit interfaces]
gr-fpc/pic/port {
  unit logical-unit-number {
    clear-dont-fragment-bit;
    ...
    family inet {
      mtu 1000;
      ...
    }
  }
}
```

This statement clears the Don't Fragment (DF) bit in the packet header, regardless of the packet size. If the packet size exceeds the tunnel MTU value, the packet is fragmented before encapsulation. The maximum MTU size configurable on the AS or Multiservices PIC is 9192 bytes.



NOTE: The `clear-dont-fragment-bit` statement is supported only on MX Series routers and all M Series routers except the M320 router.



NOTE: On SRX platforms the clearing of the DF bit on a GRE tunnel is supported only when the device is in packet or selective packet mode; This feature is not supported in flow mode. As a result, when in flow mode, a packet that exceeds the MTU of the GRE interface with the DF bit set is dropped, despite having the `clear-dont-fragment-bit` configured on the GRE interface.

Fragmentation is enabled only on IPv4 packets being encapsulated in IPv4-based GRE tunnels.



NOTE: This configuration is supported only on GRE tunnels on AS or Multiservices interfaces. If you commit `gre-fragmentation` as the encapsulation type on a standard Tunnel PIC interface, the following console log message appears when the PIC comes online:

gr-fpc/pic/port: does not support this encapsulation

The Packet Forwarding Engine updates the IP identification field in the outer IP header of GRE-encapsulated packets, so that reassembly of the packets is possible after fragmentation. The previous CLI constraint check that required you to configure either the `clear-dont-fragment-bit` statement or a tunnel key with the `allow-fragmentation` statement is no longer enforced.

When you configure the `clear-dont-fragment-bit` statement on an interface with the MPLS protocol family enabled, you must specify an MTU value. This MTU value must not be greater than maximum supported value (9192).

SEE ALSO

[Configuring Unicast Tunnels | 97](#)

Soft GRE Capability

IN THIS SECTION

- [Soft GRE Capability Overview | 92](#)
- [Configure Tunnel Attributes for GRE or UDP Dynamic Tunnels | 94](#)

Soft GRE Capability Overview

IN THIS SECTION

- [Benefits of Soft GRE Capability | 93](#)
- [Understanding Dynamic Tunnel Configuration | 93](#)

The Soft GRE Capability enables tunneling of Q-in-Q Ethernet frames across network infrastructure, effectively facilitating routing, switching, and tunneling roles. This feature encapsulates and decapsulates Ethernet frames over GRE tunnels, preserving original Ethernet headers and MAC addresses for accurate traffic forwarding using VLAN tags. Quality of Service (QoS) is meticulously supported with traffic marking, classification, and prioritization based on PCP and DSCP values, alongside comprehensive policing mechanisms. The implementation also includes dynamic tunnel creation without signaling protocols and enhanced Connectivity Fault Management (CFM) support with detailed Ethernet OAM functionalities, ensuring robust performance monitoring and fault management.

Benefits of Soft GRE Capability

- Enables efficient tunneling of Q-in-Q Ethernet frames over GRE tunnels, preserving original Ethernet headers and MAC addresses for accurate traffic forwarding using VLAN tags.
- Supports comprehensive QoS mechanisms, including traffic marking, classification, and prioritization based on PCP and DSCP values, ensuring optimal traffic management.
- Facilitates the dynamic creation of GRE tunnels without the need for signaling protocols, simplifying the setup process and reducing the complexity of network configuration.
- Provides enhanced CFM support with detailed Ethernet OAM functionalities, allowing for robust performance monitoring and effective fault management in the network.

Understanding Dynamic Tunnel Configuration

The implementation of the Soft GRE capability enables the encapsulation and decapsulation of Q-in-Q Ethernet frames over GRE tunnels. This process begins by mapping S-VLAN, C-VLAN, and LAG interfaces to MPLS labels and GRE tunnel endpoints. By preserving the original Ethernet headers and MAC addresses, the feature ensures that traffic is forwarded accurately using VLAN tags. Through these mappings, you can achieve efficient tunneling of Ethernet traffic within network infrastructure.

Dynamic tunnel creation is a key aspect of the Soft GRE capability, as it allows GRE tunnels to be established without the need for signaling protocols. Instead, tunnels are dynamically created and bound to Q-in-Q interfaces using Layer 2 circuits. This simplifies the setup process and reduces network configuration complexity.

Configuring GRE or UDP dynamic tunnels involves setting specific attributes that enhance security and routing efficiency within complex network environments. By disabling anti-spoofing measures and preserving the input logical interface (IFL), you ensure that traffic is accurately routed based on the original input interface instead of the tunnel ID.

Configure Tunnel Attributes for GRE or UDP Dynamic Tunnels

You can configure GRE or UDP dynamic tunnels with specific attributes, to control how traffic is managed and routed within your network. By disabling anti-spoofing measures, you can ensure that legitimate traffic is not erroneously blocked.

The anti-spoofing configuration for GRE/UDP tunnels provides a powerful mechanism to secure your network against unauthorized traffic

This functionality allows you to configure the Tunnel Composite Nexthop (TCNH) to verify the source of incoming traffic by performing a reverse path lookup using the tunnel ID, thereby preventing malicious traffic from being routed through your network. Depending on the anti-spoofing setting, the system can either use the tunnel ID for validation or preserve the original WAN logical interface (IFL). Enabling anti-spoofing adds an essential layer of security, safeguarding your network against unauthorized access, while disabling it prioritizes performance in trusted environments.

The following example shows how to configure the policy options and the tunnel attributes. The policy options configuration involves specifying route filters and applying dynamic tunnel attributes to ensure that traffic destined for specific routes follows the defined policies. The routing options commands are used to set the dynamic tunnel types, disable anti-spoofing, and configure the dynamic tunnels based on next-hop addresses and source addresses.

To configure tunnel attributes and policy options configuration:

1. Configure a policy with `dynamic-tunnel-attributes` as the action associates the dynamic tunnel to the prefix list. The policy `from` action allows the creation of tunnel with specified attributes for any matching condition.

```
user@host# set policy-options policy-statement pol_tnl_gre from route-filter 10.1.255.2/32
exact
user@host# set policy-options policy-statement pol_tnl_gre then dynamic-tunnel-attributes
gre_tnl_attr
user@host# set policy-options policy-statement pol_tnl_gre then accept
```

2. Create dynamic tunnel profiles. The dynamic tunnel profile specifies the tunnel type and the anchor Packet Forwarding Engine information. Enable next-hop-based dynamic GRE tunnel configuration.

```
user@host# set routing-options dynamic-tunnels tunnel-attributes gre_tnl_attr dynamic-tunnel-
type GRE
user@host# set routing-options dynamic-tunnels tunnel-attributes gre_tnl_attr dynamic-tunnel-
anti-spoof off
user@host# set routing-options dynamic-tunnels gre next-hop-based-tunnel
user@host# set routing-options dynamic-tunnels demux0_1 source-address 10.1.1.2
```

```
user@host# set routing-options dynamic-tunnels demux0_1 gre
user@host# set routing-options dynamic-tunnels demux0_1 destination-networks 10.1.255.2/32
dyn-tunnel-attribute-policy pol_tnl_gre
```

Configuring IP Tunnel Interfaces

IN THIS SECTION

- [Configuring IPv6-over-IPv4 Tunnels | 95](#)
- [Example: Configuring an IPv6-over-IPv4 Tunnel | 96](#)

Configuring IPv6-over-IPv4 Tunnels

If you have a Tunnel PIC installed in your M Series or T Series router, you can configure IPv6-over-IPv4 tunnels. To define a tunnel, you configure a unicast tunnel across an existing IPv4 network infrastructure. IPv6/IPv4 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.

On SRX Series Firewalls, Generic Routing Encapsulation (GRE) and IP-IP tunnels use internal interfaces, `gr-0/0/0` and `ip-0/0/0`, respectively. The Junos OS creates these interfaces at system bootup; they are not associated with a physical interface.

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 Unicast Tunnels" on page 97](#). For an IPv6-over-IPv4 tunnel configuration example, see ["Tunnel Services Overview" on page 2](#).

SEE ALSO

| [Tunnel Services Overview | 2](#)

Example: Configuring an IPv6-over-IPv4 Tunnel

Configure a tunnel on both sides of the connection.

Configuration 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 2001:DB8::1:1/126;
      }
    }
  }
}
```

Configuration 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 2001:DB8::2:1/126;
      }
    }
  }
}
```


SEE ALSO

[Tunnel Services Overview](#) | 2

Filtering Unicast Packets Through Multicast Tunnel Interfaces

IN THIS SECTION

- [Configuring Unicast Tunnels](#) | 97
- [Examples: Configuring Unicast Tunnels](#) | 103
- [Restricting Tunnels to Multicast Traffic](#) | 105

Configuring Unicast Tunnels

IN THIS SECTION

- [Configuring a Key Number on GRE Tunnels](#) | 99
- [Enabling Packet Fragmentation on GRE Tunnels Prior to GRE Encapsulation](#) | 100
- [Specifying an MTU Setting for the Tunnel](#) | 101
- [Configuring a GRE Tunnel to Copy ToS Bits to the Outer IP Header](#) | 101
- [Enabling Fragmentation and Reassembly on Packets After GRE-Encapsulation](#) | 102
- [Support for IPv6 GRE tunnels](#) | 103

To configure a unicast tunnel, you configure a `gr-` interface (to use GRE encapsulation) or an `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 {
```

```

copy-tos-to-outer-ip-header;
reassemble-packets;
tunnel {
    allow-fragmentation;
    destination destination-address;
    do-not-fragment;
    key number;
    routing-instance {
        destination routing-instance-name;
    }
    source address;
    ttl number;
}
family family {
    address address {
        destination address;
    }
}
}
}

```

You can configure these statements at the following hierarchy levels:

- [edit interfaces]
- [edit logical-systems *logical-system-name* interfaces]

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



NOTE: On M Series and T Series routers, you can configure the interface on a service PIC or a tunnel PIC. On MX Series routers, configure the interface on a Multiservices DPC.

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 reverse path forwarding (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, but are supported on tunnel-pic interfaces.

However, class-of-service (CoS) information obtained from the GRE or IP-IP header is carried over the tunnel and is used by the re-entering packets. For more information, see the [Junos OS Class of Service User Guide for Routing Devices](#).

To prevent an invalid configuration, the Junos OS disallows setting the address specified by the source or destination statement at the `[edit interfaces gr-fpc/pic/port unit logical-unit-number tunnel]` hierarchy level to be the same as the interface's own subnet address, specified by the address statement at the `[edit interfaces gr-fpc/pic/port unit logical-unit-number family family-name]` hierarchy level.

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 include the appropriate statements at the `[edit protocols]` hierarchy level to enable Resource Reservation Protocol (RSVP), MPLS, and label-switched paths (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 ["Tunnel Services Overview" on page 2](#) and the [MPLS Applications User Guide](#).

For a GRE tunnel, the default is to set the ToS bits in the outer IP header to all zeros. To have the Routing Engine copy the ToS bits from the inner IP header to the outer, include the `copy-tos-bits-to-outer-ip-header` statement. (This inner-to-outer ToS bits copying is already the default behavior for IP-IP tunnels.)

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

Configuring a Key Number on GRE Tunnels

For Adaptive Services and Multiservices interfaces on M Series and T Series routers, you can assign a key value to identify an individual traffic flow within a GRE tunnel, as defined in RFC 2890, *Key and Sequence Number Extensions to GRE*. However, only one key is allowed for each tunnel source and destination pair.

Each IP version 4 (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 or Multiservices 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-systems *logical-system-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.

The following example illustrates the use of the `key` statement in a GRE tunnel configuration:

```
interfaces {
  gr-1/2/0 {
    unit 0 {
      tunnel {
        source 10.58.255.193;
        destination 10.58.255.195;
        key 1234;
      }
      ...
      family inet {
        mtu 1500;
        address 10.200.0.1/30;
        ...
      }
    }
  }
}
```

Enabling Packet Fragmentation on GRE Tunnels Prior to GRE Encapsulation

For GRE tunnel interfaces on Adaptive Services and Multiservices interfaces only, you can enable fragmentation of IPv4 packets before they are GRE-encapsulated 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-systems *logical-system-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 IP Security (IPsec) tunnels. For more information, see *Configuring IPsec Rules*.

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 99](#) and ["Enabling Packet Fragmentation on GRE Tunnels Prior to GRE Encapsulation" on page 100](#)), 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-system *logical-system-name* interfaces *gr-fpc/pic/port* unit *logical-unit-number* family inet]

For more information about MTU settings, see the [Junos OS Network Interfaces Library for Routing Devices](#).

Configuring a GRE Tunnel to Copy ToS Bits to the Outer IP Header

Unlike IP-IP tunnels, GRE tunnels do not copy the ToS bits to the outer IP header by default. To have the Routing Engine copy the inner ToS bits to the outer IP header (which is required for some tunneled routing protocols) on packets sent by the Routing Engine, include the `copy-tos-to-outer-ip-header`

statement at the logical unit hierarchy level of a GRE interface. This example copies the inner ToS bits to the outer IP header on a GRE tunnel:

```
[edit interfaces]
gr-0/0/0 {
  unit 0 {
    copy-tos-to-outer-ip-header;
    family inet;
  }
}
```

Enabling Fragmentation and Reassembly on Packets After GRE-Encapsulation

You can enable the fragmentation and reassembly of packets after they are GRE-encapsulated for a GRE tunnel. When the size of a GRE-encapsulated packet is greater than the MTU of a link that the packet passes through, the GRE-encapsulated packet is fragmented. You configure the GRE interface at the endpoint of the tunnel to reassemble the fragmented GRE-encapsulated packets before they are processed further on the network.

For each tunnel you configure on an interface, you can enable or disable fragmentation of GRE-encapsulated packets by including the `allow-fragmentation` or `do-not-fragment` statement:

```
allow-fragmentation;
do-not-fragment;
```

You can configure these statements at the following hierarchy levels:

- [edit interfaces *interface-name* unit *logical-unit-number* tunnel]
- [edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number* tunnel]

If you configure `allow-fragmentation` on a tunnel, the DF bit is not set in the outer IP header of the GRE-encapsulated packet, enabling fragmentation. By default, GRE-encapsulated packets that exceed the MTU size of a link are not fragmented and are dropped.

To enable reassembly of fragmented GRE-encapsulated packets on the GRE interface at the endpoint of the tunnel, include the `reassemble-packets` statement:

```
reassemble-packets;
```

You can configure this statement at the following hierarchy levels:

- [edit interfaces *interface-name* unit *logical-unit-number*]
- [edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number*]

Starting with Junos OS Release 17.3R1, you can configure fragmentation and reassembly of GRE-encapsulated packets on GRE tunnel interfaces on MX Series routers with MPC7Es, MPC8Es, and MPC9Es.

Starting with Junos OS Release 17.1R1, you can configure fragmentation and reassembly of GRE-encapsulated packets on GRE tunnel interfaces on MX Series routers with MPC2E-NGs, MPC3E-NGs, MPC5Es, and MPC6Es.

Starting with Junos OS Release 14.2, you can configure fragmentation and reassembly of GRE-encapsulated packets on GRE tunnel interfaces on MX Series routers with MPC1s, MPC2s, MPC3s, MPC4s, and MPC-16X10GEs.

In Junos OS Release 14.1 and earlier, fragmentation and reassembly of GRE-encapsulated packets is supported only on MX Series routers with MS-DPCs.

Support for IPv6 GRE tunnels

Starting in Junos OS Release 17.3R1, you can configure IPv6 generic routing encapsulation (GRE) tunnel interfaces on MX Series routers. This lets you run a GRE tunnel over an IPv6 network. Packet payload families that can be encapsulated within the IPv6 GRE tunnels include IPv4, IPv6, MPLS, and ISO. Fragmentation and reassembly of the IPv6 delivery packets is not supported.

To configure an IPv6 GRE tunnel interface, specify IPv6 addresses for source and destination at the [interfaces gr-0/0/0 unit 0 tunnel] hierarchy level, specify family inet6 at the [interfaces gr-0/0/0 unit 0] hierarchy level, and specify an IPv6 address for address at the [interfaces gr-0/0/0 unit 0 family inet6] hierarchy level.

SEE ALSO

[Tunnel Services Overview | 2](#)

Examples: 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;
}
}

```

Configure numbered tunnel interfaces by including an address at the [edit interfaces ip-0/3/0 unit (0 | 1) family inet] hierarchy level:

```

[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;
        }
    }
}
}

```


Configure an MPLS over GRE tunnel by including the `family mpls` statement at the `[edit interfaces gr-1/2/0 unit 0]` hierarchy level:

```
[edit 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 mpls;
  }
}
```

SEE ALSO

| [Tunnel Services Overview](#) | 2

Restricting Tunnels to Multicast Traffic

For interfaces that carry IPv4 or IP version 6 (IPv6) traffic, you can configure a tunnel interface to allow multicast traffic only. To configure a multicast-only tunnel, include the `multicast-only` statement:

```
multicast-only;
```

You can configure this statement at the following hierarchy levels:

- `[edit interfaces interface-name unit logical-unit-number family family]`
- `[edit logical-systems logical-system-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 (mt) interfaces only.



NOTE: If your router has a Tunnel Services PIC, the Junos OS automatically configures one multicast tunnel interface (mt) for each virtual private network (VPN) you configure. You do not need to configure multicast tunnel interfaces.

SEE ALSO

[Tunnel Services Overview](#) | 2

Connecting Logical Systems Using Logical Tunnel Interfaces

IN THIS SECTION

- [Configuring Logical Tunnel Interfaces](#) | 107
- [Guidelines for Configuring Logical Tunnels on MX Series Routers](#) | 108
- [Configuring Bridge Domain over Logical Tunnel Physical Interface](#) | 110
- [Configuring Recycle Bandwidth for Logical Tunnel Physical Interface](#) | 111
- [Layer 3 VPN Support over Logical Tunnel Interfaces](#) | 112
- [Example: Configuring Logical Tunnels](#) | 114
- [Configuring an Interface in the VRF Domain to Receive Multicast Traffic](#) | 115
- [Redundant Logical Tunnels Overview](#) | 118
- [Configuring Redundant Logical Tunnels](#) | 121
- [Configuring Single Link Targeting for Redundant Logical Tunnels](#) | 122
- [Configuring Minimum Active Links for Redundant Logical Tunnels](#) | 122
- [Example: Configuring Redundant Logical Tunnels](#) | 123

Configuring Logical Tunnel Interfaces

IN THIS SECTION

- [Connecting Logical Systems | 107](#)

Logical tunnel (lt-) interfaces provide quite different services depending on the host router:

- On M Series, MX Series, and T Series routers, logical tunnel interfaces allow you to connect logical systems, virtual routers, or VPN instances. M Series and T Series routers must be equipped with a Tunnel Services PIC or an Adaptive Services Module (only available on M7i routers). MX Series routers must be equipped with a Trio MPC/MIC module. For more information about connecting these applications, see the [Junos OS VPNs Library for Routing Devices](#).
- On SRX Series Firewalls, the logical tunnel interface is used to interconnect logical systems. See the [Logical Systems and Tenant Systems User Guide for Security Devices](#) for information about using the logical tunnel interface on the SRX Series.

Connecting Logical Systems

To connect two logical systems, you configure a logical tunnel interface on both logical systems. 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 systems, configure the logical tunnel interface by including the `lt-fpc/pic/port` statement:

```
lt-fpc/pic/port {
  unit logical-unit-number {
    encapsulation encapsulation;
    peer-unit unit-number;    # peering logical system unit number
    dlci dlci-number;
    family (inet | inet6 | iso | mpls);
  }
}
```

You can include this statement at the following hierarchy levels:

- [edit interfaces]

- [edit logical-systems *logical-system-name* interfaces]

When configuring logical tunnel interfaces, note the following:

- You can configure each logical tunnel interface with one of the following encapsulation types: Ethernet, Ethernet circuit cross-connect (CCC), Ethernet VPLS, Frame Relay, Frame Relay CCC, VLAN, VLAN CCC, or VLAN VPLS.
- You can configure the IP, IPv6, International Organization for Standardization (ISO), or MPLS protocol family.
- Do not reconfigure a logical tunnel interface that is an anchor point with pseudowire devices stacked above it unless you first deactivate all broadband subscribers that are using the pseudowire subscriber interface.
- The peering logical interfaces must belong to the same logical tunnel interface derived from the Tunnel Services PIC or Adaptive Services Module.
- 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, Multiservices, or Link Services PICs (but they are supported on the Adaptive Services Module on M7i routers, as noted above).
- On M Series routers other than the M40e router, logical tunnel interfaces require an Enhanced Flexible PIC Concentrator (FPC).
- On MX Series routers, logical tunnel interfaces require Trio MPC/MIC modules. They do not require a Tunnel Services PIC in the same system.

SEE ALSO

[Tunnel Services Overview](#) | 2

Guidelines for Configuring Logical Tunnels on MX Series Routers

When you configure a logical tunnel on an MX series router which has one of the peer configured in layer 2 mode, ensure that the peer layer 2 logical tunnel is part of a bridge domain or VPLS instance, for bidirectional traffic flow.

To configure a logical tunnel with bridge encapsulation, you must first configure the logical tunnel to be part of the bridge domain. The following sample configuration allows you to configure a logical tunnel, lt-2/1/0.3 with bridge encapsulation.

```

user@host# edit bridge-domains {
    bd1 {
        domain-type bridge;
        vlan-id 1
    }
}

user@host# edit chassis
lt-2/1/0 {
    unit 3 {
        description "MPLS port mirroring Bridge ingress interface";
        encapsulation ethernet-bridge;
        mtu 4500;
        peer-unit 4;
        family bridge {
            interface-mode access;
            vlan-id 1;
        }
    }

    unit 4 {
        description "MPLS Port mirroring L2/CCC egress interface";
        encapsulation ethernet-ccc;
        mtu 4500;
        peer-unit 3;
        family ccc {
            filter {
                input HighPriority;
            }
        }
    }
}

```

Configuring Bridge Domain over Logical Tunnel Physical Interface

On the ACX7K series routers, you can configure a logical tunnel physical interface (IFD) to communicate between two bridge domains (BDs). For this logical tunnel physical interface, you can create logical tunnel interfaces and map the logical tunnel interfaces to each service or bridge domain. Now, the traffic can be forwarded from one service to other using these logical tunnel interfaces. You can also configure bandwidth per logical tunnel interface.

1. Encapsulate the logical interface for service provider style bridging configuration.

```
[edit]
user@host# set interfaces et-0/0/2 flexible-vlan-tagging
user@host# set interfaces et-0/0/2 encapsulation flexible-ethernet-services
user@host# set interfaces et-0/0/3 flexible-vlan-tagging
user@host# set interfaces et-0/0/3 encapsulation flexible-ethernet-services
user@host# set interfaces et-0/0/2 unit 0 encapsulation ethernet-bridge vlan-id 100
user@host# set interfaces et-0/0/3 unit 0 encapsulation ethernet-bridge vlan-id 100
```

2. Create BD1 and associate logical tunnel interface.

```
[edit]
user@host# set vlans bd1 interface et-0/0/3.0
user@host# set vlans bd1 interface et-0/0/2.0
user@host# set vlans bd1 interface lt-0/0/0:3.0
```

3. Encapsulate the logical interface for service provider style bridging configuration.

```
[edit]
user@host# set interfaces et-0/0/6 flexible-vlan-tagging
user@host# set interfaces et-0/0/6 encapsulation flexible-ethernet-services
user@host# set interfaces et-0/0/7 flexible-vlan-tagging
user@host# set interfaces et-0/0/7 encapsulation flexible-ethernet-services
user@host# set interfaces et-0/0/6 unit 0 encapsulation ethernet-bridge vlan-id 100
user@host# set interfaces et-0/0/7 unit 0 encapsulation ethernet-bridge vlan-id 100
```

4. Create BD2 and associate logical tunnel interface.

```
[edit]
user@host# set vlans bd2 vlan-id 100
user@host# set vlans bd2 interface et-0/0/7.0
user@host# set vlans bd2 interface et-0/0/6.0
user@host# set vlans bd2 interface lt-0/0/0:3.1
```

Configuring Recycle Bandwidth for Logical Tunnel Physical Interface

Logical tunnel interfaces on ACX7K series routers use the internal recycle interfaces to recirculate traffic between two interconnecting services.

The recycle mechanism has two modes of operation:

- Default Recycle Bandwidth Mode
- Configurable Recycle Bandwidth Mode

For information on recycle infrastructure in ACX7K series platforms, see [Recycle Bandwidth Management](#).

By default, logical tunnel interfaces operates in the default mode. To enable configuration mode for logical tunnel interfaces recycle application using following sample configuration:

1. Configure percentage of the calendar bandwidth for logical tunnel application.

```
[edit]
user@host# set system packet-forwarding-options recycle-bandwidth-profiles prof1 tunnel-
services 80
```

In this example, you reserve 80% of the calendar bandwidth for logical tunnel application. In this case, 80% of 800Gbps, i.e., 640Gbps is reserved for logical tunnel application.

2. Apply the configured bandwidth to the logical tunnel application.

```
[edit]
user@host# set system packet-forwarding-options recycle-bandwidth profile prof1
```

In the default mode, logical tunnel bandwidth is shared with other recycle applications in best effort mode. In the configuration mode, the sum total of all logical tunnel interfaces' bandwidth is capped by the total logical tunnel recycle application bandwidth. If the sum of configured bandwidth for all logical tunnel interfaces is larger than the bandwidth derived by logical tunnel recycle application then, the sum of logical tunnel interfaces' bandwidth is limited to logical tunnel recycle application value.

For example, configure bandwidth of 10Gbps for logical tunnel1 and bandwidth of 100Gbps for logical tunnel2. The logical tunnel application percentage is equal to 100 Gbps. Sum of bandwidth of logical tunnel1 and of logical tunnel2 will be 100 Gbps and not 110 Gbps. In such cases, logical tunnel recycle application bandwidth is distributed in the ratio of individual logical tunnel interface bandwidth. In this case, 1:10 ratio of 100 Gbps.

Layer 3 VPN Support over Logical Tunnel Interfaces

Starting in Junos OS Evolved Release 24.1R1, we support layer 3 VPN service over logical tunnel interface on the ACX7K Series routers. The feature includes:

- VRF over logical tunnel interface
- Stitching of layer 3 VPN and layer 2 services through logical tunnel interface

The following sample configuration allows you to configure layer 3 VPN over a logical tunnel interface:

```
user@host# edit chassis {
lt-0/0/1:0 {
  unit 0 {
    vlan-id 10;
    encapsulation vlan-bridge;
    peer-unit 1;
  }
  unit 1 {
    vlan-id 10;
    family inet {
      address 100.1.1.100/24;
    }
    family inet6 {
      address 2001::1/128;
    }
    peer-unit 0;
  }
}
```



```

routing-instances {
  vpn100 {
    instance-type vrf;
    interface lt-0/0/1:0.1;
    route-distinguisher <...>;
    vrf-import vpn100-imp;
    vrf-export vpn100-exp;
    vrf-table-label;
  }
}

policy-options {
  policy-statement vpn100-exp {
    term 1 {
      from {
        <...>;
      }
      then {
        community add vpn100;
        accept;
      }
    }
  }
  policy-statement vpn100-imp {
    term 1 {
      from {
        <...>;
        community vpn100;
      }
      then {
        accept;
      }
    }
  }
  community vpn100 members <...>;
}

```

Example: Configuring Logical Tunnels

Configure three logical tunnels:

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

```

SEE ALSO

| [Tunnel Services Overview](#) | 2

Configuring an Interface in the VRF Domain to Receive Multicast Traffic

IN THIS SECTION

- [Configuring a Proxy Logical Interface in the Global Domain](#) | 116
- [Associating the Proxy Logical Interface to a Logical Interface in a VRF Domain](#) | 117
- [Limitations](#) | 117

You can configure an ACX Series router to receive multicast traffic in a VRF domain. In an IPTV solution, IPTV sources and receivers can be spread across different end points of a network in a VRF domain. To receive the multicast traffic at the receiver's side, it is necessary for the multicast traffic to be tunneled across the network to reach the end receiving device or the subscriber. This tunneling is usually done using the Multicast Virtual Private Network (MVPN) technology.

ACX Series routers do not support MVPN technology. An alternate method for receiving the multicast traffic in the VRF domain in ACX Series router is by associating a global logical interface to a logical interface in the VRF domain. The global logical interface acts as a proxy for receiving the multicast traffic on the logical interface in the VRF domain. To associate a global logical interface to a logical interface in

the VRF domain, you need to configure an IRB interface in a global domain to act as a proxy for the logical interface in the VRF domain.

Configuring a Proxy Logical Interface in the Global Domain

To configure a proxy logical interface in the global domain, you need to create logical tunnel (lt-) interface and IRB interface and then associate the IRB interface to a bridge domain. The following is an example to configure a proxy logical interface in the global domain:

1. Create an logical tunnel (lt-) interface.

```
[edit]
user@host# set chassis aggregated-devices ethernet device-count 1
user@host# set chassis fpc 0 pic 0 tunnel-services bandwidth 1g
user@host# set interfaces lt-0/0/10 unit 0 encapsulation vlan-bridge
user@host# set interfaces lt-0/0/10 unit 0 vlan-id 101
user@host# set interfaces lt-0/0/10 unit 0 peer-unit 1
user@host# set interfaces lt-0/0/10 unit 1 encapsulation vlan-ccc
user@host# set interfaces lt-0/0/10 unit 1 vlan-id 101
user@host# set interfaces lt-0/0/10 unit 1 peer-unit 0
```

2. Create an IRB interface.

```
[edit]
user@host# set interfaces irb unit 0 family inet address 192.168.1.2/24
```

3. Associate the IRB interface to a bridge domain.

```
[edit]
user@host# set bridge-domains b1 vlan-id 101
user@host# set bridge-domains b1 interface lt-0/0/10.0
user@host# set bridge-domains b1 routing-interface irb.0
```

Associating the Proxy Logical Interface to a Logical Interface in a VRF Domain

To associate the proxy logical interface to a logical interface in a VRF domain, you need to run the following PFE commands:

- `test pfe acx vrf-mc-leak enable`—Enables proxy association.
- `test pfe acx entry add VRF-logical-interface-name logical-tunnel-logical-interface-name IRB-logical-interface-name IRB-IP-address + 1`—Creates an association between proxy logical interface and the logical interface in a VRF domain.
- `test pfe acx vrf-mc-leak disable`—Disables proxy association.
- `test pfe acx entry del VRF-logical-interface-name logical-tunnel-logical-interface-name IRB-logical-interface-name IRB-IP-address + 1`—Deletes the association between the proxy logical interface and the logical interface in a VRF domain.
- `show pfe vrf-mc-leak`—Displays the association entries between proxy logical interface and the logical interface in a VRF domain.



NOTE: When the router or PFE is rebooted, the proxy associations of logical interfaces is removed and you need to once again create the proxy associations of logical interface.

Limitations

The following limitations need to be considered for receiving multicast traffic in a VRF domain:

- Maximum of 5 proxy associations of logical interfaces can be configured.
- VRF IPv6 multicast is not supported.
- AE interface as a VRF interface (requesting multicast traffic) is not be supported.
- Multicast traffic cannot be forwarded from the logical interface in a VRF domain if the first hop router is an ACX router.

Redundant Logical Tunnels Overview

IN THIS SECTION

- [Redundant Logical Tunnel Configuration | 119](#)
- [Single Link Targeting | 120](#)
- [Minimum Active Links | 120](#)
- [Redundant Logical Tunnel Failure Detection and Failover | 120](#)

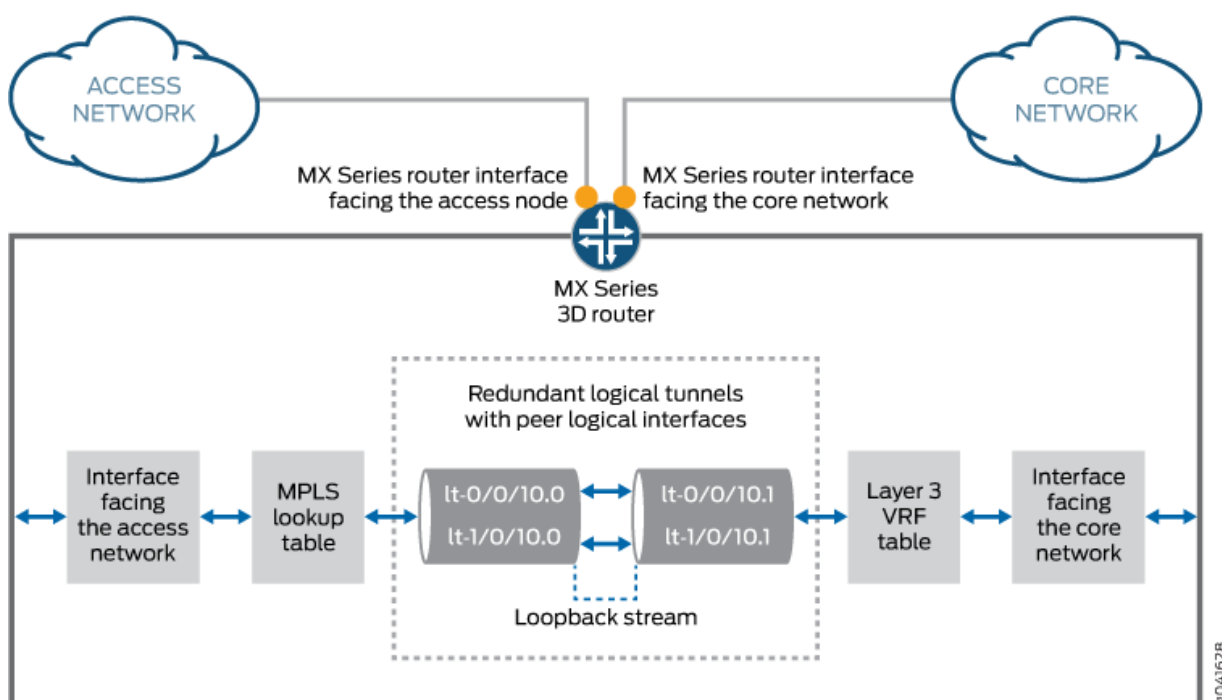
You can connect two devices, such as an access-facing device and a core-facing device, through logical tunnels. To provide redundancy for the tunnels, you can create and configure multiple physical logical tunnels and add them to a virtual redundant logical tunnel.



NOTE: Redundant logical tunnels are supported only on MX Series routers with MPCs. Starting in Junos OS Release 18.4R3, redundant logical tunnels are supported on MX Series Virtual Chassis..

For example, in an MPLS access network, you can configure multiple pseudowires between an access node and an MX Series router with MPCs and add them to a redundant logical tunnel. You can then add multiple logical tunnels to the redundant logical tunnel. [Figure 4 on page 119](#) shows a redundant logical tunnel between the access node and the MX Series router.

Figure 4: Redundant Logical Tunnels



The redundant logical tunnel has peer logical interfaces at each end, rlt0.0 and rlt0.1. You can configure router features on these interfaces for the redundant logical tunnel and its members.

Each member logical tunnel has peer logical interfaces. In [Figure 4 on page 119](#), lt-0/0/10.0 and lt-0/0/10.1 are peers.

The MX Series router performs IP lookup in the Layer 3 VPN routing and forwarding (VRF) table on the router where the pseudowires that are grouped in logical tunnels terminate.

Redundant Logical Tunnel Configuration

In Junos OS Releases 14.1R1 and earlier, you can create up to 16 redundant logical tunnels, depending on the number of Packet Forwarding Engines and the number of loopback interfaces on each Packet Forwarding Engine on your device. Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255.

You can add up to 32 logical tunnels as members of a redundant logical tunnel.

When you add more than two members to the redundant logical tunnel, they are in active mode. The traffic is load-balanced by default over all the tunnel members. You can also configure your RLT for single-link targeting and specify a minimum number of active links for the RLT.

When you add only two members to the redundant logical tunnel, you can configure the members in one of these ways:

- Both members in active mode
- One member in active mode and the other in backup mode

Single Link Targeting

You can configure your RLT anchor to use single link targeting. In this mode, all traffic flowing through your pseudowire or PWHT interface will be directed through just one link in the *r/t* bundle. If the targeted link goes down, all subscribers on the RLT will be terminated.

Minimum Active Links

In this mode, you can specify the minimum number of links that must be active for the RLT interface to remain up. If the number of active links on the RLT drops below the minimum, the RLT will go down. Any pseudowire and PWHT interfaces stacked on the RLT will also go down, and all subscribers will be terminated.

Redundant Logical Tunnel Failure Detection and Failover

A logical tunnel fails and is removed from the redundant logical tunnel group, and the backup logical tunnel becomes active due to one of these events:

- A hardware failure on the MPC module occurs.
- An MPC failure occurs due to a microkernel crash.
- The MPC module is administratively shut down and removed from the redundant logical tunnel.
- A power failure on the MPC module occurs.



NOTE: You can decrease the time it takes for failure detection and failover to occur. Configure the `enhanced-ip` statement at the `[edit chassis network-services]` hierarchy level to enable Packet Forwarding Engine liveliness detection.

SEE ALSO

Pseudowire Subscriber Logical Interfaces Overview

Configuring a Pseudowire Subscriber Logical Interface Device

Configuring Redundant Logical Tunnels

Use redundant logical tunnels to provide redundancy for logical tunnels between two devices, such as an access-facing device and a core-facing device.

When configuring redundant logical tunnel interfaces, note the following:

- Starting in Junos OS Release 13.3, you can configure redundant logical tunnels only on MX Series routers with MPCs.

In Junos OS Releases 14.1R1 and earlier, you can create up to 16 redundant logical tunnels, depending on the number of Packet Forwarding Engines and the number of loopback interfaces on each Packet Forwarding Engine on your device. Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255. The command is shown below.

```
set chassis redundancy-group interface-type redundant-logical-tunnel device-count [number];
```

You can add up to 32 logical tunnels as members.

- When a logical tunnel with an existing configuration joins a redundant logical tunnel, you must configure the redundant logical tunnel with the settings from the existing configuration.
- You can add member logical tunnels to a parent logical tunnel for redundancy.
- When you add more than two logical tunnels to the redundant logical tunnel, the members are in active mode by default.
- When you add only two members, you can configure the members in one of these ways:
 - Both members in active mode
 - One member in active mode and the other in backup mode

To configure a redundant logical tunnel between two devices:

1. Create the logical tunnel and redundant logical tunnel interfaces.

```
[edit chassis]
user@host# set redundancy-group interface-type redundant-logical-tunnel device-count count
user@host# set fpc slot-number pic number tunnel-services bandwidth 1g
```

2. Bind the member logical tunnels to the redundant logical tunnel.

```
[edit interfaces]
user@host# set interface-name redundancy-group member-interface interface-name
```

3. Configure the redundant logical tunnel interfaces.
4. Attach the redundant logical tunnel interface to a Layer 2 circuit.
5. Add the peer redundant logical tunnel interface to a Layer 3 VRF instance.
6. Configure MPLS and LDP in the pseudowires and the Layer 3 VPN.

```
[edit protocols]
user@host# set mpls no-cspf
user@host# set mpls interface all
user@host# set ldp interface all
```

7. Configure BGP in the Layer 3 VPN.
8. Configure OSPF on the core-facing interfaces and the router local loopback interface.
9. Set the policy options for BGP.
10. Set the router ID and the autonomous system (AS) number.

Configuring Single Link Targeting for Redundant Logical Tunnels

Use single link targeting to direct all traffic on a redundant logical tunnel to one specific logical tunnel interface.

When single link targeting is active, all subscriber traffic carried on the redundant logical tunnel will be terminated if that link goes down.

To configure single link targeting:

Configure the interface to use `single-targeted-link` under the `targeted-options` entry and specify the logical tunnel link to target.

```
[edit interfaces interface-name]
user@host# set targeted-options single-targeted-link interface-name
```

Configuring Minimum Active Links for Redundant Logical Tunnels

You can use Minimum Active Links to specify the number of tunnel links that must be active for the redundant logical tunnel to remain up.

When minimum active links is configured, the redundant logical tunnel (RLT) will go down if the number of active links falls below the configured number. When the RLT goes down, all subscriber traffic stacked on top of the RLT will be terminated, including pseudowire and PWHT traffic.

To configure minimum active links:

Configure the redundant logical tunnel interface with the `minimum-links` option. This option is found under the `redundancy-group` hierarchy.

```
[edit interfaces rlt-interface]
user@host# set redundancy-group minimum-links number-of-links
```

Example: Configuring Redundant Logical Tunnels

IN THIS SECTION

- [Requirements | 123](#)
- [Overview | 123](#)
- [Configuration | 125](#)
- [Verification | 133](#)

This example shows how to configure redundant logical tunnels in an MPLS access network.

Requirements

In Junos OS Release 13.3 or later, you can configure redundant logical tunnels only on MX Series routers with MPCs.

Overview

IN THIS SECTION

- [Topology | 124](#)

When a logical tunnel with an existing configuration joins a redundant logical tunnel, you must configure the redundant logical tunnel with the settings from the existing configuration.

You can add member logical tunnels to a parent logical tunnel for redundancy.

On MX Series routers with MPCs, you can configure redundant logical tunnels as follows:

- In Junos OS Releases 14.1R1 and earlier, you can create up to 16 redundant logical tunnels, depending on the number of Packet Forwarding Engines and the number of loopback interfaces on each Packet Forwarding Engine on your device. Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255. The command is shown below.

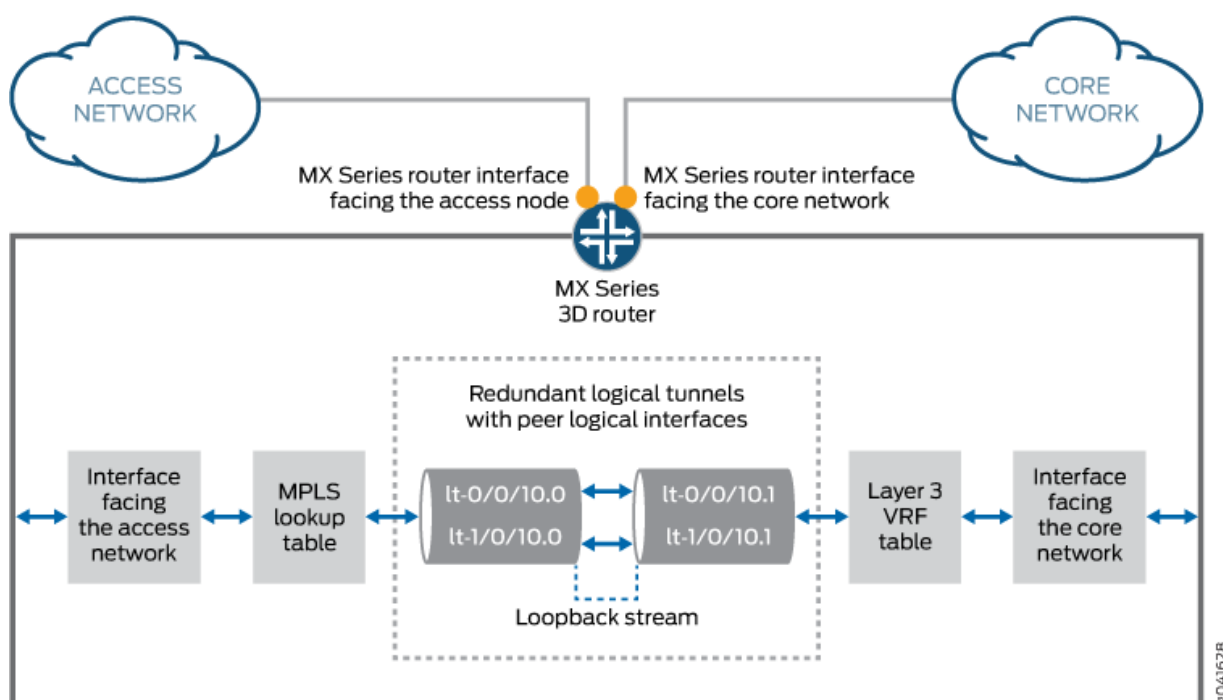
```
set chassis redundancy-group interface-type redundant-logical-tunnel device-count [number];
```

- You can add up to 32 logical tunnels as members.
- When you add more than two logical tunnels to a redundant logical tunnel, the members are in active mode by default.
- When you add only two members, you can configure the members in one of these ways:
 - Both members in active mode
 - One member in active mode and the other in backup mode

Topology

[Figure 5 on page 125](#) shows a redundant logical tunnel between the access node and the MX Series router in an MPLS access network.

Figure 5: Redundant Logical Tunnels



The redundant logical tunnel has peer logical interfaces at each end, **lt0.0** and **lt0.1**. You can configure router features on these interfaces for the redundant logical tunnel and its members.

Each member logical tunnel has peer logical interfaces on the access-facing and core-facing devices. In [Figure 5 on page 125](#), **lt-0/0/10.0** and **lt-0/0/10.1** are peers.

The MX Series router performs IP lookup in the Layer 3 VPN routing and forwarding (VRF) table on the router where the pseudowires that are grouped in logical tunnels terminate.

Configuration

IN THIS SECTION

- [CLI Quick Configuration | 126](#)
- [Procedure | 127](#)
- [Results | 129](#)

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level.

```
set chassis redundancy-group interface-type redundant-logical-tunnel device-count 4
set chassis fpc 1 pic 0 tunnel-services bandwidth 1g
set chassis fpc 2 pic 2 tunnel-services bandwidth 1g
set interfaces rlt0 redundancy-group member-interface lt-1/0/10
set interfaces rlt0 redundancy-group member-interface lt-2/0/10
set interfaces rlt0 unit 0 description "Towards Layer 2 Circuit"
set interfaces rlt0 unit 0 encapsulation vlan-ccc
set interfaces rlt0 unit 0 vlan-id 600
set interfaces rlt0 unit 0 peer-unit 1
set interfaces rlt0 unit 0 family ccc
set interfaces rlt0 unit 1 description "Towards Layer 3 VRF"
set interfaces rlt0 unit 1 encapsulation vlan
set interfaces rlt0 unit 1 vlan-id 600
set interfaces rlt0 unit 1 peer-unit 0
set interfaces rlt0 unit 1 family inet address 10.10.10.2/24
set protocols l2circuit neighbor 192.0.2.2 interface rlt0.0 virtual-circuit-id 100
set protocols l2circuit neighbor 192.0.2.2 interface rlt0.0 no-control-word
set routing-instances pe-vrf instance-type vrf
set routing-instances pe-vrf interface rlt0.1
set routing-instances pe-vrf route-distinguisher 65056:1
set routing-instances pe-vrf vrf-import VPN-A-Import
set routing-instances pe-vrf vrf-export VPN-A-Export
set routing-instances pe-vrf vrf-table-label
set routing-instances pe-vrf protocols ospf export VPN-A-Import
set routing-instances pe-vrf protocols ospf area 0.0.0.0 interface rlt0.1
set protocols mpls no-cspf
set protocols mpls interface all
set protocols ldp interface all
set protocols bgp export local-routes
set protocols bgp group internal type internal
set protocols bgp group internal local-address 198.51.100.3
set protocols bgp group internal family inet any
set protocols bgp group internal family inet-vpn unicast
set protocols bgp group internal neighbor 203.0.113.4
set protocols ospf area 0.0.0.0 interface ge-5/3/8.0
set protocols ospf area 0.0.0.0 interface ge-5/2/5.0
```

```

set protocols ospf area 0.0.0.0 interface lo0.3 passive
set policy-options policy-statement VPN-A-Export term a then community add VPN-A
set policy-options policy-statement VPN-A-Export term a then accept
set policy-options policy-statement VPN-A-Export term b then reject
set policy-options policy-statement VPN-A-Import term a from protocol bgp
set policy-options policy-statement VPN-A-Import term a from community VPN-A
set policy-options policy-statement VPN-A-Import term a then accept
set policy-options policy-statement VPN-A-Import term b then reject
set policy-options policy-statement local-routes then accept
set policy-options community VPN-A members target:100:100
set routing-options router-id 198.51.100.3
set routing-options autonomous-system 65056

```

Procedure

Step-by-Step Procedure

In this example, all the logical tunnels are in active mode.

1. Create the logical tunnel and redundant logical tunnel interfaces.

```

[edit chassis]
user@host# set redundancy-group interface-type redundant-logical-tunnel device-count 4
user@host# set fpc 1 pic 0 tunnel-services bandwidth 1g
user@host# set fpc 2 pic 2 tunnel-services bandwidth 1g

```

2. Bind the member logical tunnels to the redundant logical tunnel.

```

[edit interfaces]
user@host# set rlt0 redundancy-group member-interface lt-1/0/10
user@host# set rlt0 redundancy-group member-interface lt-2/0/10

```

3. Configure the redundant logical tunnel interfaces.

```

[edit interfaces]
user@host# set rlt0 unit 0 description "Towards Layer 2 Circuit"
user@host# set rlt0 unit 0 encapsulation vlan-ccc
user@host# set rlt0 unit 0 vlan-id 600
user@host# set rlt0 unit 0 peer-unit 1

```

```

user@host# set rlt0 unit 0 family ccc
user@host# set rlt0 unit 1 description "Towards Layer 3 VRF"
user@host# set rlt0 unit 1 encapsulation vlan
user@host# set rlt0 unit 1 vlan-id 600
user@host# set rlt0 unit 1 peer-unit 0
user@host# set rlt0 unit 1 family inet address 10.10.10.2/24

```

4. Attach rlt0.0 to a Layer 2 circuit.

```

[edit protocols]
user@host# set l2circuit neighbor 192.0.2.2 interface rlt0.0 virtual-circuit-id 100
user@host# set l2circuit neighbor 192.0.2.2 interface rlt0.0 no-control-word

```

5. Add rlt0.1 to a Layer 3 VRF instance.

```

[edit routing-instances]
user@host# set pe-vrf instance-type vrf
user@host# set pe-vrf interface rlt0.1
user@host# set pe-vrf route-distinguisher 65056:1
user@host# set pe-vrf vrf-import VPN-A-Import
user@host# set pe-vrf vrf-export VPN-A-Export
user@host# set pe-vrf vrf-table-label
user@host# set pe-vrf protocols ospf export VPN-A-Import
user@host# set pe-vrf protocols ospf area 0.0.0.0 interface rlt0.1

```

6. Configure MPLS and LDP in the pseudowires and the Layer 3 VPN.

```

[edit protocols]
user@host# set mpls no-cspf
user@host# set mpls interface all
user@host# set ldp interface all

```

7. Configure BGP in the Layer 3 VPN.

```

[edit protocols]
user@host# set bgp export local-routes
user@host# set bgp group internal type internal
user@host# set bgp group internal local-address 198.51.100.3
user@host# set bgp group internal family inet any

```



```
user@host# set bgp group internal family inet-vpn unicast
user@host# set bgp group internal neighbor 203.0.113.4
```

8. Configure OSPF on the core-facing interfaces and the router local loopback interface.

```
[edit protocols]
user@host# set ospf area 0.0.0.0 interface ge-5/3/8.0
user@host# set ospf area 0.0.0.0 interface ge-5/2/5.0
user@host# set ospf area 0.0.0.0 interface lo0.3 passive
```

9. Set the policy options for BGP.

```
[edit policy-options]
user@host# set policy-statement VPN-A-Export term a then community add VPN-A
user@host# set policy-statement VPN-A-Export term a then accept
user@host# set policy-statement VPN-A-Export term b then reject
user@host# set policy-statement VPN-A-Import term a from protocol bgp
user@host# set policy-statement VPN-A-Import term a from community VPN-A
user@host# set policy-statement VPN-A-Import term a then accept
user@host# set policy-statement VPN-A-Import term b then reject
user@host# set policy-statement local-routes then accept
user@host# set community VPN-A members target:100:100
```

10. Set the router ID and the autonomous system (AS) number.

```
[edit routing-options]
user@host# set router-id 198.51.100.3
user@host# set autonomous-system 65056
```

Results

From configuration mode, confirm your configuration by entering the following commands:

- show chassis
- show interfaces
- show policy-options
- show protocols

- show routing-instances
- show routing-options

If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show chassis
redundancy-group {
  interface-type {
    redundant-logical-tunnel {
      device-count 4;
    }
  }
}
fpc 1 {
  pic 0 {
    tunnel-services {
      bandwidth 1g;
    }
  }
}
fpc 1 {
  pic 2 {
    tunnel-services {
      bandwidth 1g;
    }
  }
}
```

```
user@host# show interfaces rlt0
redundancy-group {
  member-interface lt-1/0/10;
  member-interface lt-2/0/10;
}
unit 0 {
  description "Towards Layer 2 Circuit";
  encapsulation vlan-ccc;
  vlan-id 600;
  peer-unit 1;
  family ccc;
```

```

}
unit 1 {
    description "Towards Layer 3 VRF";
    encapsulation vlan;
    vlan-id 600;
    peer-unit 0;
    family inet {
        address 10.10.10.2/24;
    }
}

```

```

user@host# show protocols l2circuit
neighbor 192.0.2.2 {
    interface rlt0.0 {
        virtual-circuit-id 100;
        no-control-word;
    }
}

```

```

user@host# show protocols
mpls {
    no-cspf;
    interface all;
}
bgp {
    export local-routes;
    group internal {
        type internal;
        local-address 198.51.100.3;
        family inet {
            any;
        }
        family inet-vpn {
            unicast;
        }
        neighbor 203.0.113.4;
    }
}
ospf {
    area 0.0.0.0 {

```

```

        interface ge-5/3/8.0;
        interface ge-5/2/5.0;
        interface lo0.3 {
            passive;
        }
    }
}
ldp {
    interface all;
}
l2circuit {
    neighbor 192.0.2.2 {
        interface rlt0.0 {
            virtual-circuit-id 100;
            no-control-word;
        }
    }
}
}

```

```

user@host# routing-instances
pe-vrf {
    instance-type vrf;
    interface rlt0.1;
    route-distinguisher 65056:1;
    vrf-import VPN-A-Import;
    vrf-export VPN-A-Export;
    vrf-table-label;
    protocols {
        ospf {
            export VPN-A-Import;
            area 0.0.0.0 {
                interface rlt0.1;
            }
        }
    }
}
}

```

```

user@host# policy-options
policy-statement VPN-A-Export {
    term a {

```

```

        then {
            community add VPN-A;
            accept;
        }
    }
    term b {
        then reject;
    }
}
policy-statement VPN-A-Import {
    term a {
        from {
            protocol bgp;
            community VPN-A;
        }
        then accept;
    }
    term b {
        then reject;
    }
}
policy-statement local-routes {
    then accept;
}
community VPN-A members target:100:100;

```

```

user@host# routing-options
router-id 198.51.100.3;
autonomous-system 65056;

```

Verification

IN THIS SECTION

- [Verifying the Redundant Logical Tunnel Configuration | 134](#)
- [Verifying the Layer 2 Circuit | 134](#)
- [Verifying OSPF Neighbors | 135](#)
- [Verifying the BGP Group | 135](#)

● Verifying the BGP Routes in the Routing Table | 136

Confirm that the configuration is working properly.

Verifying the Redundant Logical Tunnel Configuration

Purpose

Verify that the redundant logical tunnel with the child logical tunnel interfaces are created with the correct encapsulations.

Action

```
user@host# run show interfaces terse | match rlt0
lt-1/0/10.0      up    up    container--> rlt0.0
lt-1/0/10.1      up    up    container--> rlt0.1
lt-2/0/10.0      up    up    container--> rlt0.0
lt-2/0/10.1      up    up    container--> rlt0.1
rlt0              up    up
rlt0.0            up    up    ccc
rlt0.1            up    up    inet    10.10.10.2/24
```

Verifying the Layer 2 Circuit

Purpose

Verify that the Layer 2 circuit is up.

Action

```
user@host# run show l2circuit connections
Layer-2 Circuit Connections:

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch              Dn -- down
```

```

EM -- encapsulation mismatch      VC-Dn -- Virtual circuit Down
CM -- control-word mismatch       Up -- operational
VM -- vlan id mismatch            CF -- Call admission control failure
OL -- no outgoing label           IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC     TM -- TDM misconfiguration
BK -- Backup Connection           ST -- Standby Connection
CB -- rcvd cell-bundle size bad   SP -- Static Pseudowire
LD -- local site signaled down    RS -- remote site standby
RD -- remote site signaled down   HS -- Hot-standby Connection
XX -- unknown

```

Legend for interface status

Up -- operational

Dn -- down

Neighbor: 192.0.2.2

Interface	Type	St	Time last up	# Up trans
rlt0.0(vc 100)	rmt	Up	Aug 8 00:28:04 2013	1

Remote PE: 192.0.2.2, Negotiated control-word: No
Incoming label: 299776, Outgoing label: 299776
Negotiated PW status TLV: No
Local interface: rlt0.0, Status: Up, Encapsulation: VLAN

Verifying OSPF Neighbors

Purpose

Verify that routers are adjacent and able to exchange OSPF data.

Action

```

user@host# run show ospf neighbor

```

Address	Interface	State	ID	Pri	Dead
198.168.30.2	ge-5/2/5.0	Full	203.0.113.4	128	38
198.168.20.1	ge-5/3/8.0	Full	192.0.2.2	128	38

Verifying the BGP Group

Purpose

Verify that the BGP group is created.

Action

```

user@host# run show bgp group internal
Group Type: Internal    AS: 65056                Local AS: 65056
  Name: internal        Index: 0                  Flags: <Export Eval>
  Export: [ local-routes ]
  Holdtime: 0
  Total peers: 1        Established: 1
  203.0.113.4+179
  inet.0: 1/6/3/0
  inet.2: 0/0/0/0
  bgp.l3vpn.0: 2/2/2/0
  pe-vrf.inet.0: 2/2/2/0

```

Verifying the BGP Routes in the Routing Table

Purpose

Verify that the BGP routes are in the pe-vrf.inet.0 routing table.

Action

```

user@host# run show route protocol bgp table pe-vrf.inet.0
pe-vrf.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

198.168.50.0/24    *[BGP/170] 01:18:14, localpref 100, from 203.0.113.4
                   AS path: I, validation-state: unverified
                   > to 198.168.30.2 via ge-5/2/5.0, Push 16
198.168.51.0/24    *[BGP/170] 01:18:14, MED 2, localpref 100, from 203.0.113.4
                   AS path: I, validation-state: unverified
                   > to 198.168.30.2 via ge-5/2/5.0, Push 16

```

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
18.4R3	Starting in Junos OS Release 18.4R3, redundant logical tunnels are supported on MX Series Virtual Chassis.
14.2	Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255.
14.2	Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255.
14.2	Starting in Junos OS Release 14.2 and for 13.3R3 and 14.1R2, the valid range for device-count is from 1 to 255. The command is shown below.
13.3	Starting in Junos OS Release 13.3, you can configure redundant logical tunnels only on MX Series routers with MPCs.

Configuring Layer 2 Ethernet Services over GRE Tunnel Interfaces

IN THIS SECTION

- [Layer 2 Services over GRE Tunnel Interfaces on MX Series with MPCs | 138](#)
- [Format of GRE Frames and Processing of GRE Interfaces for Layer 2 Ethernet Packets | 138](#)
- [Guidelines for Configuring Layer 2 Ethernet Traffic Over GRE Tunnels | 139](#)
- [Sample Scenarios of Configuring Layer 2 Ethernet Traffic Over GRE Tunnels | 141](#)
- [Configuring Layer 2 Services over GRE Logical Interfaces in Bridge Domains | 142](#)
- [Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains | 143](#)
- [Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains with IPv6 Transport | 150](#)

Layer 2 Services over GRE Tunnel Interfaces on MX Series with MPCs

Starting in Junos OS Release 15.1, you can configure Layer 2 Ethernet services over GRE interfaces (*gr-fpc/pic/port* to use GRE encapsulation).

Starting in Release 19.1R1, Junos OS supports Layer 2 Ethernet services over GRE interfaces (to use GRE encapsulation) with IPv6 traffic.

The outputs of the `show bridge mac-table` and `show vpls mac-table` commands have been enhanced to display the MAC addresses learned on a GRE logical interface and the status of MAC address learning properties in the MAC address and MAC flags fields. Also, the L2 Routing Instance and L3 Routing Instance fields are added to the output of the `show interfaces gr` command to display the names of the routing instances associated with the GRE interfaces are displayed.

To enable Layer 2 Ethernet packets to be terminated on GRE tunnels, you must configure the bridge domain protocol family on the *gr-* interfaces and associate the *gr-* interfaces with the bridge domain. You must configure the GRE interfaces as core-facing interfaces, and they must be access or trunk interfaces. To configure the bridge domain family on *gr-* interfaces, include the `family bridge` statement at the `[edit interfaces gr-fpc/pic/port unit logical-unit-number]` hierarchy level. To associate the *gr-* interface with a bridge domain, include the `interface gr-fpc/pic/port` statement at the `[edit routing-instances routing-instance-name bridge-domains bridge-domain-name]` hierarchy level.

You can associate GRE interfaces in a bridge domain with the corresponding VLAN ID or list of VLAN IDs in a bridge domain by including the `vlan-id (all | none | number)` statement or the `vlan-id-list [vlan-id-numbers]` statement at the `[edit bridge-domains bridge-domain-name]` hierarchy level. The VLAN IDs configured for the bridge domain must match with the VLAN IDs that you configure for GRE interfaces by using the `vlan-id (all | none | number)` statement or the `vlan-id-list [vlan-id-numbers]` statement at the `[edit interfaces gr-fpc/pic/port unit logical-unit-number]` hierarchy level. You can also configure GRE interfaces within a bridge domain associated with a virtual switch instance. Layer 2 Ethernet packets over GRE tunnels are also supported with the GRE key option. The `gre-key` match condition allows a user to match against the GRE key field, which is an optional field in GRE encapsulated packets. The key can be matched as a single key value, a range of key values, or both.

Format of GRE Frames and Processing of GRE Interfaces for Layer 2 Ethernet Packets

The GRE frame contains the outer MAC header, outer IP header, GRE header, original layer 2 frame, and frame checksum (FCS).

In the outer MAC header, the following fields are present:

- The outer destination MAC address is set as the next-hop MAC address

- The outer source MAC address is set as the source address of the MX Series router that functions as the gateway
- The outer VLAN tag information

In the outer IP header, the following fields are contained:

- The outer source address is set as the source address of the MX Series router gateway
- The outer destination address is set as the remote GRE tunnel address
- The outer protocol type is set as 47 (encapsulation type is GRE)
- The VLAN ID configuration within the bridge domain updates the VLAN ID of the original Layer 2 header

The gr-interface supports GRE encapsulation over IPv4 and IPv6, which is supported over Layer 3 over GRE. Support for bridging over GRE enables you to configure bridge domain families on gr- interfaces and also enable integrated routing and bridging (IRB) on gr- interfaces. The device control daemon (dcd) that controls the physical and logical interface processes enables the processing of bridge domain families under the GRE interfaces. The kernel supports IRB to send and receive packets on IRB interfaces.

The Packet Forwarding Engine supports the Layer 2 encapsulation and decapsulation over GRE interfaces. The chassis daemon is responsible for creating the GRE physical interface when an FPC comes online and triggering the deletion of the GRE interfaces when the FPC goes offline. The kernel receives the GRE logical interface that is added over the underlying physical interface and propagates the GRE logical interface to other clients, including the Packet Forwarding Engine to create the Layer 2 over GRE data path in the hardware. In addition, it adds the GRE logical interface into a bridge domain. The Packet Forwarding Engine receives interprocess communication message (IPC) from the kernel and adds the interface into the forwarding plane. The existing MTU size for the GRE interface is increased by 22 bytes for the L2 header addition (6 DMAC + 6 SMAC + 4 CVLAN + 4 SVLAN + 2 EtherType)

Guidelines for Configuring Layer 2 Ethernet Traffic Over GRE Tunnels

Observe the following guidelines while configuring Layer 2 packets to be transmitted over GRE tunnel interfaces on MX Series routers with MPCs:

- For integrated routing and bridging (IRB) to work, at least one Layer 2 interface must be up and active, and it must be associated with the bridge domain as an IRB interface along with a GRE Layer 2 logical interface. This configuration is required to leverage the existing broadcast infrastructure of Layer 2 with IRB.
- Graceful Routing Engine switchover (GRES) is supported and unified ISSU is not currently supported.

- MAC addresses learned from the GRE networks are learned on the bridge domain interfaces associated with the `gr-fpc/pic/port.unit` logical interface. The MAC addresses are learned on GRE logical interfaces and the Layer 2 token used for forwarding is the token associated with the GRE interface. Destination MAC lookup yields an L2 token, which causes the next-hop lookup. This next-hop is used to forward the packet.
- The GRE tunnel encapsulation and decapsulation next-hops are enhanced to support this functionality. The GRE tunnel encapsulation next-hop is used to encapsulate the outer IP and GRE headers with the incoming L2 packet. The GRE tunnel decapsulation next-hop is used to decapsulate the outer IP and GRE headers, parse the inner Layer 2 packet, and set the protocol as bridge for further bridge domain properties processing in the Packet Forwarding Engine.
- The following packet flows are supported:
 - As part of Layer 2 packet flows, L2 unicast from L2 to GRE, L2 unicast from GRE to L2, Layer 2 broadcast, unknown unicast, and multicast (L2 BUM) from L2 to GRE, and L2 BUM from GRE to L2 are supported.
 - As part of Layer 3 packet flows, L3 Unicast from L2 to GRE, L3 Unicast from GRE to L2, L3 Multicast from L2 to GRE, L3 Multicast from GRE to L2, and L3 Multicast from Internet to GRE and L2 are supported.
- Support for L2 control protocols is not available.
- At the GRE decapsulation side, packets destined to the tunnel IP are processed and decapsulated by the forwarding plane, and inner L2 packets are processed. MAC learned packets are generated for control plane processing for newly learned MAC entries. However, these entries are throttled for MAC learning.
- 802.1x authentication can be used to validate the individual endpoints and protect them from unauthorized access.
- With the capability to configure bridge domain families on GRE tunnel interfaces, the maximum number of GRE interfaces supported depends on the maximum number of tunnel devices allocated, where each tunnel device can host upto 4000 logical interfaces. The maximum number of logical tunnel interfaces supported is not changed with the support for Layer 2 GRE tunnels. For example, in a 4x10 MIC on MX960 routers, 8000 tunnel logical interfaces can be created.
- The tunnels are pinned to a specific Packet Forwarding Engine instance.
- Statistical information for GRE Layer 2 tunnels is displayed in the output of the `show interfaces gr-fpc/pic/port` command.
- Only trunk and access mode configuration is supported for the bridge family of GRE interfaces; subinterface style configuration is not supported.

- You can enable a connection to a traditional Layer 2 network. Connection to a VPLS network is not currently supported. IRB in bridge domains with GRE interfaces is supported.
- Virtual switch instances are supported.
- Configuration of the GRE Key and using it to perform the hash load-balancing at the GRE tunnel-initiated and transit routers is supported.

Sample Scenarios of Configuring Layer 2 Ethernet Traffic Over GRE Tunnels

You can configure Layer 2 Ethernet services over GRE interfaces (*gr-fpc/pic/port* to use GRE encapsulation). This topic contains the following sections that illustrate sample network deployments that support Layer 2 packets over GRE tunnel interfaces:

GRE Tunnels with an MX Series Router as the Gateway in Layer 3

You can configure an MX Series router as the gateway that contains GRE tunnels configured to connect to legacy switches on the one end and to a Layer 3 network on the other end. The Layer 3 network in turn can be linked with multiple servers on a LAN where the GRE tunnel is terminated from the WAN.

GRE Tunnels With an MX Series Router as the Gateway and Aggregator

You can configure an MX Series router as the gateway with GRE tunnels configured and also with aggregation specified. The gateway can be connected to legacy switches on one end of the network and the aggregator can be connected to a top-of-rack (ToR) switch, as a QFX Series device, which handles GRE tunneled packets with load balancing. The ToR switch can be connected, in turn, over a Layer 3 GRE tunnel to several servers in data centers.

GRE Tunnels with MX Series Gateways for Enterprise and Data Center Servers

You can configure an MX Series router as the gateway with GRE tunnels configured. Over the Internet, GRE tunnels connect multiple gateways, which are MX routers, to servers in enterprises where the GRE tunnel is terminated from the WAN on one end, and to servers in data centers on the other end.

The following configuration scenarios are supported for Layer 2 Ethernet over GRE tunnels:

- In a Layer 2 Ethernet over GRE with VPLS environment, an MX Series router supports Layer 2 over GRE tunnels (without the MPLS layer) and terminate these tunnels into a VPLS or an routed VLAN interface (RVI) into a L3VPN. The tunnels serve to cross the cable modem termination system (CMTS) and cable modem CM infrastructure transparently, up to the MX Series router that serves as the gateway. Every GRE tunnel terminates over a VLAN interface, a VPLS instance, or an IRB interface.

- In a Layer 2 Ethernet over GRE without VPLS environment, Layer 2 VPN encapsulations are needed for conditions that do not involve these protocols. Certain data center users terminate the other end of GRE tunnels directly on the servers on the LAN, while an MX Series router functions as the gateway router between the WAN and LAN. This type of termination of tunnels enables users to build overlay networks within the data center without having to configure end-user VLANs, IP addresses, and other network parameters on the underlying switches. Such a setup simplifies data center network design and provisioning.



NOTE: Layer 2 over GRE is not supported in ACX2200 router.

Configuring Layer 2 Services over GRE Logical Interfaces in Bridge Domains

You can configure Layer 2 Ethernet services over GRE interfaces (*gr-fpc/pic/port* to use GRE encapsulation).

To configure a GRE tunnel interface, associate it in a bridge domain within a virtual-switch instance, and specify the amount of bandwidth reserved for tunnel services traffic:

1. Configure GRE tunnel interface and specify the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine.

```
[edit]
user@host# set chassis fpc slot-number pic slot-number tunnel-services bandwidth (1g | 10g | 20g | 40g)
```

2. Configure the interfaces and their VLAN IDs.

```
[edit]
user@host# set interfaces gr-interface-name unit logical-unit-number family family-name address address
user@host# set interfaces gr-interface-name unit logical-unit-number family family-name interface-mode trunk
user@host# set interfaces gr-interface-name unit logical-unit-number family family-name vlan-id-list vlan-id-list
user@host# set interfaces gr-interface-name unit logical-unit-number tunnel source source-address
```

```
user@host# set interfaces gr-interface-name unit logical-unit-number tunnel destination
destination-address
```

3. Create a virtual switch instance with a bridge domain and associate the GRE logical interfaces.

```
[edit routing-instances]
user@host# set routing-instance-name instance-type virtual-switch
user@host# set routing-instance-name interface interface-name unit logical-unit-number
user@host# set routing-instance-name bridge-domains bridge-domain-name vlan-id vlan-id
```

The VLAN IDs configured for the bridge domain must match with the VLAN IDs that you configure for GRE interfaces by using the `vlan-id` (`all` | `none` | `number`) statement or the `vlan-id-list` [*vlan-id-numbers*] statement at the `[edit interfaces gr-fpc/pic/port unit logical-unit-number]` hierarchy level.

Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains

IN THIS SECTION

- [Requirements | 144](#)
- [Overview | 144](#)
- [Configuration | 144](#)
- [Verification | 148](#)

This example illustrates how you can configure GRE logical interfaces in a bridge domain. You can also configure a virtual switch instance associated with a bridge domain and include GRE interfaces in the bridge domain. This type of configuration enables you to specify Layer 2 Ethernet packets to be terminated on GRE tunnels. In a Layer 2 Ethernet over GRE with VPLS environment, an MX Series router supports Layer 2 over GRE tunnels (without the MPLS layer) and terminate these tunnels into a VPLS or a routed VLAN interface (RVI) into a L3VPN. The tunnels serve to cross the cable modem termination system (CMTS) and cable modem CM infrastructure transparently, up to the MX Series router that serves as the gateway. Every GRE tunnel terminates over a VLAN interface, a VPLS instance, or an IRB interface.

Requirements

This example uses the following hardware and software components:

- An MX Series router
- Junos OS Release 15.1R1 or later running on an MX Series router with MPCs.

Overview

GRE encapsulates packets into IP packets and redirects them to an intermediate host, where they are de-encapsulated and routed to their final destination. Because the route to the intermediate host appears to the inner datagrams as one hop, Juniper Networks EX Series Ethernet switches can operate as if they have a virtual point-to-point connection with each other. GRE tunnels allow routing protocols like RIP and OSPF to forward data packets from one switch to another switch across the Internet. In addition, GRE tunnels can encapsulate multicast data streams for transmission over the Internet.

Ethernet frames have all the essentials for networking, such as globally unique source and destination addresses, error control, and so on. •Ethernet frames can carry any kind of packet. Networking at Layer 2 is protocol independent (independent of the Layer 3 protocol). If more of the end-to-end transfer of information from a source to a destination can be done in the form of Ethernet frames, more of the benefits of Ethernet can be realized on the network. Networking at Layer 2 can be a powerful adjunct to IP networking, but it is not usually a substitute for IP networking.

Consider a sample network topology in which a GRE tunnel interface is configured with the bandwidth set as 10 gigabits per second for tunnel traffic on each Packet Forwarding Engine. The GRE interface, gr-0/1/10.0, is specified with the source address of 192.0.2.2 and the destination address of 192.0.2.1. Two gigabit Ethernet interfaces, ge-0/1/2.0 and ge-0/1/6.0, are also configured. A virtual switch instance, VS1, is defined and a bridge domain, bd0, is associated with VS1. The bridge domain contains the VLAN ID of 10. The GRE interface is configured as a trunk interface and associated with the bridge domain, bd0. With such a setup, Layer 2 Ethernet services can be terminated over GRE tunnel interfaces in virtual switch instances that contain bridge domains.

Configuration

IN THIS SECTION

- [Procedure | 145](#)

To configure a GRE tunnel interface, associate it in a bridge domain within a virtual-switch instance, and specify the amount of bandwidth reserved for tunnel services traffic.

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them in a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level:

```
set chassis fpc 0 pic 1 tunnel-services bandwidth 1g
set chassis network-services enhanced-ip
set interfaces ge-0/1/2 unit 0 family inet address 192.0.2.2/30
set interfaces ge-0/1/6 unit 0 family bridge interface-mode trunk
set interfaces ge-0/1/6 unit 0 family bridge vlan-id-list 1-100
set interfaces gr-0/1/10 unit 0 tunnel source 192.0.2.2
set interfaces gr-0/1/10 unit 0 tunnel destination 192.0.2.1
set interfaces gr-0/1/10 unit 0 family bridge interface-mode trunk
set interfaces gr-0/1/10 unit 0 family bridge vlan-id-list 1-100
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 bridge-domains bd0 vlan-id 10
set routing-instances VS1 interface ge-0/1/6.0
set routing-instances VS1 interface gr-0/1/10.0
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the *CLI User Guide*.

To configure GRE logical tunnel interfaces for Layer 2 services in bridge domains:

1. Configure GRE tunnel interface and specify the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine.

```
[edit]
user@host# set chassis fpc 0 pic 1 tunnel-services bandwidth 1g
user@host# set chassis network-services enhanced-ip
```

2. Configure the interfaces and their VLAN IDs.

```
[edit]
user@host# set interfaces ge-0/1/2 unit 0 family inet address 192.0.2.2/30
user@host# set interfaces ge-0/1/6 unit 0 family bridge interface-mode trunk
user@host# set interfaces ge-0/1/6 unit 0 family bridge vlan-id-list 1-100
user@host# set interfaces gr-0/1/10 unit 0 tunnel source 192.0.2.2
user@host# set interfaces gr-0/1/10 unit 0 tunnel destination 192.0.2.1
user@host# set interfaces gr-0/1/10 unit 0 family bridge interface-mode trunk
user@host# set interfaces gr-0/1/10 unit 0 family bridge vlan-id-list 1-100
```

3. Configure the bridge domain in a virtual switch instance and associate the GRE interface with it.

```
[edit]
user@host# set routing-instances VS1 instance-type virtual-switch
user@host# set routing-instances VS1 bridge-domains bd0 vlan-id 10
user@host# set routing-instances VS1 interface ge-0/1/6.0
user@host# set routing-instances VS1 interface gr-0/1/10.0
```

Results

Display the results of the configuration:

```
user@host> show configuration

chassis {
  fpc 0 {
    pic 1 {
      tunnel-services {
        bandwidth 1g;
      }
    }
  }
  network-services enhanced-ip;
}

interfaces {
  ge-0/1/2 {
    unit 0 {
      family inet {
        address 192.0.2.2/30;
      }
    }
  }
  ge-0/1/6 {
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 1-100;
      }
    }
  }
  gr-0/1/10 {
    unit 0 {
      tunnel {
        source 192.0.2.2;
        destination 192.0.2.1;
      }
      family bridge {
        interface-mode trunk;
        vlan-id-list 1-100;
      }
    }
  }
}
```

```

    }
  }
}
ge-0/1/6 {
  unit 0 {
    family bridge {
      interface-mode trunk;
      vlan-id-list 1-100;
    }
  }
}
gr-0/1/10 {
  unit 0 {
    tunnel {
      source 192.0.2.2;
      destination 192.0.2.1;
    }
    family bridge {
      interface-mode trunk;
      vlan-id-list 1-100;
    }
  }
}
}

VS-1 {
  instance-type virtual-switch;
  interface ge-0/1/6.0;
  interface gr-0/1/10.0;
  bridge-domains {
    bd0 {
      vlan-id 10;
    }
  }
}
}

```

Verification

IN THIS SECTION

- [Verifying the MAC Addresses Learned on GRE Interfaces | 148](#)
- [Verifying the MAC Address Learning Status | 149](#)

To confirm that the configuration is working properly, perform these tasks:

Verifying the MAC Addresses Learned on GRE Interfaces

Purpose

Display the MAC addresses learned on a GRE logical interface.

Action

From operational mode, use the `show bridge mac-table` command

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)
```

```
Routing instance : default-switch
Bridging domain : vlan-1, VLAN : 1
  MAC          MAC      Logical
  address      flags    interface
  00:00:5e:00:53:f7 D,SE   gr-1/2/10.0
  00:00:5e:00:53:32 D,SE   gr-1/2/10.0
  00:00:5e:00:53:21 DL      ge-1/0/0.0
  00:00:5e:00:53:11 DL      ge-1/1/0.0
```

```
Routing instance : default-switch
Bridging domain : vlan-2, VLAN : 2
  MAC          MAC      Logical
  address      flags    interface
  00:00:5e:00:53:33 D,SE   gr-1/2/10.1
```

00:00:5e:00:53:10	DL	ge-1/0/0.1
00:00:5e:00:53:23	DL	ge-1/1/0.1

Meaning

The output displays MAC addresses learned on GRE logical tunnels.

Verifying the MAC Address Learning Status

Purpose

Display the status of MAC address learning properties in the MAC address and MAC flags fields.

Action

From operational mode, enter the `show vpls mac-table` command.

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : vpls_4site:1000
Bridging domain : __vpls_4site:1000__,   MAC      Logical
address      flags      interface
00:00:5e:00:53:f4  D,SE    ge-4/2/0.1000
00:00:5e:00:53:02  D,SE    lsi.1052004
00:00:5e:00:53:03  D,SE    lsi.1048840
00:00:5e:00:53:04  D,SE    lsi.1052005
00:00:5e:00:53:33  D,SE    gr-1/2/10.10

user@host> show interfaces gr-2/2/10
Physical interface: gr-2/2/10, Enabled, Physical link is Up
  Interface index: 214, SNMP ifIndex: 690
  Type: GRE, Link-level type: GRE, MTU: Unlimited, Speed: 1000mbps
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)

Logical interface gr-2/2/10.0 (Index 342) (SNMP ifIndex 10834)
  Flags: Up Point-To-Point SNMP-Traps 0x4000 IP-Header
```

```

198.51.100.1:198.51.100.254:47:df:64:0000000000000000 Encapsulation: GRE-NULL
  L2 Routing Instance: vs1, L3 Routing Instance: default
  Copy-tos-to-outer-ip-header: Off
  Gre keepalives configured: Off, Gre keepalives adjacency state: down
  Input packets : 2
  Output packets: 0
  Protocol bridge, MTU: 1476
    Flags: Sendbcast-pkt-to-re
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 6/8, Local: 6.0.0.1, Broadcast: 6.255.255.255

user@host> show interfaces gr-2/2/10.0
  Logical interface gr-2/2/10.0 (Index 342) (SNMP ifIndex 10834)
    Flags: Up Point-To-Point SNMP-Traps 0x4000 IP-Header
198.51.100.1:198.51.100.254:47:df:64:0000000000000000 Encapsulation: GRE-NULL
  L2 Routing Instance: vs1, L3 Routing Instance: default
  Copy-tos-to-outer-ip-header: Off
  Gre keepalives configured: Off, Gre keepalives adjacency state: down
  Input packets : 2
  Output packets: 0
  Protocol bridge, MTU: 1476
    Flags: Sendbcast-pkt-to-re
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 6/8, Local: 6.0.0.1, Broadcast: 6.255.255.255

```

Meaning

The output displays the status of MAC address learning properties in the MAC address and MAC flags fields. The output displays the names of the routing instances associated with the GRE interfaces are displayed.

Example: Configuring Layer 2 Services Over GRE Logical Interfaces in Bridge Domains with IPv6 Transport

IN THIS SECTION

● [Requirements](#) | 151

- Overview | 151
- Configuration | 152
- Verification | 158

This example illustrates how you can configure GRE logical interfaces in a bridge domain. You can also configure a virtual switch instance associated with a bridge domain and include GRE interfaces in the bridge domain. This type of configuration enables you to specify Layer 2 Ethernet packets to be terminated on GRE tunnels. In a Layer 2 Ethernet over GRE with VPLS environment, an MX Series router supports Layer 2 over GRE tunnels (without the MPLS layer) and terminate these tunnels into a VPLS or an routed VLAN interface (RVI) into a L3VPN. The tunnels serve to cross the cable modem termination system (CMTS) and cable modem CM infrastructure transparently, up to the MX Series router that serves as the gateway. Every GRE tunnel terminates over a VLAN interface, a VPLS instance, or an IRB interface.

Requirements

This example uses the following hardware and software components:

- Two MX Series routers
- Junos OS Release 19.R1 or later running on MX Series routers with MPCs.

Overview

GRE encapsulated IPv6 packets are redirected to an intermediate host where GRE header is decapsulated and routed to the IPv6 destination.

Consider a sample network topology with two devices. On device 1, GRE tunnel interface is configured with the bandwidth set as 1 gigabits per second for tunnel traffic on each Packet Forwarding Engine. The GRE interface, gr-0/0/10.0, is specified with the source address of 2001:DB8::2:1 and the destination address of 2001:DB8::3:1. Two interfaces, ae0 and xe-0/0/19, are also configured. A virtual switch instance, VS1, is defined and a bridge domain, bd1, is associated with VS1. The bridge domain contains the VLAN ID of 20. The GRE interface is configured as a trunk interface and associated with the bridge domain, bd1. With such a setup, Layer 2 Ethernet services can be terminated over GRE tunnel interfaces in virtual switch instances that contain bridge domains.

On device 2, GRE tunnel interface is configured with the bandwidth set as 1 gigabits per second for tunnel traffic on each Packet Forwarding Engine. The GRE interface, gr-0/0/10.0, is specified with the source address of 2001:DB8::21:1 and the destination address of 2001:DB8::31:1. Two interfaces, ae0 and xe-0/0/1, are also configured. A virtual switch instance, VS1, is defined and a bridge domain, bd1, is

associated with VS1. The bridge domain contains the VLAN ID of 20. The GRE interface is configured as an access interface and associated with the bridge domain, bd1.

Configuration

IN THIS SECTION

- [Procedure](#) | 152

To configure a GRE tunnel interface, associate it in a bridge domain within a virtual-switch instance, and specify the amount of bandwidth reserved for tunnel services traffic.

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them in a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level:

For Device 1:

```
set chassis aggregated-devices ethernet device-count 2
set chassis fpc 0 pic 0 tunnel-services bandwidth 1g
set chassis network-services enhanced-ip
set interfaces ae0 unit 0 family inet6 address 2001:DB8::1:1/32;
set interfaces xe-0/0/19 unit 0 family bridge interface-mode trunk
set interfaces xe-0/0/19 unit 0 family bridge vlan-id-list 20-21
set interfaces xe-1/0/2 gigether-options 802.3ad ae0
set interfaces xe-1/0/3 gigether-options 802.3ad ae0
set interfaces gr-0/0/10.0 unit 0 tunnel source 2001:DB8::2:1
set interfaces gr-0/0/10.0 unit 0 tunnel destination 2001:DB8::3:1/32
set interfaces gr-0/0/10.0 unit 0 family bridge interface-mode trunk
set interfaces gr-0/0/10.0 unit 0 family bridge vlan-id-list 20-30
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 bridge-domains bd1 vlan-id 20
set routing-instances VS1 interface xe-0/0/19.0
set routing-instances VS1 interface gr-0/0/10.0
```


For device 2:

```
set chassis aggregated-devices ethernet device-count 2
set chassis fpc 0 pic 0 tunnel-services bandwidth 1g
set chassis network-services enhanced-ip
set interfaces ae0 unit 0 family inet6 address 2001:DB8::11:1/32;
set interfaces xe-0/0/1 unit 0 family bridge interface-mode trunk
set interfaces xe-0/0/1 unit 0 family bridge vlan-id-list 20-21
set interfaces xe-1/0/2 gigether-options 802.3ad ae0
set interfaces xe-1/0/3 gigether-options 802.3ad ae0
set interfaces gr-0/0/10.0 unit 0 tunnel source 2001:DB8::21:1
set interfaces gr-0/0/10.0 unit 0 tunnel destination 2001:DB8::31:1/32
set interfaces gr-0/0/10.0 unit 0 family bridge interface-mode access
set interfaces gr-0/0/10.0 unit 0 family bridge vlan-id-list 20-30
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 bridge-domains bd1 vlan-id 20
set routing-instances VS1 interface xe-0/0/1.0
set routing-instances VS1 interface gr-0/0/10.0
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the *CLI User Guide*.

To configure GRE logical tunnel interfaces over IPv6 for Layer 2 services in bridge domains for Device1 and Device2:

1. Configure GRE tunnel interface and specify the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine of Device1.

```
[edit]
user@host# set chassis aggregated-devices ethernet device-count 2
user@host# set chassis fpc 0 pic 0 tunnel-services bandwidth 1g
user@host# set chassis network-services enhanced-ip
```

2. Configure the interfaces and their VLAN IDs.

```
[edit]
user@host# set interfaces ae0 unit 0 family inet6 address 2001:DB8::1:1/32;
```

```

user@host# set interfaces xe-0/0/19.0 unit 0 family bridge interface-mode trunk
user@host# set interfaces xe-0/0/19.0 unit 0 family bridge vlan-id-list 20-21
user@host# set interfaces xe-1/0/2 gigether-options 802.3ad ae0
user@host# set interfaces xe-1/0/3 gigether-options 802.3ad ae0
user@host# set interfaces gr-0/0/10.0 unit 0 tunnel source 2001:DB8::2:1
user@host# set interfaces gr-0/0/10.0 unit 0 tunnel destination 2001:DB8::3:1
user@host# set interfaces gr-0/0/10.0 unit 0 family bridge interface-mode trunk
user@host# set interfaces gr-0/0/10.0 unit 0 family bridge vlan-id-list 20-30

```

3. Configure the bridge domain in a virtual switch instance and associate the GRE interface with it.

```

[edit]
user@host# set routing-instances VS1 instance-type virtual-switch
user@host# set routing-instances VS1 bridge-domains bd1 vlan-id 20
user@host# set routing-instances VS1 interface xe-0/0/19.0
user@host# set routing-instances VS1 interface gr-0/0/10.0

```

4. Configure GRE tunnel interface and specify the amount of bandwidth to reserve for tunnel traffic on each Packet Forwarding Engine of Device2.

```

[edit]
user@host# set chassis aggregated-devices ethernet device-count 2
user@host# set chassis fpc 0 pic 0 tunnel-services bandwidth 1g
user@host# set chassis network-services enhanced-ip

```

5. Configure the interfaces and their VLAN IDs.

```

[edit]
user@host# set interfaces ae0 unit 0 family inet6 address 2001:DB8::11:1/32;
user@host# set interfaces xe-0/0/1 unit 0 family bridge interface-mode trunk
user@host# set interfaces xe-0/0/1 unit 0 family bridge vlan-id-list 20-21
user@host# set interfaces xe-1/0/2 gigether-options 802.3ad ae0
user@host# set interfaces xe-1/0/3 gigether-options 802.3ad ae0
user@host# set interfaces gr-0/0/10.0 unit 0 tunnel source 2001:DB8::21:1
user@host# set interfaces gr-0/0/10.0 unit 0 tunnel destination 2001:DB8::31:1/32
user@host# set interfaces gr-0/0/10.0 unit 0 family bridge interface-mode trunk
user@host# set interfaces gr-0/0/10.0 unit 0 family bridge vlan-id-list 20-30

```

6. Configure the bridge domain in a virtual switch instance and associate the GRE interface with it.

```
[edit]
user@host# set routing-instances VS1 instance-type virtual-switch
user@host# set routing-instances VS1 bridge-domains bd1 vlan-id 20
user@host# set routing-instances VS1 bridge-domains routing-interface irb.0
user@host# set routing-instances VS1 interface xe-0/0/1.0
user@host# set routing-instances VS1 interface gr-0/0/10.0
```

Results

Display the results of the configuration on Device 1:

```
user@host> show configuration
chassis {
  fpc 0 {
    pic 0 {
      tunnel-services {
        bandwidth 1g;
      }
    }
  }
  network-services enhanced-ip;
}
interfaces {
  ae0 {
    unit 0 {
      family inet6 {
        address 2001:DB8::1:1/32;
      }
    }
  }
  xe-0/0/19 {
    unit 0 {
      family bridge {
        interface-mode trunk;
        vlan-id-list 20-21;
      }
    }
  }
  gr-0/0/10 {
```

```

        unit 0 {
            tunnel {
                source 2001:DB8::2:1;
                destination 2001:DB8::3:1;
            }
            family bridge {
                interface-mode trunk;
                vlan-id-list 20-30;
            }
        }
    }
}

VS-1 {
    instance-type virtual-switch;
    interface xe-0/0/19.0;
    interface gr-0/0/10.0;
    bridge-domains {
        bd1 {
            vlan-id 20;
        }
    }
}

```

Display the results of the configuration on Device 2:

```

user@host> show configuration
chassis {
    fpc 0 {
        pic 0 {
            tunnel-services {
                bandwidth 1g;
            }
        }
    }
    network-services enhanced-ip;
}

interfaces {
    ae0 {
        unit 0 {
            family inet6 {
                address 2001:DB8::11:1/32;
            }
        }
    }
}

```

```

    }
  }
}
xe-0/0/1 {
  unit 0 {
    family bridge {
      interface-mode trunk;
      vlan-id-list 20-21;
    }
  }
}
gr-0/0/10 {
  unit 0 {
    tunnel {
      source 2001:DB8::21:1;
      destination 2001:DB8::31:1;
    }
    family bridge {
      interface-mode access;
      vlan-id-list 20-30;
    }
  }
}
}

VS-1 {
  instance-type virtual-switch;
  interface xe-0/0/1.0;
  interface gr-0/0/10.0;
  bridge-domains {
    bd1 {
      vlan-id 20;
    }
  }
}
}

```

Verification

IN THIS SECTION

- [Verifying the MAC Addresses Learned on GRE Interfaces | 158](#)

To confirm that the configuration is working properly, perform these tasks:

Verifying the MAC Addresses Learned on GRE Interfaces

Purpose

Display the MAC addresses learned on a GRE logical interface.

Action

From operational mode, use the show bridge mac-table command

```
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : VS1
Bridging domain : bd1, VLAN : 20
  MAC      MAC   Logical      NH    RTR
address    flags  interface   Index ID
00:00:00:11:11:11 D    gr-0/0/10.0
00:00:00:11:11:12 D    gr-0/0/10.0
00:00:00:11:11:13 D    gr-0/0/10.0
00:00:00:11:11:14 D    gr-0/0/10.0
00:00:00:11:11:15 D    gr-0/0/10.0
00:00:00:11:11:16 D    gr-0/0/10.0
00:00:00:11:11:17 D    gr-0/0/10.0
00:00:00:11:11:18 D    gr-0/0/10.0
00:00:00:11:11:19 D    gr-0/0/10.0
00:00:00:11:11:1a D    gr-0/0/10.0
00:00:00:22:22:22 D    xe-0/0/19.0
00:00:00:22:22:23 D    xe-0/0/19.0
00:00:00:22:22:24 D    xe-0/0/19.0
```

00:00:00:22:22:25	D	xe-0/0/19.0
00:00:00:22:22:26	D	xe-0/0/19.0
00:00:00:22:22:27	D	xe-0/0/19.0
00:00:00:22:22:28	D	xe-0/0/19.0
00:00:00:22:22:29	D	xe-0/0/19.0
00:00:00:22:22:2a	D	xe-0/0/19.0
00:00:00:22:22:2b	D	xe-0/0/19.0

Meaning

The output displays MAC addresses learned on GRE logical tunnels.

Configuring 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 OS 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.



NOTE: The **pe** and **pd** interfaces do not support class-of-service (CoS) configurations.

RELATED DOCUMENTATION

[Tunnel Services Overview](#) | 2

Facilitating VRF Table Lookup Using Virtual Loopback Tunnel Interfaces

IN THIS SECTION

- [Configuring Virtual Loopback Tunnels for VRF Table Lookup | 160](#)
- [Configuring Tunnel Interfaces for Routing Table Lookup | 162](#)
- [Example: Configuring a Virtual Loopback Tunnel for VRF Table Lookup | 163](#)
- [Example: Virtual Routing and Forwarding \(VRF\) and Service Configuration | 164](#)

Configuring Virtual Loopback Tunnels for VRF Table Lookup

To enable egress filtering, you can either configure filtering based on the IP header, or you can configure a virtual loopback tunnel on routers equipped with a Tunnel PIC. [Table 10 on page 160](#) describes each method.

Table 10: Methods for Configuring Egress Filtering

Method	Interface Type	Configuration Guidelines	Comments
Filter traffic based on the IP header	Nonchannelized Point-to-Point Protocol / High Level Data Link Control (PPP/ HDLC) core-facing SONET/SDH interfaces	<p>Include the <code>vrf-table-label</code> statement at the <code>[edit routing-instances <i>instance-name</i>]</code> hierarchy level.</p> <p>For more information, see the Junos OS VPNs Library for Routing Devices.</p>	There is no restriction on customer-edge (CE) router-to-provider edge (PE) router interfaces.

Table 10: Methods for Configuring Egress Filtering (*Continued*)

Method	Interface Type	Configuration Guidelines	Comments
Configure a virtual loopback tunnel on routers equipped with a Tunnel PIC	All interfaces	See the guidelines in this section.	<p>Router must be equipped with a Tunnel PIC.</p> <p>There is no restriction on the type of core-facing interface used or CE router-to-PE router interface used.</p> <p>You cannot configure a virtual loopback tunnel and the <code>vrf-table-label</code> statement at the same time.</p>

You can configure a virtual loopback tunnel to facilitate 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 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 for route lookup. To specify a virtual loopback tunnel interface name, you configure the virtual loopback tunnel interface at the `[edit interfaces]` hierarchy level and include the `family inet` and `family mpls` statements:

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

```

        family inet;
    }
}

```

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

```
interface vt-fpc/pic/port;
```



NOTE: On virtual loopback tunnel interfaces, none of the logical interface statements except the `family` statement is supported. Note that you can configure only `inet` and `mpls` families, and you cannot configure IPv4 or IPv6 addresses on virtual loopback tunnel interfaces. Also, virtual loopback tunnel interfaces do not support class-of-service (CoS) configurations.

SEE ALSO

| [Tunnel Services Overview](#) | 2

Configuring Tunnel Interfaces 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 OS 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 include this statement at the following hierarchy levels:

- [edit interfaces *gr-fpc/pic/port* unit *logical-unit-number* tunnel]
- [edit logical-systems *logical-system-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 ["Configuring Virtual Loopback Tunnels for VRF Table Lookup"](#) on page 160.

For more information about VPNs, see the [Junos OS VPNs Library for Routing Devices](#).

SEE ALSO

[Tunnel Services Overview | 2](#)

destination (Routing Instance)

Example: Configuring a Virtual Loopback Tunnel for VRF Table Lookup

Configure a virtual loopback tunnel 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;
    }
}

```

SEE ALSO

| [Tunnel Services Overview](#) | 2

Example: Virtual Routing and Forwarding (VRF) and Service Configuration

The following example combines virtual routing and forwarding (*VRF*) and services configuration:

```

[edit policy-options]
policy-statement test-policy {
    term t1 {
        then reject;
    }
}
[edit routing-instances]
test {
    interface ge-0/2/0.0;
    interface sp-1/3/0.20;
}

```

```

instance-type vrf;
route-distinguisher 10.58.255.1:37;
vrf-import test-policy;
vrf-export test-policy;
routing-options {
    static {
        route 0.0.0.0/0 next-table inet.0;
    }
}
[edit interfaces]
ge-0/2/0 {
    unit 0 {
        family inet {
            service {
                input service-set nat-me;
                output service-set nat-me;
            }
        }
    }
}
sp-1/3/0 {
    unit 0 {
        family inet;
    }
    unit 20 {
        family inet;
        service-domain inside;
    }
    unit 21 {
        family inet;
        service-domain outside;
    }
}
[edit services]
stateful-firewall {
    rule allow-any-input {
        match-direction input;
        term t1 {
            then accept;
        }
    }
}
nat {

```

```

    pool hide-pool {
        address 10.58.16.100;
        port automatic;
    }
    rule hide-all-input {
        match-direction input;
        term t1 {
            then {
                translated {
                    source-pool hide-pool;
                    translation-type source napt-44;
                }
            }
        }
    }
}

service-set nat-me {
    stateful-firewall-rules allow-any-input;
    nat-rules hide-all-input;
    interface-service {
        service-interface sp-1/3/0.20;
    }
}
}

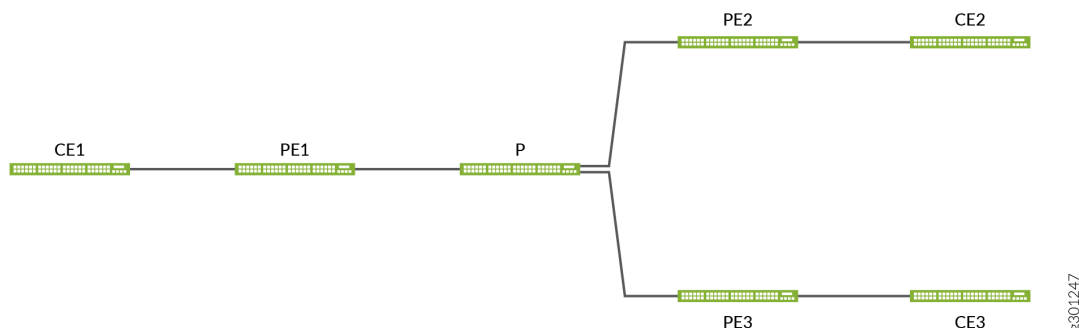
```

BGP Layer 3 VPN over IP-IP Tunnels Overview

The traditional VPN services use the label-based forwarding technique of MPLS. Some networks might transition from MPLS network to IP fabric core network. VPN label is not supported in IP fabric core network.

We introduce support for BGP Layer 3 VPN over IP over IP (IP-IP) tunnels to create a new transport service that does not require a VPN label to identify the VRF at Egress PE. This offer carrying different types of traffic using the same infrastructure and same BGP topology with RD and route targets over the IP-IP network. IP-IP tunnels terminate into service-layer VRF, so you do not need to use a service label. This feature allows interoperability between the new VRF and traditional VRF, so both types of overlays can coexist in your network. You can use this feature to transition from an MPLS network to an IP fabric core network and to protect your network from distributed denial-of-service (DDoS) attacks.

Figure 6: Sample Network That Shows Coexistence of New and Traditional VRFs



In [Figure 6 on page 167](#), PE1 is an Egress PE that supports both traditional and new types of tunnels, so it will advertise both these types of tunnels in its L3VPN routes. PE2 is an Ingress PE that uses new type of tunnel to reach CE1 and PE3 is an Ingress PE that uses traditional tunnel to reach CE1. If an IP-IP tunnel is selected to forward the traffic, the application service label is ignored and a firewall filter is used to demultiplex the traffic. L3VPN traffic is decapsulated and then do route lookup in the VRF while IPv4/IPv6 traffic is decapsulated and do route lookup in the inet.0/inet6.0 table.

Threat Mitigation system prefixes are exchanged via BGP inetvpn or inet6vpn unicast updates. These BGP updates carry tunnel encapsulate attributes. The BGP L3VPN separates the threat mitigation prefix from the internet routes and does not impact the internet route's best path selection, thus minimizing the probability of forming a routing loop. In Junos OS 20.3R1 and later releases, you can choose to use VPN over MPLS transport or switch to IP over IP tunnel according to the tunnel attribute. If the receiver router is running on Junos OS 20.2 or previous releases, the path attribute is ignored and uses the traditional MPLS transport service.

To use VPN over an IP-IP tunnel, you need to configure a tunnel-attribute. When routes are advertised directly from the VRF table, you can use a BGP or VRF export policy to attach the tunnel attribute. When the advertise-from-main-vpn-tables statement is enabled, or when the device acts as a route reflector or an AS boundary router, you must attach the tunnel attribute using a BGP export policy under BGP.

You can configure the receiver to program the dynamic tunnel using the tunnel attribute to enable tunnel-attribute on trusted BGP peers. The route reflector is able to reflect the route with the tunnel attribute even if this is not configured.

2

CHAPTER

Encryption Services

IN THIS CHAPTER

- [Encryption Services Overview | 169](#)
 - [Configuring Encryption Interfaces | 169](#)
-

Encryption Services Overview

The IP Security (IPsec) architecture provides a security suite for the IP version 4 (IPv4) and IP version 6 (IPv6) network layers. The suite provides functionality such as authentication of origin, data integrity, confidentiality, replay protection, and nonrepudiation of source. It also defines mechanisms for key generation and exchange, management of security associations, and support for digital certificates.

IPsec defines a security association (SA) and key management framework that can be used with any network layer protocol. The SA specifies what protection policy to apply to traffic between two IP-layer entities. For more information, see the [Junos OS Administration Library for Routing Devices](#). The standards are defined in the following RFCs:

- RFC 2401, *Security Architecture for the Internet Protocol*
- RFC 2406, *IP Encapsulating Security Payload (ESP)*

Configuring Encryption Interfaces

IN THIS SECTION

- [Configuring Encryption Interfaces | 170](#)
- [Configuring Filters for Traffic Transiting the ES PIC | 172](#)
- [Configuring an ES Tunnel Interface for a Layer 3 VPN | 179](#)
- [Configuring ES PIC Redundancy | 179](#)
- [Configuring IPsec Tunnel Redundancy | 180](#)

Configuring Encryption Interfaces

IN THIS SECTION

- [Specifying the Security Association Name for Encryption Interfaces | 171](#)
- [Configuring the MTU for Encryption Interfaces | 171](#)
- [Example: Configuring an Encryption Interface | 171](#)

When you configure the encryption interface, you associate the configured SA with a logical interface. This configuration defines the tunnel, including the logical unit, tunnel addresses, maximum transmission unit (MTU), optional interface addresses, and the name of the IPsec SA to apply to traffic. To configure an encryption interface, include the following statements at the `[edit interfaces es-fpc/pic/port unit logical-unit-number]` hierarchy level:

```
family inet {
  ipsec-sa ipsec-sa; # name of security association to apply to packet
  address address;      # local interface address inside local VPN
  destination address;  # destination address inside remote VPN
}
tunnel {
  source source-address;
  destination destination-address;
}
```

The addresses configured as the tunnel source and destination are the addresses in the outer IP header of the tunnel.



NOTE: You must configure the tunnel source address locally on the router, and the tunnel destination address must be a valid address for the security gateway terminating the tunnel.

The ES Physical Interface Card (PIC) is supported on M Series and T Series routers.

The SA must be a valid tunnel-mode SA. The interface address and destination address listed are optional. The destination address allows the user to configure a static route to encrypt traffic. If a static route uses that destination address as the next hop, traffic is forwarded through the portion of the tunnel in which encryption occurs.

Specifying the Security Association Name for Encryption Interfaces

The security association is the set of properties that defines the protocols for encrypting Internet traffic. To configure encryption interfaces, you specify the SA name associated with the interface by including the `ipsec-sa` statement at the `[edit interfaces es-fpc/pic/port unit logical-unit-number family inet]` hierarchy level:

```
ipsec-sa sa-name;
```

For information about configuring the security association, see ["Configuring Filters for Traffic Transiting the ES PIC" on page 172](#).

Configuring the MTU for Encryption Interfaces

The protocol MTU value for encryption interfaces must always be less than the default interface MTU value of 3900 bytes; the configuration fails to commit if you select a greater value. To set the MTU value, include the `mtu` statement at the `[edit interfaces interface-name unit logical-unit-number family inet]` hierarchy level:

```
mtu bytes;
```

For more information, see the [Junos OS Network Interfaces Library for Routing Devices](#).

Example: Configuring an Encryption Interface

Configure an IPsec tunnel as a logical interface on the ES PIC. The logical interface specifies the tunnel through which the encrypted traffic travels. The `ipsec-sa` statement associates the security profile with the interface.

```
[edit interfaces]
es-0/0/0 {
  unit 0 {
    tunnel {
      source 10.5.5.5;           # tunnel source address
      destination 10.6.6.6;     # tunnel destination address
    }
    family inet {
      ipsec-sa manual-sa1; # name of security association to apply to packet
      mtu 3800;
      address 10.1.1.8/32 { # local interface address inside local VPN
```

```

        destination 10.2.2.254; # destination address inside remote VPN
    }
}
}

```

Configuring Filters for Traffic Transiting the ES PIC

IN THIS SECTION

- [Traffic Overview | 172](#)
- [Configuring the Security Association | 174](#)
- [Configuring an Outbound Traffic Filter | 174](#)
- [Applying the Outbound Traffic Filter | 176](#)
- [Configuring an Inbound Traffic Filter | 176](#)
- [Applying the Inbound Traffic Filter to the Encryption Interface | 177](#)

This section contains the following topics:

Traffic Overview

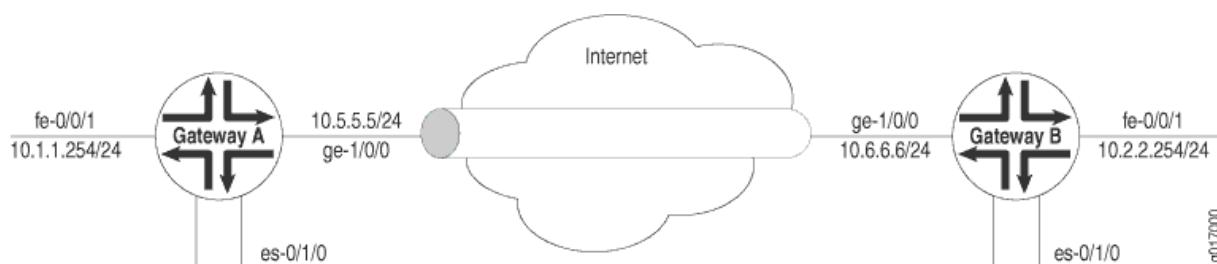
Traffic configuration defines the traffic that must flow through the tunnel. You configure outbound and inbound firewall filters, which identify and direct traffic to be encrypted and confirm that decrypted traffic parameters match those defined for the given tunnel. The outbound filter is applied to the LAN or WAN interface for the incoming traffic you want to encrypt. The inbound filter is applied to the ES PIC to check the policy for traffic coming in from the remote host. Because of the complexity of configuring a router to forward packets, no automatic checking is done to ensure that the configuration is correct.



NOTE: The valid firewall filters statements for IPsec are destination-port, source-port, protocol, destination-address, and source-address.

In [Figure 7 on page 173](#), Gateway A protects the network 10.1.1.0/24, and Gateway B protects the network 10.2.2.0/24. The gateways are connected by an IPsec tunnel. For more information about firewalls, see the [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#).

Figure 7: Example: IPsec Tunnel Connecting Security Gateways



The SA and ES interface for security Gateway A are configured as follows:

```
[edit security ipsec]
security-association manual-sa1 {
  manual {
    direction bidirectional {
      protocol esp;
      spi 2312;
      authentication {
        algorithm hmac-md5-96;
        key ascii-text 1234123412341234;
      }
      encryption {
        algorithm 3des-cbc;
        key ascii-text 123456789009876543211234;
      }
    }
  }
}
[edit interfaces es-0/1/0]
unit 0 {
  tunnel {
    source 10.5.5.5;
    destination 10.6.6.6;
  }
  family inet {
    ipsec-sa manual-sa1;
    address 10.1.1.8/32 {
      destination 10.2.2.254;
    }
  }
}
```

Configuring the Security Association

To configure the SA, include the security-association statement at the [edit security] hierarchy level:

```
security-association name {
  mode (tunnel | transport);
  manual {
    direction (inbound | outbound | bi-directional) {
      auxiliary-spi auxiliary-spi-value;
      spi spi-value;
      protocol (ah | esp | bundle);
      authentication {
        algorithm (hmac-md5-96 | hmac-sha1-96);
        key (ascii-text key | hexadecimal key);
      }
      encryption {
        algorithm (des-cbc | 3des-cbc);
        key (ascii-text key | hexadecimal key);
      }
    }
    dynamic {
      replay-window-size (32 | 64);
      ipsec-policy policy-name;
    }
  }
}
```

For more information about configuring an SA, see the [Junos OS Administration Library for Routing Devices](#). For information about applying the SA to an interface, see [147531 "Specifying the Security Association Name for Encryption Interfaces" on page 170](#).

Configuring an Outbound Traffic Filter

To configure the outbound traffic filter, include the filter statement at the [edit firewall] hierarchy level:

```
filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
```

```

        action;
        action-modifiers;
    }
}
}

```

For more information, see the [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#).

Example: Configuring an Outbound Traffic Filter

Firewall filters for outbound traffic direct the traffic through the desired IPsec tunnel and ensure that the tunneled traffic goes out the appropriate interface (see [Figure 7 on page 173](#)). Here, an outbound firewall filter is created on security Gateway A; it identifies the traffic to be encrypted and adds it to the input side of the interface that carries the internal virtual private network (VPN) traffic:

```

[edit firewall]
filter ipsec-encrypt-policy-filter {
    term term1 {
        from {
            source-address {          # local network
                10.1.1.0/24;
            }
            destination-address {     # remote network
                10.2.2.0/24;
            }
        }
    }
    then ipsec-sa manual-sa1;        # apply SA name to packet
    term default {
        then accept;
    }
}

```



NOTE: The source address, port, and protocol on the outbound traffic filter must match the destination address, port, and protocol on the inbound traffic filter. The destination address, port, and protocol on the outbound traffic filter must match the source address, port, and protocol on the inbound traffic filter.

Applying the Outbound Traffic Filter

After you have configured the outbound firewall filter, you apply it by including the filter statement at the [edit interfaces *interface-name* unit *logical-unit-number* family inet] hierarchy level:

```
filter {
    input filter-name;
}
```

Example: Applying the Outbound Traffic Filter

Apply the outbound traffic filter. The outbound filter is applied on the Fast Ethernet interface at the [edit interfaces fe-0/0/1 unit 0 family inet] hierarchy level. Any packet matching the IPsec action term (term 1) on the input filter (ipsec-encrypt-policy-filter), configured on the Fast Ethernet interface, is directed to the ES PIC interface at the [edit interfaces es-0/1/0 unit 0 family inet] hierarchy level. So, if a packet arrives from the source address 10.1.1.0/24 and goes to the destination address 10.2.2.0/24, the Packet Forwarding Engine directs the packet to the ES PIC interface, which is configured with the manual-sa1 SA. The ES PIC receives the packet, applies the manual-sa1 SA, and sends the packet through the tunnel.

The router must have a route to the tunnel end point; add a static route if necessary.

```
[edit interfaces]
fe-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input ipsec-encrypt-policy-filter;
            }
            address 10.1.1.254/24;
        }
    }
}
```

Configuring an Inbound Traffic Filter

To configure an inbound traffic filter, include the filter statement at the [edit firewall] hierarchy level:

```
filter filter-name {
    term term-name {
```



```

        from {
            match-conditions;
        }
        then {
            action;
            action-modifiers;
        }
    }
}

```

For more information, see the [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#).

Example: Configuring an Inbound Traffic Filter

Configure an inbound firewall filter. This filter performs the final IPsec policy check and is created on security gateway A. The policy check ensures that only packets that match the traffic configured for this tunnel are accepted.

```

[edit firewall]
filter ipsec-decrypt-policy-filter {
    term term1 {                                # perform policy check
        from {
            source-address {                    # remote network
                10.2.2.0/24;
            }
            destination-address {                # local network
                10.1.1.0/24;
            }
        }
    }
    then accept;
}

```

Applying the Inbound Traffic Filter to the Encryption Interface

After you create the inbound firewall filter, you can apply it to the ES PIC. To apply the filter to the ES PIC, include the filter statement at the [edit interfaces es-fpc/pic/port unit *logical-unit-number* family inet filter] hierarchy level:

```

filter {
    input filter;
}

```

The input filter is the name of the filter applied to received traffic. For a configuration example, see ["Example: Configuring an Inbound Traffic Filter" on page 177](#). For more information about firewall filters, see the [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#).

Example: Applying the Inbound Traffic Filter to the Encryption Interface

Apply the inbound firewall filter (ipsec-decrypt-policy-filter) to the decrypted packet to perform the final policy check. The IPsec manual-sa1 SA is referenced at the [edit interfaces es-1/2/0 unit 0 family inet] hierarchy level and decrypts the incoming packet.

The Packet Forwarding Engine directs IPsec packets to the ES PIC. It uses the packet's security parameter index (SPI), protocol, and destination address to look up the SA configured on one of the ES interfaces. The IPsec manual-sa1 SA is referenced at the [edit interfaces es-1/2/0 unit 0 family inet] hierarchy level and is used to decrypt the incoming packet. When the packets are processed (decrypted, authenticated, or both), the input firewall filter (ipsec-decrypt-policy-filter) is applied on the decrypted packet to perform the final policy check. term1 defines the decrypted (and verified) traffic and performs the required policy check. For information about term1, see ["Example: Configuring an Inbound Traffic Filter" on page 177](#).



NOTE: The inbound traffic filter is applied after the ES PIC has processed the packet, so the decrypted traffic is defined as any traffic that the remote gateway is encrypting and sending to this router. IKE uses this filter to determine the policy required for a tunnel. This policy is used during the negotiation with the remote gateway to find the matching SA configuration.

```
[edit interfaces]
es-1/2/0 {
  unit 0 {
    tunnel {
      source 10.5.5.5;                # tunnel source address
      destination 10.6.6.6;          # tunnel destination address
    }
    family inet {
      filter {
        input ipsec-decrypt-policy-filter;
      }
      ipsec-sa manual-sa1;            # SA name applied to packet
      address 10.1.1.8/32 { # local interface address inside local VPN
        destination 10.2.2.254;      # destination address inside remote VPN
      }
    }
  }
}
```

```
}
}
```

Configuring an ES Tunnel Interface for a Layer 3 VPN

To configure an ES tunnel interface for a Layer 3 VPN, you need to configure an ES tunnel interface on the provider edge (PE) router and on the customer edge (CE) router. You also need to configure IPsec on the PE and CE routers. For more information about configuring an ES tunnel for a Layer 3 VPN, see the [Junos OS VPNs Library for Routing Devices](#).

Configuring ES PIC Redundancy

IN THIS SECTION

- [Example: Configuring ES PIC Redundancy | 180](#)

You can configure ES PIC redundancy on M Series and T Series routers that have multiple ES PICs. With ES PIC redundancy, one ES PIC is active and another ES PIC is on standby. When the primary ES PIC has a servicing failure, the backup becomes active, inherits all the tunnels and SAs, and acts as the new next hop for IPsec traffic. Reestablishment of tunnels on the backup ES PIC does not require new Internet Key Exchange (IKE) negotiations. If the primary ES PIC comes online, it remains in standby and does not preempt the backup. To determine which PIC is currently active, use the `show ipsec redundancy` command.



NOTE: ES PIC redundancy is supported on M Series and T Series routers.

To configure an ES PIC as the backup, include the `backup-interface` statement at the `[edit interfaces fpc/pic/port es-options]` hierarchy level:

```
backup-interface es-fpc/pic/port;
```

Example: Configuring ES PIC Redundancy

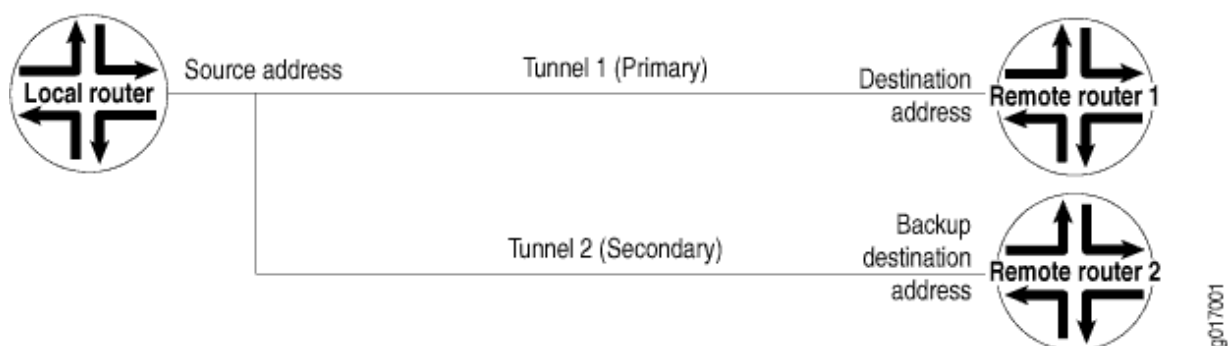
After you create the inbound firewall filter, apply it to the primary ES PIC. Here, the inbound firewall filter (ipsec-decrypt-policy-filter) is applied on the decrypted packet to perform the final policy check. The IPsec manual-sa1 SA is referenced at the [edit interfaces es-1/2/0 unit 0 family inet] hierarchy level and decrypts the incoming packet. This example does not show SA and filter configuration. For information about SA and filter configuration, see the [Junos OS Administration Library for Routing Devices](#), the [Routing Policies, Firewall Filters, and Traffic Policers User Guide](#), and "Example: Configuring an Inbound Traffic Filter" on page 172.

```
[edit interfaces]
es-1/2/0 {
  es-options {
    backup-interface es-1/0/0;
  }
  unit 0 {
    tunnel {
      source 10.5.5.5;
      destination 10.6.6.6;
    }
    family inet {
      ipsec-sa manual-sa1;
      filter {
        input ipsec-decrypt-policy-filter;
      }
      address 10.1.1.8/32 {
        destination 10.2.2.254;
      }
    }
  }
}
```

Configuring IPsec Tunnel Redundancy

You can configure IPsec tunnel redundancy by specifying a backup destination address. The local router sends keepalives to determine the remote site's reachability. When the peer is no longer reachable, a new tunnel is established. For up to 60 seconds during failover, traffic is dropped without notification being sent. [Figure 8 on page 181](#) shows IPsec primary and backup tunnels.

Figure 8: IPsec Tunnel Redundancy



To configure IPsec tunnel redundancy, include the backup-destination statement at the [edit interfaces unit *logical-unit-number* tunnel] hierarchy level:

```
backup-destination address;  
destination address;  
source address;
```



NOTE: Tunnel redundancy is supported on M Series and T Series routers.
The primary and backup destinations must be on different routers.
The tunnels must be distinct from each other and policies must match.

For more information about tunnels, see ["Tunnel Interface Configuration on MX Series Routers Overview" on page 16](#).

3

CHAPTER

Configuration Statements and Operational Commands

IN THIS CHAPTER

- [Junos CLI Reference Overview](#) | 183
-

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- [Junos CLI Reference](#)

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- [Configuration Statements](#)
- [Operational Commands](#)