



DANOS-Vyatta edition

Disaggregated Network Operating System Version 2009a

LAN Interfaces Configuration Guide
October 2020

Contents

Chapter 1. Copyright Statement	1
Chapter 2. Preface	2
Document conventions.....	2
Chapter 3. About This Guide	4
Chapter 4. Loopback Interface	5
Loopback interface overview.....	5
Examples of loopback interface configuration.....	6
Configuring network addresses.....	6
IPv6 on the loopback interface.....	7
Related commands in other guides.....	7
Chapter 5. Loopback Interface Commands	8
clear interfaces loopback counters.....	8
interfaces loopback.....	8
interfaces loopback address.....	9
interfaces loopback description.....	10
interfaces loopback ipv6 address.....	10
interfaces loopback ipv6 disable.....	11
interfaces loopback ipv6 disable-forwarding.....	12
interfaces loopback ipv6 dup-addr-detect-transmits.....	13
interfaces loopback ipv6 ospfv3 area.....	14
interfaces loopback lo redirect.....	15
show interfaces loopback.....	15
Chapter 6. Data Plane Interfaces	18
Data plane interfaces overview.....	18
Examples of data plane interface configuration.....	18
Viewing system interfaces.....	18
Basic configuration of a data plane interface.....	20
Ethernet multinetting.....	21
IPv6 on data plane interfaces.....	21
Hardware binding for data plane interfaces.....	21

Chapter 7. Data Plane Interfaces Commands.....	25
clear interfaces dataplane counters.....	25
interfaces dataplane.....	25
interfaces dataplane address.....	26
interfaces dataplane bond-group.....	27
interfaces dataplane description.....	28
interfaces dataplane dhcp-options no-rfc3442.....	28
interfaces dataplane dhcpv6-options.....	30
interfaces dataplane disable.....	31
interfaces dataplane disable-link-detect.....	31
interfaces dataplane hardware dev-port.....	32
interfaces dataplane hardware firmware-index.....	33
interfaces dataplane hardware mac.....	33
interfaces dataplane hardware pci-address.....	34
interfaces dataplane hardware pci-function.....	35
interfaces dataplane hardware pci-slot.....	36
interfaces dataplane hardware port.....	37
interfaces dataplane ip disable-forwarding.....	37
interfaces dataplane ip enable-proxy-arp.....	38
interfaces dataplane ip pim mode.....	39
interfaces dataplane ip rpf-check.....	39
interfaces dataplane ip tcp-mss limit.....	40
interfaces dataplane ip tcp-mss mtu.....	41
interfaces dataplane ip tcp-mss mtu-minus.....	42
interfaces dataplane ipv6 address.....	42
interfaces dataplane ipv6 disable.....	44
interfaces dataplane ipv6 disable-forwarding.....	45
interfaces dataplane ipv6 dup-addr-detect-transmits.....	45
interfaces dataplane ipv6 router-advert.....	46
interfaces dataplane ipv6 tcp-mss limit.....	50
interfaces dataplane ipv6 tcp-mss mtu.....	50
interfaces dataplane ipv6 tcp-mss mtu-minus.....	51
interfaces dataplane mac.....	52

interfaces dataplane mtu.....	52
interfaces dataplane sflow.....	53
interfaces dataplane speed.....	54
interfaces dataplane switch-group port-parameters mode.....	55
interfaces dataplane switch-group port-parameters vlan-parameters vlans.....	56
interfaces dataplane switch-group switch.....	57
interfaces dataplane vif dhcp-options no-rfc3442.....	57
interfaces dataplane vif policy qos.....	58
interfaces dataplane vrrp vrrp-group notify bgp.....	59
monitor interfaces dataplane traffic.....	60
show interfaces dataplane.....	61
show interfaces dataplane detail.....	62
show interfaces dataplane brief.....	63
show interfaces dataplane physical.....	64
Related commands documented elsewhere.....	65
Chapter 8. Ethernet Link Bonding Interface.....	66
Ethernet link bonding overview.....	66
Ethernet bonding configuration examples.....	67
Basic Ethernet bonding.....	67
Ethernet bonding with VLAN.....	68
Chapter 9. Ethernet Link Bonding Interface Commands.....	70
interfaces bonding.....	70
interfaces bonding address.....	70
interfaces bonding description.....	71
interfaces bonding dhcp-options no-rfc3442.....	72
interfaces bonding dhcpv6-options.....	73
interfaces bonding disable.....	74
interfaces bonding flow-monitoring.....	75
interfaces bonding ip enable-proxy-arp.....	75
interfaces bonding ip rip receive.....	76
interfaces bonding ip rip send.....	77
interfaces bonding ip tcp-mss limit.....	78
interfaces bonding ip tcp-mss mtu.....	79

interfaces bonding ip tcp-mss mtu-minus.....	79
interfaces bonding ipv6 address.....	80
interfaces bonding ipv6 ospfv3 area.....	81
interfaces bonding ipv6 ospfv3 process instance-id.....	82
interfaces bonding ipv6 tcp-mss limit.....	83
interfaces bonding ipv6 tcp-mss mtu.....	83
interfaces bonding ipv6 tcp-mss mtu-minus.....	84
interfaces bonding lacp-options activity.....	85
interfaces bonding lacp-options key.....	85
interfaces bonding mac.....	86
interfaces bonding mode.....	87
interfaces bonding mtu.....	88
interfaces bonding primary.....	89
show interfaces bonding.....	90
Related commands documented elsewhere.....	92
Chapter 10. VLAN Interfaces.....	93
VLAN interface overview.....	93
VLAN operation using virtual interfaces.....	93
Interface types that support VLAN operation.....	93
VLAN operation as opposed to multinetting.....	93
Simultaneous Ethernet and 802.1q operation.....	94
Referring to VLAN interfaces in commands.....	94
IPv6 support.....	94
Examples of VLAN interface configuration.....	94
VLAN configuration.....	94
IPv6 on VLAN interfaces.....	96
Chapter 11. VLAN Interfaces Commands.....	97
interfaces bonding vif ip tcp-mss limit.....	97
interfaces bonding vif ip tcp-mss mtu.....	97
interfaces bonding vif ip tcp-mss mtu-minus.....	98
interfaces bonding vif ipv6 tcp-mss limit.....	99
interfaces bonding vif ipv6 tcp-mss mtu.....	100
interfaces bonding vif ipv6 tcp-mss mtu-minus.....	101

interfaces dataplane vif.....	101
interfaces dataplane vif address.....	102
interfaces dataplane vif bridge-group.....	103
interfaces dataplane vif description.....	104
interfaces dataplane vif dhcpv6-options.....	105
interfaces dataplane vif disable.....	106
interfaces dataplane vif disable-link-detect.....	107
interfaces dataplane vif ip disable-forwarding.....	108
interfaces dataplane vif ip enable-proxy-arp.....	108
interfaces dataplane vif ip rip receive.....	109
interfaces dataplane vif ip rip send.....	110
interfaces dataplane vif ip rpf-check.....	111
interfaces dataplane vif ip tcp-mss limit.....	112
interfaces dataplane vif ip tcp-mss mtu.....	113
interfaces dataplane vif ip tcp-mss mtu-minus.....	113
interfaces dataplane vif ipv6 address.....	114
interfaces dataplane vif ipv6 disable.....	116
interfaces dataplane vif ipv6 disable-forwarding.....	116
interfaces dataplane vif ipv6 dup-addr-detect-transmits.....	117
interfaces dataplane vif ipv6 mld.....	118
interfaces dataplane vif ipv6 ospfv3.....	119
interfaces dataplane vif ipv6 pim.....	120
interfaces dataplane vif ipv6 router-advert.....	121
interfaces dataplane vif ipv6 tcp-mss limit.....	125
interfaces dataplane vif ipv6 tcp-mss mtu.....	125
interfaces dataplane vif ipv6 tcp-mss mtu-minus.....	126
interfaces dataplane vif ipv6 unnumbered donor-interface preferred-address.....	127
interfaces dataplane vif mtu.....	128
interfaces dataplane vif vlan.....	129
interfaces dataplane vif inner-vlan.....	130
interfaces switch default-port-parameters mode.....	131
interfaces switch default-port-parameters vlan-parameters vlans.....	131
interfaces switch physical-switch.....	132

interfaces dataplane interface duplex.....	133
interfaces loopback interface ipv6 disable.....	133
interfaces loopback interface ipv6 ospfv3 instance.....	134
interfaces dataplane bridge-group.....	136
interfaces dataplane interface duplex.....	137
interfaces dataplane switch-group.....	137
interfaces dataplane tagnode switch-group port-parameters vlan-parameters.....	138
interfaces dataplane tagnode switch-group port-parameters policy.....	139
interfaces dataplane vif bridge-group.....	140
interfaces l2tpeth vif ip tcp-mss limit.....	141
interfaces l2tpeth vif ip tcp-mss mtu.....	142
interfaces l2tpeth vif ip tcp-mss mtu-minus.....	143
interfaces l2tpeth vif ipv6 tcp-mss limit.....	144
interfaces l2tpeth vif ipv6 tcp-mss mtu.....	144
interfaces l2tpeth vif ipv6 tcp-mss mtu-minus.....	145
interfaces switch default-port-parameters vlan-parameters.....	146
Related commands documented elsewhere.....	147
policy vlan-modify rule.....	148
policy vlan-modify rule action.....	148
policy vlan-modify rule action.....	149
policy vlan-modify rule action swap.....	150
policy vlan-modify rule selector.....	151
show interfaces dataplane vif.....	152
show policy vlan-modify.....	152
Chapter 12. Switch Virtual Interfaces.....	154
interfaces switch vif address.....	154
interfaces switch vif description.....	155
interfaces switch vif ip ospf authentication key-chain.....	155
interfaces switch vif ip ospf fall-over bfd.....	156
interfaces switch vif ip ospf network.....	157
interfaces switch vif ipv6 ospfv3 area.....	158
interfaces switch vif ip ospfv3 fall-over bfd.....	159
interfaces switch vif ip ospfv3 network.....	160

Chapter 13. Q-in-Q Interface.....	162
Q-in-Q interface overview.....	162
Q-in-Q interface features.....	162
Packet processing using QoS schedulers.....	163
Configuring the Q-in-Q interface.....	163
Disabling the interface.....	164
Ignoring link state changes.....	164
Viewing interface queuing.....	164
Chapter 14. Q-in-Q Interface Commands.....	165
interfaces dataplane vif inner-vlan.....	165
interfaces dataplane vif vlan.....	165
interfaces dataplane vlan-protocol.....	166
policy qos name shaper vlan.....	167
Chapter 15. IP Unnumbered Interfaces.....	169
Overview.....	169
Limitations of the feature.....	169
How unnumbered interfaces work.....	169
Unnumbered interface configuration example.....	170
Basic IP unnumbered interface configuration.....	170
Configuring BGP on an unnumbered interface.....	171
Configuring OSPF on an unnumbered interface.....	173
Chapter 16. IP Unnumbered Interfaces Command.....	175
interfaces dataplane vif ip unnumbered donor-interface.....	175
Chapter 17. L2TPv3 Data Plane Interfaces.....	177
Overview.....	177
LAC-to-LAC tunneling reference model.....	177
LAC-to-LNS tunneling reference model.....	178
LNS-to-LNS tunneling reference model.....	178
Configuration commands.....	178
Before configuring an L2TP tunnel.....	179
L2TPv3 configuration examples.....	179
Before you begin.....	179
Configuring a LAC-to-LAC IPv4 tunnel.....	179

Configuring a LAC-to-LAC IPv6 tunnel.....	183
Configuring an LNS-to-LNS IPv4 tunnel.....	186
Configuring an LNS-to-LNS tunnel with BGP routes.....	189
Configuring an LNS-to-LNS tunnel with OSPF routes.....	193
Configuring an LNS-to-LNS tunnel with RIP routes.....	196
Chapter 18. L2TPv3 Data Plane Interfaces Commands.....	200
interfaces l2tpeth ip tcp-mss limit.....	200
interfaces l2tpeth ip tcp-mss mtu.....	200
interfaces l2tpeth ip tcp-mss mtu-minus.....	201
interfaces l2tpeth ipv6 tcp-mss limit.....	202
interfaces l2tpeth ipv6 tcp-mss mtu.....	203
interfaces l2tpeth ipv6 tcp-mss mtu-minus.....	203
interfaces l2tpeth l2tp-session encapsulation.....	204
interfaces l2tpeth l2tp-session local-cookie.....	205
interfaces l2tpeth l2tp-session local-ip.....	205
interfaces l2tpeth l2tp-session local-session-id.....	206
interfaces l2tpeth l2tp-session local-udp-port.....	207
interfaces l2tpeth l2tp-session remote-cookie.....	207
interfaces l2tpeth l2tp-session remote-ip.....	208
interfaces l2tpeth l2tp-session remote-session-id.....	208
interfaces l2tpeth l2tp-session remote-udp-port.....	209
interfaces dataplane xconnect l2tpeth.....	210
interfaces dataplane vif xconnect l2tpeth.....	210
show interfaces.....	211
show l2tpeth.....	212
show l2tpeth.....	212
Chapter 19. Data Plane Interface Names.....	214
Chapter 20. TCP Maximum Segment Size.....	215
Overview.....	215
Chapter 21. List of Acronyms.....	218

Chapter 1. Copyright Statement

© 2020 IP Infusion Inc. All Rights Reserved.

This documentation is subject to change without notice. The software described in this document and this documentation are furnished under a license agreement or nondisclosure agreement. The software and documentation may be used or copied only in accordance with the terms of the applicable agreement. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's internal use without the written permission of IP Infusion Inc.

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900

<http://www.ipinfusion.com/>.

For support, questions, or comments via E-mail, contact:

support@ipinfusion.com.

Trademarks:

IP Infusion is a trademark of IP Infusion. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Use of certain software included in this equipment is subject to the IP Infusion, Inc. End User License Agreement at <http://www.ipinfusion.com/license>. By using the equipment, you accept the terms of the End User License Agreement.


Chapter 2. Preface


Document conventions


The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in this document.


Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

 **Note:** A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

 **Attention:** An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

 **CAUTION:** A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

 **DANGER:** A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font are used to highlight specific words or phrases.

Format	Description
bold text	Identifies command names. Identifies keywords and operands.
<i>italic text</i>	Identifies emphasis. Identifies variables. Identifies document titles.
<code>Courier font</code>	Identifies CLI output. Identifies command syntax examples.

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
<i>italic text</i>	Identifies a variable.
[]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ x y z }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x y	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, <i>member</i> [<i>member</i> ...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Chapter 3. About This Guide

This guide describes how to configure LAN interfaces on DANOS-Vyatta edition.

Chapter 4. Loopback Interface

Loopback interface overview

A loopback interface is a special software-only interface that emulates a physical interface and allows the router to “connect” to itself. Packets routed to the loopback interface are rerouted back to the router and processed locally. Packets routed out the loopback interface but not destined for the loopback interface are dropped.

The router supports multiple loopback interfaces. These interfaces, with unique IP addressing, can be used as preferred source addresses for routing protocols such as BGP. These interfaces can also be configured as null or blackhole interfaces.

The router supports multiple IPv4 and IPv6 addresses on each loopback interface. These interfaces (lo and lo1 through loN) have unique IP addressing, can be used as preferred source addresses for routing protocols such as BGP, and can be configured as null or blackhole interfaces.

The loopback interface provides a number of advantages.

- As long as the router is functioning, the loopback interface is always up, and so is very reliable. When even only one link to the router is functioning, the loopback interface can be accessed. The loopback interface thus eliminates the need to try each IP address of the router until it finds one that is still up.
- Because the loopback interface is always up, a routing session (such as a BGP session) can continue even if the outbound interface fails.
- You can simplify collection of management information by specifying the loopback interface as the interface for sending and receiving management information such as logs and SNMP traps.
- The loopback interface can be used to increase security by filtering incoming traffic with access control rules that specify the local interface as the only acceptable destination.
- In OSPF, you can advertise a loopback interface as an interface route into the network, regardless of whether physical links are up or down. This increases reliability by allowing traffic to take alternate paths if one or more physical links go down.
- In BGP, parallel paths can be configured to the loopback interface on a peer device. These parallel paths provide improved load sharing and redundancy.

The router automatically creates the loopback interface on startup with an interface name of lo. It also automatically configures the loopback address with standard IP addressing.

- According to RFC 5735, the 127.0.0.1/8 IPv4 address is assigned to the loopback address. This address is hidden from the `show` command output. Typically, the IPv4

address that is assigned to the loopback device is 127.0.0.1 for IPv4, although any address in the range from 127.0.0.0 through 127.255.255.255 is mapped to it.

- According to RFC 3513, the ::1/128 IPv6 address is assigned to the loopback interface.
- According to RFC 2606, the localhost domain name is mapped to the loopback addresses.

When configuring the router, it is good practice to take advantage of the reliability of the loopback interface with these practices:

- The host name of the router should be mapped to the loopback interface address, rather than to a physical interface.
- In OSPF and BGP, the router ID should be set to the loopback address. This prevents a possible dynamic recalculation and reassignment of the loopback address when physical interfaces are added or removed from the system. This action is disruptive to active BGP and OSPF sessions.

The router has extensive support for IPv6, including IPv6 interface addressing. The commands for configuring IPv6 on the loopback interface are given in this chapter. A full description of IPv6 support is provided in the *IPv6 Support Configuration Guide*.

Examples of loopback interface configuration

This section presents the following topics:

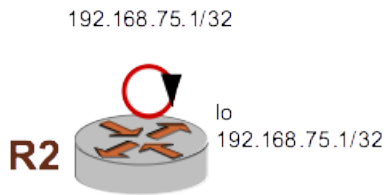
- [Configuring network addresses](#)
- [IPv6 on the loopback interface](#)

Configuring network addresses

The system automatically creates and addresses the loopback interface, so you do not need to configure any additional addressing. If you delete the loopback node, the system re-creates and readdresses the loopback address again the next time the system starts.

You may at times want to configure a smaller network prefix than /8 to the loopback interface. The example in this section shows how to assign the 192.168.75.1/32 address to the loopback interface. When you finish the example, the interface is configured as in the following figure.

Figure 1. Configuring the loopback interface



To configure the loopback interface, perform the following steps in configuration mode.

Table 1. Configuring the loopback interface

Step	Command
Assign the IP address to the loopback interface.	<pre>vyatta@R1# set interfaces loopback lo address 192.168.75.1/32</pre>
Commit the configuration.	<pre>vyatta@R1# commit</pre>
View the configuration.	<pre>vyatta@R1# show interfaces loopback loopback lo { address 192.168.75.1/32 }</pre>

IPv6 on the loopback interface

IPv6 Support Configuration Guide provides examples of configuring IPv6 on interfaces.

Related commands in other guides

Commands for using other system features with loopback interfaces are located in the following guides.

Related Commands Documented Elsewhere	
OSPF	OSPF is supported on the loopback interface. Commands for configuring OSPF are described in <i>OSPF Configuration Guide</i> .
QoS	QoS traffic policies are supported on the loopback interface. Commands for configuring quality of service on the loopback interface are described in <i>QoS Configuration Guide</i> .
RIP	RIP is supported on the loopback interface. Commands for configuring RIP are described in <i>RIP Configuration Guide</i> .
RIP-ng	RIPng is supported on the loopback interface. Commands for configuring RIPng are described in <i>RIPng Configuration Guide</i> .

Chapter 5. Loopback Interface Commands

clear interfaces loopback counters

Clears statistics counters for loopback interfaces.

```
clear interfaces loopback [ interface-name ] counters
```

The statistics counters for all loopback interfaces are cleared.

interface-name

Clears statistics counters for a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999. If an interface is not specified, this command clears the counters of all loopback interfaces.

Operational mode

Use this command to clear statistics counters for loopback interfaces.

interfaces loopback

Defines a loopback interface.

```
set interfaces loopback interface-name
```

```
delete interfaces loopback [ interface-name ]
```

```
show interfaces loopback
```

A configuration node is automatically created for the lo loopback interface on startup.

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

Configuration mode

```
interfaces {  
    loopback interface-name  
}
```

Use this command to configure a loopback interface.

Use the `set` form of this command to create a loopback interface. However, the system automatically creates a configuration node for the lo loopback interface on startup, so you should not need to use the `set` form of this command to create the lo loopback interface unless you have deleted it.

Use the `delete` form of this command to remove all loopback interfaces or just one. The system creates an empty configuration node for the lo interface the next time the system starts.

Use the `show` form of this command to display the configured loopback interfaces.

interfaces loopback address

Specifies the IP address and network prefix for a loopback interface.

```
set interfaces loopback interface-name address { ipv4 | ipv6 }
```

```
delete interfaces loopback interface-name address { ipv4 | ipv6 }
```

```
show interfaces loopback interface-name address { ipv4 | ipv6 }
```

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

ipv4

The IPv4 address and network prefix of the loopback interface. The format is *ip-address / prefix* (for example, 127.0.0.1/8).

You can define multiple IP addresses for the loopback interface by creating multiple **address** configuration nodes.

ipv6

The IPv6 address and network prefix of the loopback interface. The format is *ipv6-address / prefix* (for example, ::1/128).

You can define multiple IPv6 addresses for the loopback interface, by creating multiple **address** configuration nodes.

Configuration mode

```
interfaces {
  loopback interface-name {
    address ipv4
    address ipv6
  }
}
```

Use the `set` form of this command to specify the IP address and network prefix for a loopback interface. You can specify more than one IP address and network prefix for a loopback interface by creating multiple **address** configuration nodes.

Use the `delete` form of this command to delete an address and network prefix for a loopback interface.

Use the `show` form of this command to display the address configuration of a loopback interface.

interfaces loopback description

Describes a loopback interface.

```
set interfaces loopback interface-name description description
```

```
delete interfaces loopback interface-name description
```

```
show interfaces loopback interface-name description
```

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

description

A description for the loopback interface.

Configuration mode

```
interfaces {
  loopback interface-name {
    description description
  }
}
```

Use this command to describe a loopback interface.

Use the `set` form of this command to describe a loopback interface.

Use the `delete` form of this command to delete the description of a loopback interface.

Use the `show` form of this command to display the description of a loopback interface.

interfaces loopback ipv6 address

Assigns an IPv6 address to a loopback interface.

```
set interfaces loopback interface-name ipv6 address [ autoconf | eui64
ipv6prefix | link-local ipv6-address ]
```

```
delete interfaces loopback interface-name ipv6 address [ autoconf | eui64
ipv6prefix | link-local ipv6-address ]
```

```
show interfaces loopback interface-name ipv6 address [ autoconf | eui64 ]
```

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

autoconf

Generates an IPv6 address by using the SLAAC protocol. Use this keyword if the interface is performing a “host” function rather than a “router” function. You can specify this keyword in addition to static IPv6, static IPv4, and IPv4 DHCP addresses on the interface.

eui64 ipv6prefix

Specifies the 64-bit IPv6 address prefix that is used to configure an IPv6 address in EUI-64 format. The system concatenates this prefix with a 64-bit EUI-64 value that is derived from the 48-bit MAC address of the interface.

link-local ipv6-address

Specifies the 128-bit IPv6 address.

Configuration mode

```
interfaces {
  loopback interface-name {
    ipv6 {
      address {
        autoconf
        eui64 ipv6prefix
        link-local ipv6-address
      }
    }
  }
}
```

Use this command to specify a method for assigning an IPv6 address to a loopback interface.

Use the **autoconf** keyword to direct the system to automatically configure (autoconfigure) the address by using the Stateless Address Autoconfiguration (SLAAC) protocol that is defined in RFC 4862. Alternatively, you can provide an EUI-64 IPv6 address prefix so that the system constructs the IPv6 address.

If you want the system to use SLAAC to acquire an address on the interface, then in addition to setting this parameter, you must also disable IPv6 forwarding, either globally (by using the `system ipv6 disable-forwarding` command) or specifically on the interface (by using the [interfaces loopback ipv6 disable-forwarding](#) command).

Use the `set` form of this command to specify a method for assigning an IPv6 address to a loopback interface.

Use the `delete` form of this command to delete an IPv6 address from a loopback interface.

Use the `show` form of this command to display IPv6 address configuration settings for a loopback interface.

```
interfaces loopback ipv6 disable
```

Disables IPv6 on a loopback interface.

```
set interfaces loopback interface-name ipv6 disable
```

```
delete interfaces loopback interface-name ipv6 disable
```

```
show interfaces loopback interface-name ipv6 disable
```

interface-name

The name of a loopback interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode.

Configuration Statement

```
interfaces {
  loopback interface-name {
    ipv6 {
      disable
    }
  }
}
```

By default, IPv6 is enabled on all interfaces. A global command exists which can disable IPv6, namely `set system ipv6 disable`, and this will take precedence over any of the existing per-interface based, IPv6 commands.

IPv6 Forwarding can be disabled via the `set interface loopback interface-name ipv6 disable-forwarding` command, but note that IPv6 traffic can still be terminated on this interface.

IPv6 configuration can be totally disabled via the `set interface loopback interface-name ipv6 disable` command.

Use the `set` form of this command to disable IPv6 on this interface.

Use the `delete` form of this command to enable IPv6 on this interface.

Use the `show` form of this command to display the current IPv6 configuration on this interface.

interfaces loopback ipv6 disable-forwarding

Disables IPv6 packet forwarding on the loopback interface.

```
set interfaces loopback interface-name ipv6 disable-forwarding
```

```
delete interfaces loopback interface-name ipv6 disable-forwarding
```

```
show interfaces loopback interface-name ipv6 disable-forwarding
```

IPv6 packets are forwarded.

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

Configuration mode

```

interfaces {
  loopback interface-name {
    ipv6 {
      disable-forwarding
    }
  }
}

```

Use this command to disable IPv6 packet forwarding on a loopback interface.

You can also disable IPv6 forwarding globally (that is, for all interfaces) by using the `system ipv6 disable-forwarding` command.

Use the `set` form of this command to disable IPv6 packet forwarding on a loopback interface.

Use the `delete` form of this command to enable IPv6 packet forwarding on a loopback interface.

Use the `show` form of this command to display the configuration of IPv6 packet forwarding on a loopback interface.

interfaces loopback ipv6 dup-addr-detect-transmits

Specifies the number of NS packets to transmit as part of the DAD process.

```

set interfaces loopback interface-name ipv6 dup-addr-detect-transmits [ 0 |
number ]

```

```

delete interfaces loopback interface-name ipv6 dup-addr-detect-transmits

```

```

show interfaces loopback interface-name ipv6 dup-addr-detect-transmits

```

One NS packet is transmitted as part of the DAD process.

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

0

Disables DAD on the loopback interface.

number

The number of NS packets to transmit as part of the DAD process. The number ranges from 1 through *n*. The default number is 1.

Configuration mode

```

interfaces {
  loopback interface-name {
    ipv6 {
      dup-addr-detect-transmits 0
      dup-addr-detect-transmits number
    }
  }
}

```

Use this command to specify the number of Neighbor Solicitation (NS) packets to transmit as part of the Duplicate Address Detection (DAD) process.

Use the `set` form of this command to specify the number of NS packets to transmit.

Use the `delete` form of this command to delete the transmission number from a loopback interface and transmit the default number of one NS packet.

Use the `show` form of this command to display the number of NS packets that are transmitted.

interfaces loopback ipv6 ospfv3 area

Assigns an OSPFv3 area to a IPv6 loopback interface.

```
set interfaces loopback interface-name ipv6 ospfv3 area { area-id | ipv4 }
```

```
delete interfaces loopback interface-name ipv6 ospfv3 area { area-id | ipv6 }
```

```
show interfaces loopback interface-name ipv6 ospfv3 area { area-id | ipv6 }
```

interface-name

The name of a loopback interface, **lo** or **lon**, where *n* ranges from 1 through 99999.

area-id

OSPFv3 area identifier in the range 0-4294967295.

ipv4

IPv4 address.

Configuration mode

```

interfaces {
  loopback {
    ipv6 {
      ospfv3 {
        area
      }
    }
  }
}

```

Use the `set` form of this command to assign an OSPFv3 area to a IPv6 loopback interface.

Use the `delete` form of this command to delete an OSPFv3 area from a IPv6 loopback interface.

Use the `show` form of this command to display the configuration of a loopback interface.

interfaces loopback lo redirect

Redirects inbound traffic from one interface to another interface.

```
set interfaces loopback lo redirect interface
```

```
delete interfaces loopback lo redirect interface
```

```
show interfaces loopback lo redirect
```

interface

The identifier of the interface to which you are redirecting data; for example, `ifb0`.


Configuration mode

```
interfaces {
  loopback lo {
    redirect interface
  }
}
```

Use this command to redirect inbound traffic from an interface to another interface.

This feature is typically used to redirect traffic from a number of interfaces to an Input interface. (Input interfaces are described in Chapter 4: Input Interface .)

Redirecting traffic from several interfaces to a single Input interface allows you to apply a single QoS policy to the combined traffic—for example, to limit the combined inbound traffic bandwidth.

 **Note:** Input interfaces are currently not supported.

Use the `set` form of this command to specify the redirect interface.

Use the `delete` form of this command to remove the redirect configuration.

Use the `show` form of this command to view the redirect configuration.

show interfaces loopback

Displays statistics and configuration information about configured loopback interfaces.

```
show interfaces loopback [ interface-name [ brief ] | detail ]
```


When used with no option, this command displays a brief status of the loopback interface.

interface-name

The name of a loopback interface, **lo** or **lo n** , where n ranges from 1 through 99999.

brief

Displays a brief status of a loopback interface.

detail

Displays detailed information and statistics about all loopback interfaces.

Operational mode

Use this command to display information about loopback interfaces.

The following example shows how to display a brief status of a loopback interface.

```
vyatta@R1:~$ show interfaces loopback lo2 brief
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface          IP Address          S/L  Description
-----          -
lo2                192.2.0.1/24       u/u
```

The following example shows how to display detailed information about a loopback interface.

```
vyatta@R1:~$ show interfaces loopback lo2
lo2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
group default
  link/ether 62:32:92:61:c4:69 brd ff:ff:ff:ff:ff:ff
  inet 192.2.0.1/24 brd 192.2.0.255 scope global lo2
    valid_lft forever preferred_lft forever
  inet6 fe80::6032:92ff:fe61:c469/64 scope link
    valid_lft forever preferred_lft forever

RX:  bytes    packets    errors    ignored    overrun    mcast
     0         0         0         0         0         0
TX:  bytes    packets    errors    dropped    carrier    collisions
     0         0         0         0         0         0
```

The following example shows how to display detailed information about all loopback interfaces.

```
vyatta@R1:~$ show interfaces loopback lo
lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
```


```
inet 192.0.0.1/24 brd 192.0.0.255 scope global lo
    valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever

RX:  bytes    packets    errors    ignored    overrun    mcast
     1058949    6632        0         0         0         0
TX:  bytes    packets    errors    dropped    carrier    collisions
     1058949    6632        0         0         0         0
lo2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
group default
link/ether 62:32:92:61:c4:69 brd ff:ff:ff:ff:ff:ff
inet 192.2.0.1/24 brd 192.2.0.255 scope global lo2
    valid_lft forever preferred_lft forever
inet6 fe80::6032:92ff:fe61:c469/64 scope link
    valid_lft forever preferred_lft forever
...
```

Chapter 6. Data Plane Interfaces

Data plane interfaces overview

A system receives packets from neighboring systems through its network interfaces. On the LAN, an interface is typically an Ethernet interface. In the router, a data plane interface is an abstraction that represents the underlying physical or virtual Ethernet interface of the system.

 **Note:** The terms Ethernet interface and data plane interface are synonymous in this guide.

Ethernet interfaces are viewed and configured in the interfaces ethernet node of the configuration tree. The system automatically discovers the physical interfaces on the system and creates entries for them in the configuration tree on startup. For example, on a system with two Ethernet interfaces, the router automatically creates configuration nodes for dp0p1p1 and dp0p1p2.

After the interface is enabled and provided with an address, you can configure it with various system features, such as firewall, routing protocols, quality of service, and so on.

The router has extensive support for IPv6, including IPv6 interface addressing. The commands for configuring IPv6 on Ethernet interfaces are given in this chapter. *IPv6 Support Configuration Guide* fully describes router IPv6 support.

Examples of data plane interface configuration

This section presents the following topics:

- [Viewing system interfaces](#)
- [Basic configuration of a data plane interface](#)
- [Ethernet multinetting](#)
- [IPv6 on data plane interfaces](#)
- [Hardware binding for data plane interfaces](#)

Viewing system interfaces

You can configure only interfaces that are physically available to the operating system on the hardware you are using. To view all the interfaces known to the operating system, use the `show interfaces system` command in operational mode, as shown in the following example. In this example, the system has two physical Ethernet interfaces, dp0p1p1 and dp0p1p2, plus a VLAN interface (vif) configured for VLAN 40 that is under dp0p1p2.

Viewing available system interfaces

```
vyatta@vyatta:~$ show interfaces system

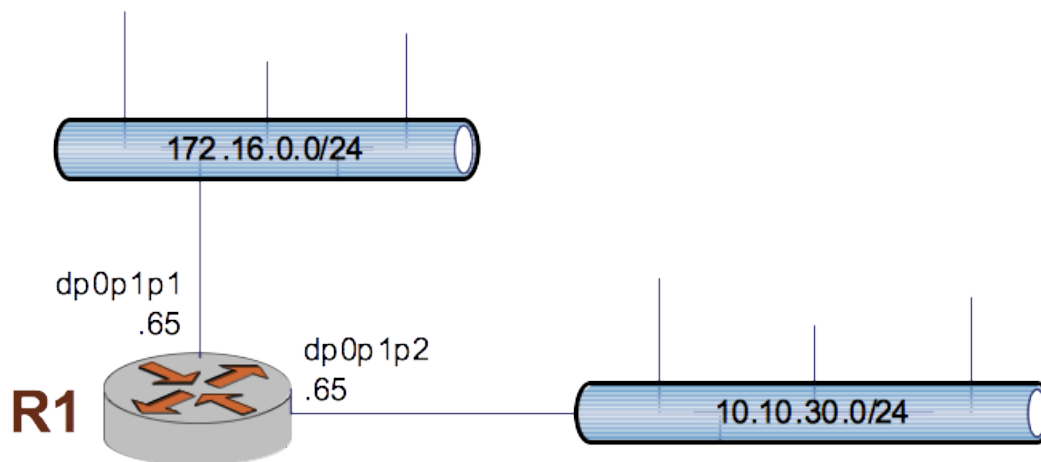
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
  DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0
    RX: bytes  packets  errors  dropped  overrun  mcast
    20843877  133515  0       0        0        0
    TX: bytes  packets  errors  dropped  carrier  collsns
    20843877  133515  0       0        0        0
6: dp0p160p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UP mode DORMANT group default qlen 500
    link/ether 00:0c:29:19:5c:20 brd ff:ff:ff:ff:ff:ff promiscuity 0
    tun
    RX: bytes  packets  errors  dropped  overrun  mcast
    7620971    125056  0       11638   0        117337
    TX: bytes  packets  errors  dropped  carrier  collsns
    152687     1589    0       0        0        0
7: dp0p192p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UP mode DORMANT group default qlen 500
    link/ether 00:0c:29:19:5c:2a brd ff:ff:ff:ff:ff:ff promiscuity 0
    tun
    RX: bytes  packets  errors  dropped  overrun  mcast
    2052       24       0       58794   0        12
    TX: bytes  packets  errors  dropped  carrier  collsns
    2278       37       0       0        0        0
8: dp0p224p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UP mode DORMANT group default qlen 500
    link/ether 00:0c:29:19:5c:34 brd ff:ff:ff:ff:ff:ff promiscuity 0
    tun
    RX: bytes  packets  errors  dropped  overrun  mcast
    876        14       0       10      0        2
    TX: bytes  packets  errors  dropped  carrier  collsns
    408        4        0       0        0        0
9: dp0p256p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UP mode DORMANT group default qlen 500
    link/ether 00:0c:29:19:5c:3e brd ff:ff:ff:ff:ff:ff promiscuity 0
    tun
    RX: bytes  packets  errors  dropped  overrun  mcast
    57684     218     0       0        0        0
    TX: bytes  packets  errors  dropped  carrier  collsns
    636       6       0       0        0        0
10: .spathintf: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UNKNOWN mode DEFAULT group default qlen 500
    link/ether 6a:9f:e0:5a:b4:de brd ff:ff:ff:ff:ff:ff promiscuity 0
    tun
    RX: bytes  packets  errors  dropped  overrun  mcast
    0          0        0       0        0        0
    TX: bytes  packets  errors  dropped  carrier  collsns
    0          0        0       0        0        0
```

Basic configuration of a data plane interface

This section presents a sample configuration for a data plane, or Ethernet, interface that is connected to an Ethernet LAN.

When you finish the sample configuration, the system is configured as shown in the following figure.

Figure 2. Basic configuration of a data plane



The following example shows how to apply IP addresses directly to the two data plane interfaces already discovered for the system—dp0p1p1 and dp0p1p2. These interfaces were automatically created by the system on startup, when the system detected the physical interfaces.

Each IP address is applied directly to the interface. The system automatically discovers the MAC address (hardware ID) of the network interface card (NIC) that houses the data plane interface and applies default values for a number of other options.

To configure these interfaces, perform the following steps in configuration mode.

Table 2. Configuring data plane interfaces

Step	Command
Assign an IP address to the dp0p1p1 interface.	<pre>vyatta@R1# set interfaces dataplane dp0p1p1 address 176.16.0.65/24</pre>
Assign an IP address to the dp0p1p2 interface.	<pre>vyatta@R1# set interfaces dataplane dp0p1p2 address 10.10.30.65/24 [edit]</pre>
Commit and view the configuration.	<pre>vyatta@R1# commit OK [edit] vyatta@R1# show interfaces dataplane dataplane dp0p1p1 { address 10.1.32.73/24 mtu 1500 }</pre>

Ethernet multinetting

Each physical interface can have multiple IP addresses assigned to it. If you want to have multiple networks on the same physical interface (called multinetting) but do not want to use 802.1Q VLANs, simply create multiple address configuration nodes directly under the primary interface.

To configure Ethernet multinetting, perform the following steps in configuration mode.

Table 3. Configuring Ethernet multinetting

Step	Command
Assign the first IP address to the dp0p1p1 interface.	<pre>vyatta@R2# set interfaces dataplane dp0p1p1 address 172.16.0.65/24</pre>
Assign the second IP address to the dp0p1p1 interface.	<pre>vyatta@R1# set interfaces dataplane dp0p1p1 address 192.168.1.17/24</pre>
Commit and view the configuration.	<pre>vyatta@R1# commit OK [edit] vyatta@R1# show interfaces dataplane dp0p1p1 address 192.168.1.81/24 address 192.168.1.17/24</pre>

IPv6 on data plane interfaces

IPv6 Support Configuration Guide provides examples of configuring IPv6 on interfaces.

Hardware binding for data plane interfaces

The router provides the option to override the names of router data plane interfaces to ensure that these names remain bound to network interface cards (NICs), which is useful in the following scenarios.

- Moving a NIC from a damaged slot to a different slot (slot-number- or PCI-address-based binding)
- Moving a NIC from one slot to another (MAC-address based binding)
- Naming data plane interfaces the same across different chassis, hypervisors, and cloud environments

The hardware-binding commands have the following syntax:

interfaces dataplane *interface-name* **hardware** *binding-keyword-and-value*

For information about hardware-binding commands, refer to [Data Plane Interfaces Commands](#).

Command dependencies

Some of the hardware-binding commands must be run with other hardware-binding commands; otherwise, commit fails.

The following examples show the commands that you can run individually and those that you must run with other commands.

- The following command binds dp0s5 to the NIC at the 0:00:05.0 PCI address.

```
set interfaces dataplane dp0s5 hardware pci-address 0:00:05.0
```

- The following command binds dp0s7 to port 0 on a system that has a single NIC.

```
set interfaces dataplane dp0s7 hardware port 0
```

- The following command binds dp0s9 to the NIC whose MAC address is b0:03:f4:02:02:05.

```
set interfaces dataplane dp0s9 hardware mac b0:03:f4:02:02:05
```

- The following command binds dp0s10 to the onboard NIC whose firmware index is 39.

```
set interfaces dataplane dp0s10 hardware firmware-index 39
```

- The following commands bind dp0s6 to function 0 of the NIC in PCI slot 6.

```
set interfaces dataplane dp0s6 hardware pci-slot 6
```

```
set interfaces dataplane dp0s6 hardware pci-function 0
```

- The following commands bind dp0s8 to port 0 of the NIC in PCI slot 8.

```
set interfaces dataplane dp0s8 hardware pci-slot 8
```

```
set interfaces dataplane dp0s8 hardware dev-port 0
```

Overriding interface names based on hardware bindings


Consider a scenario in which interfaces map to PCI slots as shown in the following table.

Table 4. Mapping of interface names to PCI slots

Interface name	PCI slot
dp0s4	4
dp0s5	5
dp0s6	6

To override the interface names, you must configure hardware binding by using the **set interfaces dataplane *interface-name* hardware pci-slot *slot-number*** command, as shown in the following example.

Table 5. Overriding interface names

Step	Command
Create new placeholder interface names.  Note: Do not move your management interface if you are on a Telnet or SSH console.	<pre>vyatta@R1# set interfaces dataplane dp0s1 hardware pci-slot 4 [edit] vyatta@R1# set interfaces dataplane dp0s2 hardware pci-slot 5 [edit] vyatta@R1# set interfaces dataplane dp0s3 hardware pci-slot 6 [edit]</pre>
Commit your changes.	<pre>vyatta@R1# commit [edit]</pre>
Save your changes.	<pre>vyatta@R1# save [edit]</pre>
Exit the configuration mode.	<pre>vyatta@R1# exit logout</pre>
Reboot the router.	<pre>vyatta@R1:~\$ reboot</pre>
Verify that you have correctly configured the override of the interface names.	<pre>vyatta@R1:~\$ show interfaces dataplane</pre>
Apply the configuration of dp0s4, dp0s5 and dp0s6 to dp0s1, dp0s2 and dp0s3, respectively.	<pre>vyatta@R1:~\$ configure ...</pre>
Delete the configuration for the dp0s4, dp0s5, and dp0s6 interfaces.	<pre>vyatta@R1# delete interface dataplane dp0s4 [edit] vyatta@R1# delete interface dataplane dp0s5 [edit] vyatta@R1# delete interface dataplane dp0s6 [edit]</pre>
Commit your changes.	<pre>vyatta@R1# commit [edit]</pre>
Save your changes.	<pre>vyatta@R1# save [edit]</pre>
Exit the configuration mode.	<pre>vyatta@R1# exit logout</pre>

Binding data plane interfaces to MAC addresses

Consider a scenario in which you are running a router in a virtual machine (VM) and the virtual interfaces might not consistently appear at the same locations on each instantiation of the VM.

On the first boot of the VM, the interface assignments might be as follows:

Table 6. Data plane interface assignments and MAC addresses

Data plane interface	MAC address
dp0s3 (physically in slot 3 on the VM)	00:01:02:03:05:06
dp0s4 (physically in slot 4 on the VM)	00:01:02:03:05:07

To keep the NICs in slots 3 and 4 assigned to dp0s3 and dp0s4, respectively, for every subsequent instantiation of your VM, enter the following commands:

```
vyatta@R1# set interfaces dataplane dp0s3 hardware mac 00:01:02:03:05:06
vyatta@R1# set interfaces dataplane dp0s4 hardware mac 00:01:02:03:05:07
vyatta@R1# commit
vyatta@R1# save
```

You do not have to reboot the router.

Chapter 7. Data Plane Interfaces Commands

clear interfaces dataplane counters

Clears statistics counters for a single data plane interface or all data plane interfaces.

```
clear interfaces dataplane [ interface-name ] counters
```

Statistics counters for all data plane interfaces are cleared.

interface-name

The identifier of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Operational mode

Use this command to clear statistics counters for a single data plane interface or all data plane interfaces.

Note:

Counters in the controller (slow path) are not cleared by this command.

interfaces dataplane

Defines a data plane interface.

```
set interfaces dataplane interface-name
```

```
delete interfaces dataplane interface-name
```

```
show interfaces dataplane interface-name
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {  
    dataplane interface-name  
}
```

Use this command to create a data plane interface.

Use the `set` form of this command to create a data plane interface.

Use the `delete` form of this command to delete a data plane interface. The system creates an empty configuration node for the interface the next time the system starts.

Use the `show` form of this command to display data plane interface configuration.

interfaces dataplane address

Specifies the IP address and network prefix for a data plane interface.

```
set interfaces dataplane interface-name address { ipv4 | ipv6 | dhcp | dhcpv6 }
```

```
delete interfaces dataplane interface-name address [ ipv4 | ipv6 | dhcp | dhcpv6 ]
```

```
show interfaces dataplane interface-name address [ ipv4 | ipv6 | dhcp | dhcpv6 ]
```

interface-name

The name of a data plane interface.

ipv4

The IPv4 address of a data plane interface. The format is *ip-address/prefix* (for example, 192.168.1.77/24).

ipv6

The IPv6 address of a data plane interface. The format is *ipv6-address/prefix* (for example, 2001:db8:1234::/48).

dhcp

Defines the interface as a DHCP client, which obtains its address and prefix from a DHCP server.

dhcpv6

Defines the interface as a DHCPv6 client, which obtains its address and prefix from a DHCPv6 server.

dhcp

Defines the interface as a Dynamic Host Configuration Protocol (DHCP) client, which obtains its address and prefix from a DHCP server.

dhcpv6

Defines the interface as a Dynamic Host Configuration Protocol for IPv6 (DHCPv6) client, which obtains its address, prefix, and parameters from a DHCPv6 server.

Configuration mode

```
interfaces {
  dataplane interface-name {
    address {
      ipv4
      ipv6
    }
  }
}
```

```

    dhcp
    dhcpv6
  }
}

```

Use this command to specify the IP address and network prefix for a data plane interface.

If set to **dhcp**, the MTU value for the interface will be set through DHCP unless it is explicitly defined using [interfaces dataplane mtu](#) which takes precedence.

Use the `set` form of this command to set an IP address and a network prefix. You can set only one IP address for the interface.

Use the `delete` form of this command to delete an IP address and a network prefix.

interfaces dataplane bond-group

Adds a data plane interface to a bonding group.

```
set interfaces dataplane interface-name bond-group bondx
```

```
delete interfaces dataplane interface-name bond-group bondx
```

```
show interfaces dataplane interface-name bond-group bondx
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

bondx

The identifier for the bond group. Supported values are **bond0** through **bond99**.

Configuration mode


```

interfaces {
  dataplane interface-name {
    bond-group bondx
  }
}

```

Use this command to add a data plane interface to a data plane link bond group.

A data plane interface can only be a member of one data plane link bond group and the bond group must first be defined using [interfaces dataplane bond-group](#). The maximum number of data plane interfaces that can be added to a bonding group depends on available system resources. For most implementations this is essentially unlimited.

 **Note:** The data plane interface will not be added to the bond group if it is disabled.

You must not configure any IP address for the data plane interface if it is to become part of a bonding group. Instead, the IP address for the group is configured on the bonding interface using [interfaces dataplane address](#).

Use the `set` form of this command to add a data plane interface to a data plane link bond group.

Use the `delete` form of this command to remove a data plane interface from a data plane link bond group.

Use the `show` form of this command to view bond group configuration.

interfaces dataplane description

Describes a data plane interface.

```
set interfaces dataplane interface-name description description
```

```
delete interfaces dataplane interface-name description
```

```
show interfaces dataplane interface-name description
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

description

A mnemonic name or description for the data plane interface.

Configuration mode

```
interfaces {
  dataplane interface-name {
    description description
  }
}
```

Use this command to describe a data plane interface.

Use the `set` form of this command to describe a data plane interface.

Use the `delete` form of this command to delete the description of a data plane interface.

Use the `show` form of this command to display the description of a data plane interface.

interfaces dataplane dhcp-options no-rfc3442

Disables support for the classless static route option for DHCP on a data plane interface.

```
set interfaces dataplane interface-name dhcp-options no-rfc3442
```

```
delete interfaces dataplane interface-name dhcp-options no-rfc3442
show interfaces dataplane interface-name dhcp-options
```

The classless static route option for DHCP is enabled.

interface-name

The router can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree and configured with an IP address.

The router automatically creates configuration nodes for all available physical interfaces on startup.


If you want to use an interface with a specific function (say, BGP) the interface must be enabled within the configuration node for that function (for example, within the BGP configuration node).


no-rfc3442

Removes the classless static route option (121) from the parameter request list that a DHCP client sends to the DHCP server. For further information, refer to RFC 3442 at <https://tools.ietf.org/html/rfc3442>.

Configuration mode

```
interfaces {
  dataplane interface-name {
    dhcp-options {
      no-rfc3442
    }
  }
}
```

 **Note:** This command is relevant only if the **dhcp** option has been set by using [interfaces dataplane address](#).

 **Note:** Normally, this command is not required. It would be used only if the remote DHCP server is configured to provide classless static routes, but these routes are not required on the router that is configured to use the DHCP address.

Use the `set` form of this command to disable support for the DHCP classless static route option on a data plane interface.

Use the `delete` form of this command to re-enable support for the DHCP classless static route option on a data plane interface.

Use the `show` form of this command to display the status of the DHCP classless static route option on a data plane interface.

interfaces dataplane dhcpv6-options

Specifies the way in which a DHCPv6 client is to acquire an address, parameters, or both from the DHCPv6 server.

```
set interfaces dataplane interface-name dhcpv6-options [ parameters-only |
temporary ]
```

```
delete interfaces dataplane interface-name dhcpv6-options [ parameters-only |
temporary ]
```

```
show interfaces dataplane interface-name dhcpv6-options
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

parameters-only

Acquires only configuration parameters (and not an IPv6 address) from the DHCPv6 server.

temporary

Acquires a temporary IPv6 address from the DHCPv6 server.

Configuration mode

```
interfaces {
  dataplane interface-name {
    dhcpv6-options {
      parameters-only
      temporary
    }
  }
}
```

Use this command to specify in what way the DHCPv6 client is to acquire an IPv6 address, parameters, or both from the DHCPv6 server.

Note that the parameters are relevant only if the **dhcpv6** option has been set for [interfaces dataplane address](#).

The **parameters-only** parameter is typically used with Stateless Address Autoconfiguration (SLAAC) or static address configuration. The **parameters-only** and **temporary** parameters are mutually exclusive.

Use the `set` form of this command to specify the DHCPv6 options.

Use the `delete` form of this command to delete the DHCPv6 options.

Use the `show` form of this command to display DHCPv6 option configuration.

interfaces dataplane disable

Disables a data plane interface without discarding its configuration.

```
set interfaces dataplane interface-name disable
```

```
delete interfaces dataplane interface-name disable
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane interface-name {
    disable
  }
}
```

Use this command to disable a data plane interface without discarding its configuration.

Use the `set` form of this command to disable an interface.

Use the `delete` form of this command to enable an interface.

interfaces dataplane disable-link-detect

Directs a data plane interface not to detect physical link-state changes.

```
set interfaces dataplane interface-name disable-link-detect
```

```
delete interfaces dataplane interface-name disable-link-detect
```

An interface detects physical link-state changes.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane interface-name {
    disable-link-detect
  }
}
```


Use this command to direct a data plane interface not to detect physical state changes to the Ethernet link (for example, when a cable is unplugged).

Use the `set` form of this command to disable detection of physical link-state changes.

Use the `delete` form of this command to enable detection of physical link-state changes.

interfaces dataplane hardware dev-port

Binds a data plane interface to a port on a PCI NIC that has multiple ports.

```
set interfaces dataplane interface-name hardware dev-port port-number
```

```
delete interfaces dataplane interface-name hardware dev-port port-number
```

```
show interfaces dataplane interface-name hardware dev-port
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

port-number

The number of a port. The number ranges from 0 through 4294967295.

Configuration mode

```
dataplane interface-name {
    hardware {
        dev-port port-number
    }
}
```

This command is useful when a NIC has multiple ports but occupies a single PCI slot. To uniquely identify a port on the NIC, you must also use the **interfaces dataplane *interface-name* hardware pci-address *pci-address*** or **interfaces dataplane *interface-name* hardware pci-slot *pci-slot*** command. For example, you can instruct the router to bind dp0s5 to port 1 of the NIC whose address is 00:19:0 or bind dp0s3 to port 2 of the NIC in slot 3.

Use the `set` form of this command to bind a data plane interface to a port on a PCI NIC.

Use the `delete` form of this command to unbind a data plane interface from a port on a PCI NIC.

Use the `show` form of this command to display the port hardware-binding information for a data plane interface.

interfaces dataplane hardware firmware-index

Binds a data plane interface to the firmware index of an onboard NIC.

```
set interfaces dataplane interface-name hardware firmware-index firmware-index
```

```
delete interfaces dataplane interface-name hardware firmware-index firmware-index
```

```
show interfaces dataplane interface-name hardware firmware-index
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

firmware-index

The ACPI firmware index of an onboard NIC. The index is a number that ranges from 0 through 4294967295. The BIOS assigns the index to the onboard NIC, which is built into the motherboard of a system. Only an OEM can change firmware index assignments.

Configuration mode

```
dataplane interface-name {
    hardware {
        firmware-index firmware-index
    }
}
```

Use the `set` form of this command to bind a data plane interface to the firmware index of an onboard NIC.

Use the `delete` form of this command to unbind a data plane interface from the firmware index of an onboard NIC.

Use the `show` form of this command to display the firmware index of an onboard NIC that is bound to a data plane interface.

interfaces dataplane hardware mac

Binds a data plane interface to the vendor-assigned MAC address of a NIC.

```
set interfaces dataplane interface-name hardware mac mac-address
```

```
delete interfaces dataplane interface-name hardware mac mac-address
```

```
show interfaces dataplane interface-name hardware mac
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

mac-address

The MAC address of a NIC. The MAC address has the following format: *h:h:h:h:h:h*, where *h* is one or two hexadecimal digits.

Configuration mode

```
dataplane interface-name {
    hardware {
        mac mac-address
    }
}
```

Use the `set` form of this command to bind a data plane interface to the MAC address of a NIC.

Use the `delete` form of this command to unbind the data plane interface from the MAC address of a NIC.

Use the `show` form of this command to display the MAC address hardware-binding information for a data plane interface.

interfaces dataplane hardware pci-address

Binds a data plane interface to the PCI address of a NIC.

```
set interfaces dataplane interface-name hardware pci-address pci-address
```

```
delete interfaces dataplane interface-name hardware pci-address pci-address
```

```
show interfaces dataplane interface-name hardware pci-address
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

pci-address

The PCI address of a NIC. The format of the address is as follows:

[domain:]bus:slot.function

- *domain*: The domain is a hexadecimal value that ranges from 0 through ffff.
- *bus*: The bus address is a hexadecimal value that ranges from 0 through ff.
- *slot*: The slot number is a hexadecimal value that ranges from 0 through 1f.
- *function*: The function number ranges from 0 through 7.

Configuration mode

```
dataplane interface-name {
```

```

    hardware {
        pci-address pci-address
    }
}

```

You can obtain the PCI address of a NIC by running the **lspci** Linux command. In the following example, the bus address of the first card is 0, slot number is 19, and function number is 0. The bus address of the second card is 2, slot number is 0, and function number is 0. If you have a single PCI domain, as most systems do, then the domain value is 0.

```

$ lspci
...
00:19.0 Ethernet controller: Intel Corporation Ethernet Connection I218-LM
  (rev 04)
...
02:00.0 Network controller: Intel Corporation Wireless 7260 (rev 73)
...

```

Use the `set` form of this command to bind a data plane interface to the PCI address of a NIC.

Use the `delete` form of this command to unbind a data plane interface from a PCI address.

Use the `show` form of this command to display the PCI address that is bound to a data plane interface.

interfaces dataplane hardware pci-function

Binds a data plane interface to a function number for a PCI NIC.

```
set interfaces dataplane interface-name hardware pci-function function-number
```

```
delete interfaces dataplane interface-name hardware pci-function function-number
```

```
show interfaces dataplane interface-name hardware pci-function
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

function-number

The number of a PCI function (a logical PCI card). The number ranges from 0 through 7.

Configuration mode

```

dataplane interface-name {
    hardware {
        pci-function function-number
    }
}

```

```
}
}
```

If a PCI card hosts several functions, use this command to associate a data plane interface with one of those functions. You must use this command with the **interfaces dataplane interface-name hardware pci-slot slot-number** command.

Use the `set` form of this command to bind a data plane interface to a function number for a PCI NIC.

Use the `delete` form of this command to unbind a data plane interface from a PCI function number.

Use the `show` form of this command to display the hardware PCI function that is bound to a data plane interface.

interfaces dataplane hardware pci-slot

Binds a data plane interface to a PCI slot into which a NIC is plugged.

```
set interfaces dataplane interface-name hardware pci-slot pci-slot-number
```

```
delete interfaces dataplane interface-name hardware pci-slot pci-slot-number
```

```
show interfaces dataplane interface-name hardware pci-slot
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

pci-slot-number

The number of a PCI slot. The number ranges from 0 through 4294967295.

Configuration mode

```
dataplane interface-name {
    hardware {
        pci-slot pci-slot-number
    }
}
```

Use the `set` form of this command to bind a data plane interface to a PCI slot.

Use the `delete` form of this command to unbind a data plane interface from a PCI slot.

Use the `show` form of this command to display the PCI slot hardware-binding information for a data plane interface.

interfaces dataplane hardware port

Binds a data plane interface to a port on a PCI NIC, based on the discovery order by DPDK.

```
set interfaces dataplane interface-name hardware port port-number
```

```
delete interfaces dataplane interface-name hardware port port-number
```

```
show interfaces dataplane interface-name hardware port
```

interface-name

The name of a data plane interface. For more information about formats for the names of supported data plane interfaces, refer to [Data Plane Interface Names](#).

port-number

The discovery order by DPDK. The discovery order is a number that ranges from 0 through 31.

Configuration mode

```
dataplane interface-name {
    hardware {
        port port-number
    }
}
```

Data Plane Development Kit (DPDK) discovers interfaces on a system in some particular, nonrandom order. For example, if a system has one NIC, you can use this command to bind dp0s0 to port 0, the first port that DPDK finds. By using this command, you can always be certain that no matter where the NIC is added to a system, the NIC always comes up as dp0s0.

Use the `set` form of this command to bind a data plane interface to a port on a PCI NIC.

Use the `delete` form of this command to unbind a data plane interface from a PCI NIC port.

Use the `show` form of this command to display the port hardware-binding information for a data plane interface.

interfaces dataplane ip disable-forwarding

Disables forwarding on a data plane interface.

```
set interfaces dataplane interface-name ip disable-forwarding
```

```
delete interfaces dataplane interface-name ip disable-forwarding
```

IPv6 packets are forwarded.

interface-name

The identifier of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane interface-name {
    ip {
      disable-forwarding
    }
  }
}
```

Use this command to disable packet forwarding on a data plane interface.

You can also disable forwarding globally (that is, for all interfaces) by using the `system ipv6 disable-forwarding` command.

Use the `set` form of this command to disable IPv6 packet forwarding on an interface.

Use the `delete` form of this command to enable IPv6 packet forwarding on an interface.

interfaces dataplane ip enable-proxy-arp

Enables proxy ARP on a data plane interface.

```
set interfaces dataplane interface-name ip enable-proxy-arp
```

```
delete interfaces dataplane interface-name ip enable-proxy-arp
```

Proxy ARP is not enabled on a data plane interface.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane interface-name {
    ip {
      enable-proxy-arp
    }
  }
}
```

Use this command to enable proxy Address Resolution Protocol (ARP) on a data plane interface.

Proxy ARP allows a data plane interface to respond with its own media access control (MAC) address to ARP requests for destination IP addresses on subnets that are attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Use the `set` form of this command to enable proxy ARP on an interface.

Use the `delete` form of this command to return the system to its default behavior, that is, proxy ARP is not enabled on a data plane interface.

interfaces dataplane ip pim mode

Specifies the PIM on a data plane interface.

```
set interfaces dataplane interface-name ip pim mode mode
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

mode

The PIM mode for an interface. The mode is as follows:

dense: Enable PIM dense mode.

dense-passive: Enable passive operation for PIM dense mode.

sparse: Enable PIM sparse mode.

sparse-passive: Enable passive operation for PIM sparse mode.

Configuration mode

```
interfaces {
  dataplane interface-name {
    ip {
      pim {
        mode sparse
      }
    }
  }
}
```

Use the `set` form of this command to specify the PIM mode on a data plane interface.

interfaces dataplane ip rpf-check

Enables the policy for reverse-path filter.

```
set interfaces dataplane interface-name ip rpf-check { strict | loose | disable }
```



```
delete interfaces dataplane interface_name ip rpf-check [ strict | loose | disable ]
```

```
show interfaces dataplane interface_name ip rpf-check
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

strict

An ingress filter using a dynamic access list. According to RFC 3704.

loose

An ingress filter that checks for the presence of a source route. According to RFC 3704.

disable

No source validation.

Configuration mode

```
interfaces {
  dataplane interface-name {
    ip {
      rpf-check
      strict
      loose
      disable
    }
  }
}
```

Use this command to enable the policy for reverse-path filter.

Use the `set` form of this command to enable the policy for reverse-path filter.

Use the `delete` form of this command to delete the policy for reverse-path filter.

Use the `show` form of this command to display the policy for reverse-path filter.

interfaces dataplane ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for a data plane interface.

```
set interfaces dataplane interface-name ip tcp-mss limit clamp-value
```

```
delete interfaces dataplane interface-name ip tcp-mss limit clamp-value
```

```
show interfaces dataplane interface-name ip tcp-mss limit
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  dataplane {
    ip {
      tcp-mss {
        limit
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane ip tcp-mss mtu

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU for a data plane interface.

```
set interfaces dataplane interface-name ip tcp-mss mtu
```

```
delete interfaces dataplane interface-name ip tcp-mss mtu
```

```
show interfaces dataplane interface-name ip tcp-mss mtu
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane {
    ip {
      tcp-mss {
        mtu
      }
    }
  }
}
```

```
}
```

Use the `set` form of this command to clamp the MSS to the MTU for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane ip tcp-mss mtu-minus

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU less a given value for a data plane interface.

```
set interfaces dataplane interface-name ip tcp-mss mtu-minus subtrahend
```

```
delete interfaces dataplane interface-name ip tcp-mss mtu-minus subtrahend
```

```
show interfaces dataplane interface-name ip tcp-mss mtu-minus
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```
interfaces {
  dataplane {
    ip {
      tcp-mss {
        mtu-minus
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane ipv6 address

Assigns an IPv6 address to a data plane interface.

```

set interfaces dataplane interface-name ipv6 address [ autoconf | eui64
ipv6prefix | link-local ipv6address ]

delete interfaces dataplane interface-name ipv6 address [ autoconf | eui64
ipv6prefix | link-local ipv6address ]

show interfaces dataplane interface-name ipv6 address [ autoconf | eui64 ]

```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

autoconf

Generates an IPv6 address using the SLAAC protocol. Use this keyword if the interface is performing a “host” function rather than a “router” function. You can specify this keyword in addition to static IPv6, static IPv4, or IPv4 DHCP addresses on the interface.

eui64 *ipv6prefix*

Specifies the 64-bit IPv6 address prefix that is used to configure an IPv6 address in EUI-64 format. The system concatenates this prefix with a 64-bit EUI-64 value that is derived from the 48-bit MAC address of the interface.

link-local *ipv6address*

Specifies the IPv6 address that is valid only for communication within the local broadcast domain.

Configuration mode

```

interfaces {
  dataplane interface-name {
    ipv6 {
      address {
        autoconf
        eui64 ipv6prefix
        link-local ipv6address
      }
    }
  }
}

```

Use this command to assign an IPv6 address to a data plane interface.

Use the **autoconf** keyword to direct the system to automatically configure (autoconfigure) the address, using the Stateless Address Autoconfiguration (SLAAC) protocol defined in RFC 4862. Alternatively, you can use the **eui64** keyword with an IPv6 address prefix so that the system constructs the IPv6 address.

If you want the system to use SLAAC to acquire an address on the interface, then in addition to setting this parameter, you must also disable IPv6 forwarding, either globally (by using the `system ipv6 disable-forwarding` command) or specifically on the interface (by using [interfaces dataplane ipv6 disable-forwarding](#)).

Use the **link-local** keyword to explicitly configure a specific link-local address, which then replaces any system generated link-local address.

Use the `set` form of this command to assign an IPv6 address to a data plane interface.

Use the `delete` form of this command to delete an IPv6 address from a data plane interface.

Use the `show` form of this command to display IPv6 address configuration settings.

interfaces dataplane ipv6 disable

Disables IPv6 on a data plane interface.

```
set interfaces dataplane interface-name ipv6 disable
```

```
delete interfaces dataplane interface-name ipv6 disable
```

```
show interfaces dataplane interface-name ipv6 disable
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode.

```
interfaces {
  dataplane interface-name {
    ipv6 {
      disable
    }
  }
}
```

By default, IPv6 is enabled on all interfaces. A global command exists which can disable IPv6, namely `set system ipv6 disable`, and this will take precedence over any of the existing per-interface based, IPv6 commands.

IPv6 Forwarding can be disabled via the `set interface dataplane interface-name ipv6 disable-forwarding` command, but note that IPv6 traffic can still be terminated on this interface.

IPv6 configuration can be totally disabled via the `set interface dataplane interface-name ipv6 disable` command.

Use the `set` form of this command to disable IPv6 on this interface.

Use the `delete` form of this command to enable IPv6 on this interface.

Use the `show` form of this command to display the current IPv6 configuration on this interface.

interfaces dataplane ipv6 disable-forwarding

Disables IPv6 forwarding on a data plane interface.

```
set interfaces dataplane interface-name ipv6 disable-forwarding
```

```
delete interfaces dataplane interface-name ipv6 disable-forwarding
```

IPv6 packets are forwarded.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```
interfaces {
  dataplane interface-name {
    ipv6 {
      disable-forwarding
    }
  }
}
```

Use this command to disable IPv6 packet forwarding on a data plane interface.

You can also disable IPv6 forwarding globally (that is, for all interfaces) by using the `system ipv6 disable-forwarding` command.

Use the `set` form of this command to disable IPv6 packet forwarding on an interface.

Use the `delete` form of this command to enable IPv6 packet forwarding on an interface.

interfaces dataplane ipv6 dup-addr-detect-transmits

Specifies the number of NS packets to transmit as part of the DAD process.

```
set interfaces dataplane interface-name ipv6 dup-addr-detect-transmits num
```

```
delete interfaces dataplane interface-name ipv6 dup-addr-detect-transmits
```

One NS packet is transmitted as part of the DAD process.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

num

The number of NS packets to transmit as part of the DAD process. The number ranges from 1 through *n*. The default number is 1.

Configuration mode

```

interfaces {
  dataplane interface-name {
    ipv6 {
      dup-addr-detect-transmits num
    }
  }
}

```

Use this command to specify the number of Neighbor Solicitation (NS) packets to transmit as part of the Duplicate Address Detection (DAD) process.

Use the `set` form of this command to specify the number of NS packets to transmit on an interface.

Use the `delete` form of this command to delete the transmission number from an interface and transmit the default number of one NS packet.

interfaces dataplane ipv6 router-advert

Specifies the router advertisements to be sent from a data plane interface.

```

set interfaces dataplane interface-name ipv6 router-advert [ cur-hop-limit
limit ] [ default-lifetime lifetime ] [ default-preference preference ] [
link-mtu mtu ] [ managed-flag state ] [ max-interval interval ] [ min-interval
interval ] [ other-config-flag state ] [ prefix ipv6net [ autonomous-flag
state | on-link-flag state | preferred-lifetime lifetime | valid-lifetime
lifetime ] ] [ reachable-time time ] [ retrans-timer time ] [ send-advert
state ]

```

```

delete interfaces dataplane interface-name ipv6 router-advert [ cur-hop-limit
] [ default-lifetime ] [ default-preference ] [ link-mtu ] [ managed-flag ]
[ max-interval ] [ min-interval ] [ other-config-flag ] [ prefix ipv6net [
autonomous-flag | on-link-flag | preferred-lifetime | valid-lifetime ] ] [
reachable-time ] [ retrans-timer ] [ send-advert ]

```

Router advertisements are not sent on an interface.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

cur-hop-limit *limit*

Specifies the hop-count field of the IP header for outgoing (unicast) IP packets. This value is placed in the Hop Count field of the IP header for outgoing (unicast) IP packets. The range is 0 to 255. The default is 64. A limit of 0 means unspecified by the router.

default-lifetime *lifetime*

Specifies the lifetime, in seconds, that is associated with the default router. A lifetime of 0 indicates that the router is not a default router. The lifetime ranges from the value that is configured for the **max-interval** option to 9000 (18.2 hours). If the lifetime is not configured, the value for this timer is three times **max-interval**.

default-preference *preference*

Specifies the preference that is associated with the default router. The preference is **low**, **medium**, or **high**. The default preference is **medium**.

link-mtu *mtu*

Specifies the MTU to be advertised for the link. The MTU is 0 or ranges from 1280 through the maximum MTU for the type of link, as defined in RFC 2464. The default MTU is 0, which means the MTU is not specified in the router advertisement message. That is because it is expected that the MTU is configured directly on the interface itself and not for routing advertisements. You can configure this option when the link MTU is not well known.

If the MTU that is set here does not match the MTU that is configured on the interface, the system issues a warning but does not fail.

managed-flag *state*

Whether to use the administered protocol for address autoconfiguration. The state is either of the following:

true: Hosts use the administered (stateful) protocol for address autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration.

false: Hosts use only stateless address autoconfiguration.

The default state is **false**.

max-interval *interval*

Specifies the maximum time, in seconds, that is allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 4 through 1800. The default is 600 (10 minutes).

min-interval *interval*

Specifies the minimum time, in seconds, that is allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 3 through 0.75 times the **max-interval** option. The default interval is 0.33 times **max-interval**.

other-config-flag *state*

Specifies that the interface use the administered (stateful) protocol for autoconfiguration of nonaddress information, as defined in RFC 4862. The state is either of the following:

true: Hosts use the administered protocol for autoconfiguration of nonaddress information.

false: Hosts use stateless autoconfiguration of nonaddress information.

The default state is **false**.

prefix *ipv6net*

Multinode. Specifies the IPv6 prefix to be advertised on the IPv6 interface, in the format *ipv6-address/prefix*.

You can define more than one IPv6 prefix by configuring multiple prefix configuration nodes.

autonomous-flag *state*

Specifies whether the prefix can be used for autonomous address configuration as defined in RFC 4862. The state is either of the following:

true: The prefix can be used for autonomous address configuration.

false: The prefix cannot be used for autonomous address configuration.

The default state is **true**.

on-link-flag state

Specifies whether the prefix can be used for onlink determination, as defined in RFC 4862. The state is either of the following:

true: The prefix can be used for onlink determination.

false: The advertisement makes no statement about onlink or off-link properties of the prefix. For instance, the prefix might be used for address configuration with some addresses belonging to the prefix being onlink and others being off-link.

The default state is **true**.

preferred-lifetime lifetime

Specifies the length of time, in seconds, that the addresses generated from the prefix by Stateless Address Autoconfiguration (SLAAC) is to remain preferred, as defined in RFC 4862. The interval is with respect to the time the packet is sent. The lifetime ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0xFFFFFFFF.) The default lifetime is 604800 (7 days).

valid-lifetime lifetime

Specifies the length of time, in seconds, that the prefix is valid for onlink determination, as defined in RFC 4862. The interval is with respect to the time the packet is sent. The lifetime ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0xFFFFFFFF.) The default lifetime is 2592000 (30 days).

reachable-time time

Specifies the length of time, in milliseconds, for which the system assumes a neighbor is reachable after having received a reachability confirmation. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Section 7.3 of RFC 2461). The time ranges from 0 through 3600000, where a value of 0 means the reachable time is not specified in the router advertisement message. The default time is 0.

retrans-timer time

Specifies the length of time, in milliseconds, between retransmitted NS messages. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Sections 7.2 and 7.3 of RFC 2461). The time ranges from 0 through 4294967295, where a value of 0 means the retransmit time is not specified in the router advertisement message. The default time is 0.

send-advert state

Specifies whether router advertisements are to be sent from this interface. The state is either of the following:

true: Sends router advertisements from this interface.

false: Does not send router advertisements from this interface. If a state is in effect, parameters in this configuration subtree are still used to configure the local implementation parameters.

The default state is **true**.

Configuration mode

```

interfaces {
  dataplane interface-name {
    ipv6 {
      router-advert {
        cur-hop-limit limit
        default-lifetime lifetime
        default-preference preference
        link-mtu mtu
        managed-flag state
        max-interval interval
        min-interval interval
        other-config-flag state
        prefix ipv6net {
          autonomous-flag state
          on-link-flag state
          preferred-lifetime lifetime
          valid-lifetime lifetime
        }
        reachable-time time
        retrans-timer time
        send-advert state
      }
    }
  }
}

```

Use this command to specify the router advertisements to be sent from a data plane interface.

Router advertisements are sent by IPv6 routers to advertise their existence to hosts on the network. IPv6 hosts do not send router advertisements.

If the **router-advert** node of the configuration tree is missing, router advertisements are not sent. In addition, if IPv6 forwarding is disabled either globally (by using the `system ipv6 disable-forwarding` command) or on the interface (by using [interfaces dataplane ipv6 disable-forwarding](#)), router advertisements are not sent.

Most router advertisement parameters are required by either the Neighbor Discovery (ND) protocol or the Stateless Address Autoconfiguration (SLAAC) protocol. These parameters are used both locally for the IPv6 implementation and become part of the RA messages sent to hosts on the network so that they can be configured appropriately.

Use the `set` form of this command to create the router-advert configuration node and begin to send router advertisements.

Use the `delete` form of this command to delete the router-advert configuration node and stop sending router advertisements.

Use the `show` form of this command to display router advertisement configuration.

interfaces dataplane ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for a data plane interface.

```
set interfaces dataplane interface-name ipv6 tcp-mss limit clamp-value
delete interfaces dataplane interface-name ipv6 tcp-mss limit clamp-value
show interfaces dataplane interface-name ipv6 tcp-mss limit
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  dataplane {
    ipv6 {
      tcp-mss {
        limit
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane ipv6 tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for a data plane interface.

```
set interfaces dataplane interface-name ipv6 tcp-mss mtu
delete interfaces dataplane interface-name ipv6 tcp-mss mtu
show interfaces dataplane interface-name ipv6 tcp-mss mtu
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Configuration mode

```

interfaces {
  dataplane {
    ipv6 {
      tcp-mss {
        mtu
      }
    }
  }
}

```

Use the `set` form of this command to clamp the MSS to the MTU for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for a data plane interface.

```
set interfaces dataplane interface-name ipv6 tcp-mss mtu-minus subtrahend
```

```
delete interfaces dataplane interface-name ipv6 tcp-mss mtu-minus subtrahend
```

```
show interfaces dataplane interface-name ipv6 tcp-mss mtu-minus
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  dataplane {
    ipv6 {
      tcp-mss {
        mtu-minus
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane mac

Sets the MAC address of a data plane interface.

```
set interfaces dataplane interface-name mac mac-addr
```

```
delete interfaces dataplane interface-name mac
```

The default MAC address for an interface is the factory-set MAC address.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

mac-addr

A MAC address. The format of the address is six 8-bit numbers, separated by colons, in hexadecimal; for example, 00:0a:59:9a:f2:ba.

Configuration mode

```
interfaces {
  dataplane interface-name {
    mac mac-addr
  }
}
```

Use this command to set the media access control (MAC) address of a data plane interface.

Some data plane interfaces provide the ability to change their MAC addresses. This command allows you to change the MAC addresses of these interfaces.

Use the `set` form of this command to set the MAC address of an interface.

Use the `delete` form of this command to delete the MAC address of an interface, restoring the factory-assigned MAC address.

interfaces dataplane mtu

Sets the size of the maximum transmission unit (MTU) for a data plane interface.

During forwarding, IPv4 packets larger than the MTU are fragmented unless the “Don't Fragment” (DF) bit is set in the IP header. In that case, the packets are dropped and an ICMP “fragmentation needed” message is returned to the sender.

Note that MTU sizes larger than 1500 cause the system to generate “jumbo frames,” which are not compatible with some data plane interface cards and devices.

If the data plane interface is part of an Ethernet link bond, MTU settings for the bonded link override MTU settings for the physical interface.

```
set interfaces dataplane interface-name mtu mtu
delete interfaces dataplane interface-name mtu
show interfaces dataplane interface-name mtu
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

mtu

The size of the MTU, in octets, for the interface as a whole, including any logical interfaces configured for it. The size of the MTU ranges from 68 through 9000.

The default size of the MTU is 1500.

Configuration mode

```
interfaces {
  dataplane interface-name {
    mtu mtu
  }
}
```

Use the `set` form of this command to specify the size of the MTU.

Use the `delete` form of this command to delete the size of the MTU and restore the default size of 1500.

Use the `show` form of this command to display the current size of the MTU.

interfaces dataplane sflow

Enables sflow on the specified interface.

```
set interfaces dataplane interface-name sflow
delete interfaces dataplane interface-name sflow
show interfaces dataplane interface-name sflow
```

interface-name

The router can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree.

The router automatically creates configuration nodes for all available physical interfaces on startup.

Configuration mode

```
interfaces {
  dataplane interface-name {
    sflow
  }
}
```

Use the `set` form of this command to enable sflow on the specified interface.

Use the `delete` form of this command to disable sflow on the specified interface.

Use the `show` form of this command to show whether sflow is enabled on the specified interface.

interfaces dataplane speed

Specifies the link speed for a data plane interface.

```
set interfaces dataplane interface-name speed { 1G | 2.5G | 10G | 25g | 40g |
100g | 100m | auto }
```

```
delete interfaces dataplane interface-name speed { 1G | 2.5G | 10G | 25g | 40g
| 100g | 100m | auto }
```

```
show interfaces dataplane interface-name speed { 1G | 2.5G | 10G | 25g | 40g |
100g | 100m | auto }
```

interface-name

The name of a data plane interface.

speed

The LAN speed of the data plane interface.

1g

1 Gbit/sec.

2.5g

2.5 Gbit/sec.

10g

10 Gbit/sec.

25g

25 Gbit/sec.

40g

40 Gbit/sec.

100g

100 Gbit/sec.

100m

100 Mbit/sec.

auto

Auto negotiation (default).

Configuration mode

```
interfaces {
  dataplane interface-name {
    speed
  }
}
```

Use the `set` form of this command to set the LAN speed.

Use the `delete` form of this command to delete the LAN speed.

interfaces dataplane switch-group port-parameters mode

Specifies the mode of a switch port.

```
set interfaces dataplane interface-name switch-group port-parameters mode {
access | trunk }
```

```
set interfaces dataplane interface-name switch-group port-parameters mode {
access | trunk }
```

```
show interfaces dataplane interface-name switch-group port-parameters mode {
access | trunk }
```

interface-name

The name of a data plane interface.

mode

The mode.

access

Only allow untagged VLANs on this port.

trunk

Allow multiple VLANs on this port

Configuration mode

```
interfaces {
  dataplane interface-name {
    switch-group {
      port-parameters {
```



```

}
}
}
mode
}
}
}
}

```

Use this command to specify the the mode of a switch port.

Use the `set` form of this command to set the mode.

Use the `delete` form of this command to delete the mode.

interfaces dataplane switch-group port-parameters vlan-parameters vlans

Specifies the VLANs allowed to ingress to this port.

```
set interfaces dataplane interface-name switch-group port-parameters vlan-parameters vlans vlan-ids
```

```
delete interfaces dataplane interface-name switch-group port-parameters vlan-parameters vlans vlan-ids
```

```
show interfaces dataplane interface-name switch-group port-parameters vlan-parameters vlans vlan-ids
```

interface-name

The name of a data plane interface.

vlan_ids

VLANs allowed to ingress to this port.

Configuration mode

```

interfaces {
  dataplane interface-name {
    switch-group {
      port-parameters {
        vlan-parameters {
          vlans
        }
      }
    }
  }
}

```

Use this command to specify the the VLANs allowed to ingress to this port..

Use the `set` form of this command to set the VLANs allowed to ingress to this port.

Use the `delete` form of this command to delete the VLANs allowed to ingress to this port.

interfaces dataplane switch-group switch

Assigns a switch to a port.

```
set interfaces dataplane interface-name switch-group switch text
delete interfaces dataplane interface-name switch-group switch text
show interfaces dataplane interface-name switch-group switch text
```

interface-name

The name of a data plane interface.

text

The switch of which this port is a member.

Configuration mode

```
interfaces {
  dataplane interface-name {
    switch-group {
      switch {
      }
    }
  }
}
```

Use the `set` form of this command to to assign a switch to a port.

Use the `delete` form of this command to delete a switch from a port.

Use the `show` form of this command to display a switch for a port.

interfaces dataplane vif dhcp-options no-rfc3442

Disables support for the classless static route option for DHCP on a virtual interface.

```
set interfaces dataplane interface-name vif vif-id dhcp-options no-rfc3442
delete interfaces dataplane interface-name vif vif-id dhcp-options no-rfc3442
show interfaces dataplane interface-name vif vif-id dhcp-options
```

The classless static route option for DHCP is enabled.

interface-name

The router can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree and configured with an IP address.

The router automatically creates configuration nodes for all available physical interfaces on startup.

If you want to use an interface with a specific function (say, BGP) the interface must be enabled within the configuration node for that function (for example, within the BGP configuration node).

vif-id


Multinode. The ID of a virtual interface. The ID ranges from 1 through 4094.


no-rfc3442

Removes the classless static route option (121) from the parameter request list that a DHCP client sends to the DHCP server. For further information, refer to RFC 3442 at <https://tools.ietf.org/html/rfc3442>.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      dhcp-options {
        no-rfc3442
      }
    }
  }
}
```

 **Note:** This command is relevant only if the **dhcp** option has been set by using [interfaces dataplane vif address](#).

 **Note:** Normally, this command is not required. It would be used only if the remote DHCP server is configured to provide classless static routes, but these routes are not required on the router that is configured to use the DHCP address.

Use the `set` form of this command to disable support for the DHCP classless static route option on a virtual interface.

Use the `delete` form of this command to re-enable support for the DHCP classless static route option on a virtual interface.

Use the `show` form of this command to display the status of the DHCP classless static route option on a virtual interface.

interfaces dataplane vif policy qos

Configures a QoS policy for a VLAN on a virtual interface.

```
set interfaces dataplane interface-name vif vif-id policy qos name
```

```
delete interfaces dataplane interface-name vif vif-id policy qos
```

interface-name

The router can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree.

The router automatically creates configuration nodes for all available physical interfaces on startup.

vif-id

Multinode. The ID of a virtual interface. The ID ranges from 1 through 4094.

name

Specifies a QoS policy name.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      policy {
        qos name
      }
    }
  }
}
```

Use the `set` form of this command to configure a QoS policy for a VLAN on a virtual interface. Before using this command, use the `set interfaces dataplane interface-name policy qos policy-name` command to set a port level QoS policy. The command is described in *QoS Configuration Guide*

Use the `delete` form of this command to remove the QoS policy on the virtual interface.

interfaces dataplane vrrp vrrp-group notify bgp

Sets the BGP update notification when the path taken by the outgoing traffic changes.

```
set interfaces dataplane interface-name vrrp vrrp-group vrrp-group-id notify bgp
```

```
delete interfaces dataplane interface-name vrrp
```

```
show interfaces dataplane interface-name vrrp
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vrrp-group-id

The ID of a vrrp downlink group instance.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vrrp {
      vrrp-group vrrp-group-id {
        notify {
          bgp
        }
      }
    }
  }
}

```

In certain topologies that involve VRRP in downlink LANs and eBGP to ISP in uplink LANs, symmetric routing can be achieved by influencing or changing the path taken by incoming traffic according to the path taken by outgoing traffic.

The edge topology described here can have two routers that act as VRRP Master and Backup pair for the downlink LAN of the end hosts. Both VRRP Master and Backup pair routers are connected to an ISP gateway router, each through eBGP. This configuration provides two possible outgoing and incoming paths for the traffic.

When VRRP state changes in downlink, that is, when the path taken by the outgoing traffic changes, routers send BGP updates to respective eBGP ISP gateway peers. The update notification contains Multi-Exit Discriminator (MED) or prepend-as-path changes according to the configuration to influence the change in BGP best paths for incoming traffic.

The as-path-string that is used on the active path or the path that includes the VRRP elected master is NONE, that is, no AS-path prepended.

This configuration is required to enable the symmetric routing on routers in association with either of the following commands:

- `set protocols bgp asn neighbor id vrrp-failover vrrp-group vrrp-group-id med med-value`
- `set protocols bgp asn neighbor id vrrp-failover vrrp-group vrrp-group-id prepend-as as-path-string`

monitor interfaces dataplane traffic

Displays captured traffic on a data plane interface.

```

monitor interfaces dataplane interface-name traffic [ detail [ filter filter-name | unlimited [ filter filter-name ] ] | filter filter-name | save filename | unlimited [ filter filter-name ] ]

```

Captured traffic for all ports on the specified interface is shown.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

detail

Provides detailed information about the monitored data plane traffic.

unlimited

Monitors an unlimited amount of traffic.

save filename

Saves the monitored traffic to the specified file.

filter filter-name

Applies the specific PCAP (packet capture) filter to traffic.

Operational mode

Use this command to display captured traffic on the first 1,000 packets through a data plane interface. Type Ctrl+c to stop the output.

The following example shows how to display captured traffic on the dp0p1p1 interface.

```
vyatta@vyatta:~$ monitor interfaces dataplane dp0p1p1 traffic
Capturing traffic on dp0p1p1 ...
 0.000000 fe80::ad08:8661:4d:b925 -> ff02::c          SSDP M-SEARCH * HTTP/1.1
 0.000067 fe80::69ca:5c11:bcf6:29da -> ff02::c          SSDP M-SEARCH *
HTTP/1.1
 2.608804 fe80::8941:71ef:b55d:e348 -> ff02::1:2        DHCPv6 Solicit
 3.010862 fe80::ad08:8661:4d:b925 -> ff02::c          SSDP M-SEARCH * HTTP/1.1
 3.010901 fe80::69ca:5c11:bcf6:29da -> ff02::c          SSDP M-SEARCH *
HTTP/1.1
 4.568357 192.168.1.254 -> 238.255.255.251 SSDP NOTIFY * HTTP/1.1
 4.568372 192.168.1.254 -> 238.255.255.251 SSDP NOTIFY * HTTP/1.1
...
```

show interfaces dataplane

Displays the operational status of a data plane interface or all data plane interfaces.

```
show interfaces dataplane [ interface-name ]
```

The operational status of all data plane interfaces is displayed.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Operational mode

Use this command to display the operational status of a data plane interface or all data plane interfaces.

Note:

This command only displays the counters for a specific interface. Counters in the controller (slow path) are not displayed by this command.

The following example shows how to display the operational status of all data plane interfaces.

```
vyatta@vyatta:~$ show interfaces dataplane
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface          IP Address          S/L  Description
-----          -
dp0p2p1           192.168.122.191/24  u/u
dp0p3p1           192.168.102.101/24  u/u
```

show interfaces dataplane detail

Displays detailed statistics and configuration information about all data plane interfaces.

```
show interfaces dataplane detail
```

Operational mode

Use this command to display detailed statistics and configuration information about all data plane interfaces.

The following example shows how to display detailed statistics and configuration information about all data plane interfaces.

```
vyatta@vyatta:~$ show interfaces dataplane detail

dp0p160p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 500
  link/ether 00:0c:29:19:5c:20 brd ff:ff:ff:ff:ff:ff
  inet 10.1.32.73/24 brd 10.1.32.255 scope global dp0p160p1
    valid_lft forever preferred_lft forever
  inet6 fe80::20c:29ff:fe19:5c20/64 scope link
    valid_lft forever preferred_lft forever

  RX:  bytes    packets    errors    dropped    overrun    mcast
      1260712    20717      0         6926      0         18727
  TX:  bytes    packets    errors    dropped    carrier    collisions
      71856      816        0         0         0         0

dp0p192p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 500
  link/ether 00:0c:29:19:5c:2a brd ff:ff:ff:ff:ff:ff
  inet 10.10.10.3/24 brd 10.10.10.255 scope global dp0p192p1
    valid_lft forever preferred_lft forever
  inet6 fe80::20c:29ff:fe19:5c2a/64 scope link
    valid_lft forever preferred_lft forever
```

```

RX:  bytes    packets    errors    dropped    overrun    mcast
     2052         24         0         10         0         12
TX:  bytes    packets    errors    dropped    carrier    collisions
     408          4         0         0         0         0
dp0p224p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 500
link/ether 00:0c:29:19:5c:34 brd ff:ff:ff:ff:ff:ff
inet 10.50.50.3/24 brd 10.50.50.255 scope global dp0p224p1
    valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe19:5c34/64 scope link
    valid_lft forever preferred_lft forever

RX:  bytes    packets    errors    dropped    overrun    mcast
     876         14         0         10         0         2
TX:  bytes    packets    errors    dropped    carrier    collisions
     408          4         0         0         0         0
dp0p256p1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 500
link/ether 00:0c:29:19:5c:3e brd ff:ff:ff:ff:ff:ff
inet 20.20.20.3/24 brd 20.20.20.255 scope global dp0p256p1
    valid_lft forever preferred_lft forever
inet6 2020:20::3/64 scope global
    valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe19:5c3e/64 scope link
    valid_lft forever preferred_lft forever

RX:  bytes    packets    errors    dropped    overrun    mcast
     9181         36         0         0         0         0
TX:  bytes    packets    errors    dropped    carrier    collisions
     636          6         0         0         0         0

```

show interfaces dataplane brief

Displays a brief status of a data plane interface.

```
show interfaces dataplane interface-name brief
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Operational mode

Use this command to display a brief status of a data plane interface.

The following example shows how to display a brief status of the dp0p1p3 interface.

```

vyatta@vyatta:~$ show interfaces dataplane dp0p1p3 brief
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address          S/L  Description
-----

```


dp0p1p3	192.18.1.1/27	u/u	10G Fibre
---------	---------------	-----	-----------

show interfaces dataplane physical

Displays information about the physical layer of a data plane interface.

```
show interfaces dataplane interface-name physical
```

interface-name


The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

Operational mode

Use this command to display information about the physical layer of a data plane interface.

The following example shows how to display information about the physical layer of a data plane interface.

```
vyatta@vyatta:~$ show interfaces dataplane dp0p1p1 physical
Settings for dp0p1p1:
  Supported ports: [ ]
  Supported link modes: Not reported
  Supported pause frame use: No
  Supports auto-negotiation: No
  Advertised link modes: Not reported
  Advertised pause frame use: No
  Advertised auto-negotiation: No
  Speed: 10000Mb/s
  Duplex: Full
  Port: Twisted Pair
  PHYAD: 0
  Transceiver: internal
  Auto-negotiation: on
  MDI-X: Unknown
  Current message level: 0xffffffffal (-95)
                                drv ifup tx_err tx_queued intr tx_done
rx_status pktdata hw wol 0xffff8000
  Link detected: yes
driver: rte_vmxnet3_pmd
version: 1.6
firmware-version:
bus-info: 0000:0b:00.0
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
vyatta@vyatta:~$
```

 **Note:** The Speed field always displays 10000Mb/s irrespective of the speed of the interface. The speed of virtual devices is dependent on the packet processing capability of the CPU and available memory instead of the hardware.

Related commands documented elsewhere

Commands for using other system features with data plane interfaces are located in the following guides or chapters in this guide.

Related Commands Documented Elsewhere	
Bridging	Layer 2 bridging is supported on data plane interfaces. Commands for configuring bridge groups are described in <i>Bridging Configuration Guide</i> .
DHCP	DHCP is supported on data plane interfaces. Commands for configuring DHCP are described in <i>Services Configuration Guide</i> .
Firewall	Firewall is supported on data plane interfaces. Commands for configuring firewall are described in <i>Firewall Configuration Guide</i> .
Link Bonding	Data plane link bonding is supported for data plane interfaces. Commands for configuring data plane link bonding are described in " Data Plane Interfaces ."
Multicast	Multicast is supported on data plane interfaces. Commands for configuring multicast are described in <i>Multicast Routing Configuration Guide</i> .
OSPF	OSPF is supported on data plane interfaces. Commands for configuring OSPF are described in <i>OSPF Configuration Guide</i> .
PIM	PIM is supported on data plane interfaces. Commands for configuring PIM are described in <i>PIM Configuration Guide</i> .
Policy Based Routing	Policy Based Routing is supported on data plane interfaces. Commands for configuring Policy Based Routing are described in <i>Policy-based Routing Configuration Guide</i> .
QoS	Quality of service traffic policies are supported on data plane interfaces. Commands for configuring QoS are described in <i>QoS Configuration Guide</i> .
RIP	RIP is supported on data plane interfaces. Commands for configuring RIP are described in <i>RIP Configuration Guide</i> .
RIPng	RIPng is supported on data plane interfaces. Commands for configuring RIPng are described in <i>RIPng Configuration Guide</i> .
System interfaces	Commands for showing the physical interfaces available on your system are described in <i>Basic System Configuration Guide</i> .
VLAN interfaces	802.1Q VLAN operation is supported on data plane interfaces. Commands for configuring VLAN interfaces (vifs) are described in Data Plane Interfaces .
VRRP	VRRP is supported on data plane interfaces and on VLAN interfaces configured under data plane interfaces. Commands for configuring VRRP are described in <i>High Availability Configuration Guide</i> .

Chapter 8. Ethernet Link Bonding Interface

Ethernet link bonding overview

In some operational scenarios, it makes sense to group multiple physical links together to create a larger virtual link. Grouping offers a way to increase performance between two devices without having to pay for a higher-speed physical link, and to provide redundancy so that connectivity still exists if a link fails. In a WAN, multilink Point-to-Point Protocol (MLPPP) is used to bundle multiple PPP links; in a LAN, Ethernet link bonding is used to bundle multiple Ethernet links.

Many implementations of Ethernet link bonding are nonstandard. The IEEE 802.3ad specification (now called IEEE 802.1ax) attempts to increase standardization. The IEEE 802.3ad standard has been adopted to varying degrees by all manufacturers. This standard specifies the general properties of the link, and defines the Link Aggregation Control Protocol (LACP).

802.3ad LACP is an active protocol that runs on Ethernet links that are configured for bonding. LACP allows peers to negotiate the automatic bonding of multiple links and helps detect situations in which one side is not configured correctly for link bonding. LACP also actively tests all physical connections between configured devices. Link failures can be detected even if other physical devices are attached to either end (for example, physical media converters) and would otherwise not show a link as down if a fault occurs in the middle of the physical link. If a link fails, traffic is redistributed dynamically to the remaining links.

The 802.3ad standard specifies that all physical links composing the bonded virtual link are full-duplex and point-to-point. Violation of either assumption can cause unexpected behavior in the bonded link.

The 802.3ad standard specifies that all packets belonging to a “conversation” must travel across the same physical link and that no packets may be duplicated. However, the abstraction of a conversation and the algorithm for assigning conversations to each link are incompletely specified. As a result, specific implementations vary, even between the ends of the bonded virtual link. This variance can lead to asymmetric traffic flow.

The number of links that can be bonded is limited by system capacity, especially memory. The Ethernet links in a bonded link need not be all the same speed.

Physical links that are added to a bonded link do not have to be operational when they are added. In the configuration for the bonded link, only the maximum transmission unit (MTU) is inherited from the bundle. That is, if you change the MTU of the bonded link, the MTU of the underlying Ethernet links is overridden. The remaining configuration is always taken from the configuration that is specified for the individual Ethernet link. The exception is that a physical link cannot be assigned an IP address if it is to be added to a bond group.

You can include VLANs within a bonded link; however, bundling multiple VLANs together as a bonded trunk is not supported. Because the purpose of bonding is to improve availability and performance, the bonded link requires actual physical links as a base.

Note: The performance of a bonding interface is limited by the performance of a single member interface, not the line rate of the aggregated interfaces.

The following actions can cause unexpected behavior and are not supported:

- Deleting a member link from a bonding interface and adding it to a different bonding interface.
- Deleting a member link from a bonding interface and adding it to the same bonding interface.
- Deleting a member link from a bonding interface and using it as an unbonded dataplane interface.
- Deleting all member links from a bonding interface and deleting the bonding interface itself.

Ethernet bonding configuration examples

This section presents the following topics:

- [Basic Ethernet bonding](#)
- [Ethernet bonding with VLAN](#)

Basic Ethernet bonding

To configure an Ethernet bonded link, create a bonding interface and configure it as any other Ethernet interface. Then, for each Ethernet interface that is to belong to the bonded link, specify the bond group (point to the bonding interface that you created).

The following figure shows a simple Ethernet link bonding scenario, with an Ethernet bonded link consisting of two physical Ethernet links.

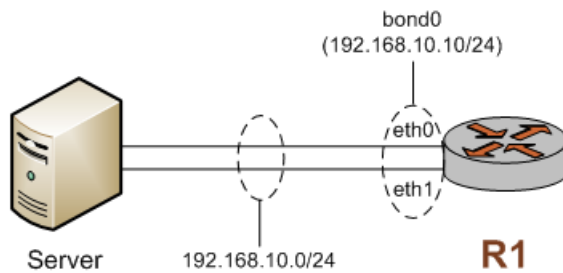
The figure has the following characteristics:

- The dp0bond0 bond group is created by using the default bonding mode (lacp).
- The dp0p1p1 and dp0p1p2 data plane interfaces represent the underlying physical links. Both are added as member links to the bonded interface dp0bond0.

No IP addresses are assigned to the individual physical Ethernet links. The bonding does not work if any of the component Ethernet links has an IP address assigned to it.

Use the `show interfaces` and `show interfaces bonding` commands to determine the status of the bonding interface and its constituent Ethernet interfaces.

Figure 3. Creating a bond group with two Ethernet interfaces



To configure this scenario, perform the following steps in configuration mode.

Table 7. Creating a bond group with two Ethernet interfaces

Step	Command
Create the dp0bond0 bonding group.	<pre>vyatta@R1# set interfaces bonding dp0bond0</pre>
Set the IP address for the bonding group to 192.168.10.10/24.	<pre>vyatta@R1# set interfaces bonding dp0bond0 address 192.168.10.10/24</pre>
Add dp0p1p1 to the dp0bond0 bonding group.	<pre>vyatta@R1# set interfaces dataplane dp0p1p1 bond-group dp0bond0</pre>
Add dp0p1p2 to the dp0bond0 bonding group.	<pre>vyatta@R1# set interfaces dataplane dp0p1p2 bond-group dp0bond0</pre>
Commit the change.	<pre>vyatta@R1# commit</pre>
Show the bonding group configuration.	<pre>vyatta@R1# show interfaces bonding dp0bond0 address 192.168.10.10/24</pre>
Show the dp0p1p1 configuration.	<pre>vyatta@R1# show interfaces bonding dp0p1p1 bond-group dp0bond0</pre>
Show the dp0p1p2 configuration.	<pre>vyatta@R1# show interfaces bonding dp0p1p2 bond-group dp0bond0</pre>

Ethernet bonding with VLAN

After a bonding interface is created, it is possible to create a VLAN within it. The following example is an extension of the previous example with the addition of a VLAN. The resulting bonding interface contains both VLAN and nonVLAN traffic.

To configure this scenario, perform the following steps in configuration mode.

Table 8. Adding a VLAN to an existing bonding interface.

Step	Command
Add the vif configuration to the bonding group.	<pre>vyatta@R1# set interfaces bonding dp0bond0 vif 192 address 10.192.248.225/24</pre>
Commit the change.	<pre>vyatta@R1# commit</pre>
Show the new bonding group configuration.	<pre>vyatta@R1# show interfaces bonding dp0bond0 address 192.168.10.10/24 mode lACP vif 192 { address 10.192.248.225/24</pre>

Table 8. Adding a VLAN to an existing bonding interface. (continued)

Step	Command
	}

Chapter 9. Ethernet Link Bonding Interface Commands

interfaces bonding

Defines an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx
delete interfaces bonding dpFbondx
show interfaces bonding [ dpFbondx ]
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

You can define more than one bond group by specifying multiple **bonding** configuration nodes.

Configuration mode

```
interfaces {
  bonding dpFbondx {
  }
}
```

Use this command to define an Ethernet link bonding interface, also known as a bond group. An Ethernet link bond group allows the bandwidth of individual links to be combined into a single virtual link.

You must create the bond group (using this command or one of its variants) before you can assign Ethernet interfaces to it.

Use the `set` form of this command to create a bond group.

Use the `delete` form of this command to remove an Ethernet link bond group.

Use the `show` form of this command to view the Ethernet link bond group configuration.

interfaces bonding address

Assigns a network address to an Ethernet link bond group.

```
set interfaces bonding dpFbondx address { address | dhcp | dhcpv6 }
delete interfaces bonding dpFbondx address
show interfaces bonding dpFbondx address
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

address

An IPv4 network address on this interface. The format of the address is *ip-address / prefix* (for example, 192.168.1.77/24).

You can define multiple IPv4 network addresses for a single interface, by creating multiple **address** configuration nodes.

dhcp

Specifies that the address and prefix are obtained from a DHCP server.

dhcpv6

Specifies that the address and prefix are obtained from a DHCPv6 server.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    address {
      address
      dhcp
      dhcpv6
    }
  }
}
```

Use this command to assign a network address to an Ethernet link bond group.

You can direct the interface to obtain its address and prefix from a Dynamic Host Configuration Protocol (DHCP) or DHCPv6 server by using the **dhcp** or **dhcpv6** option.

Use the `set` form of this command to set the IP address and network prefix. You can set more than one IP address for the interface by creating multiple **address** configuration nodes.

Use the `delete` form of this command to remove the address from an Ethernet link bond group.

Use the `show` form of this command to view the address of an Ethernet link bond group.

interfaces bonding description

Provides a description of an Ethernet link bond group.

```
set interfaces bonding dpFbondx description desc
```

```
delete interfaces bonding dpFbondx description
```

```
show interfaces bonding dpFbondx description
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

desc

A brief description of the bond group.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    description desc
  }
}
```

Use this command to provide a description of a bond group.

Use the `set` form of this command to provide a description of the bond group.

Use the `delete` form of this command to remove the description of a bond group.

Use the `show` form of this command to view the description of a bond group.

interfaces bonding dhcp-options no-rfc3442

Disables support for the classless static route option for DHCP on an Ethernet bonding interface.

```
set interfaces bonding dpFbondx dhcp-options no-rfc3442
```

```
delete interfaces bonding dpFbondx dhcp-options no-rfc3442
```

```
show interfaces bonding dpFbondx dhcp-options
```

The classless static route option for DHCP is enabled.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

no-rfc3442

Removes the classless static route option (121) from the parameter request list that a DHCP client sends to the DHCP server. For further information, refer to RFC 3442 at <https://tools.ietf.org/html/rfc3442>.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    dhcp-options {
      no-rfc3442
    }
  }
}
```

```

}
}
}

```

Note: This command is relevant only if the **dhcp** option has been set by using [interfaces dataplane address](#).

Note: Normally, this command is not required. It would be used only if the remote DHCP server is configured to provide classless static routes, but these routes are not required on the router that is configured to use the DHCP address.

Use the `set` form of this command to disable support for the DHCP classless static route option on an Ethernet bonding interface.

Use the `delete` form of this command to re-enable support for the DHCP classless static route option on an Ethernet bonding interface.

Use the `show` form of this command to display the status of the DHCP classless static route option on an Ethernet bonding interface.

interfaces bonding dhcpv6-options

Specifies the way in which a DHCPv6 client is to acquire an address, parameters, or all of them from a DHCPv6 server.

```
set interfaces bonding dpFbondx dhcpv6-options { parameters-only | temporary }
```

```
delete interfaces bonding dpFbondx dhcpv6-options { parameters-only |
temporary }
```

```
show interfaces bonding dpFbondx dhcpv6-options
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

parameters-only

Acquires only configuration parameters (and not an IPv6 address) from the DHCPv6 server.

Only one of the **parameters-only** and **temporary** parameters can be specified.

temporary

Acquires a temporary IPv6 address as described for IPv6 privacy addressing in RFC 4941.

Only one of the **parameters-only** and **temporary** parameters can be specified.

Configuration mode

```

interfaces {
  bonding dpFbondx {
    dhcpv6-options
    parameters-only
  }
}

```

```

    }
    temporary
}

```

Use this command to specify in what way the DHCPv6 client is to acquire an IPv6 address, parameters, or all of them from a DHCPv6 server.

These parameters are relevant only if the **dhcpv6** option has been set for the `interfaces bonding dpFbondx address` command.

The **parameters-only** option is typically used with Stateless Address Autoconfiguration (SLAAC) or static address configuration. It and the **temporary** parameter are mutually exclusive.

Use the `set` form of this command to specify the DHCPv6 options.

Use the `delete` form of this command to remove the DHCPv6 options.

Use the `show` form of this command to view the DHCPv6 configuration.

interfaces bonding disable

Disables an Ethernet link bond group without discarding configuration.

```
set interfaces bonding dpFbondx disable
```

```
delete interfaces bonding dpFbondx disable
```

```
show interfaces bonding dpFbondx
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

Configuration mode

```

interfaces {
    bonding dpFbondx {
        disable
    }
}

```

Use this command to disable an Ethernet link bond group without discarding configuration.

Use the `set` form of this command to disable the bond group.

Use the `delete` form of this command to enable the interface.

Use the `show` form of this command to view the configuration.

interfaces bonding flow-monitoring

Configures flow monitoring for a bonding interface.

```
set interfaces bonding dpFbondx flow-monitoring { aggregator agg-name |
exporter exp-name | selector sel-name }
```

```
delete interfaces bonding dpFbondx
```

```
show interfaces bonding dpFbondx
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

agg-name

Name of a flow monitoring aggregator.

exp-name

Name of a flow monitoring exporter.

sel-name

Name of a flow monitoring selector.

Configuration mode

```
interfaces {
  bonding dpFbondx flow-monitoring {
    aggregator agg-name
    exporter exp-name
    selector sel-name
  }
}
```

Use this command to configure flow monitoring for a bonding interface.

Use the `set` form of this command to provide a description of the bond group.

Use the `delete` form of this command to remove the description of a bond group.

Use the `show` form of this command to view the description of a bond group.

interfaces bonding ip enable-proxy-arp

Enables proxy ARP on an Ethernet link bonding interface.

```
set interfaces bonding dpFbondx ip enable-proxy-arp
```

```
delete interfaces bonding dpFbondx ip
```

```
show interfaces bonding dpFbondx
```

Proxy ARP is not enabled on the Ethernet link bonding interface.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    ip enable-proxy-arp
  }
}
```

Use this command to enable proxy Address Resolution Protocol (ARP) on an Ethernet link bonding interface.

Proxy ARP allows an Ethernet link bonding interface to respond with its own media access control (MAC) address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets that are sent to those destination IP addresses are forwarded appropriately by the system.

Use the `set` form of this command to enable proxy ARP on an interface.

Use the `delete` form of this command to disable proxy ARP on an interface.

Use the `show` form of this command to view the proxy ARP configuration.

interfaces bonding ip rip receive

Limits RIP packets that are received on an Ethernet link bonding interface to the specific version.

```
set interfaces bonding dpFbondx ip rip receive [ version ]
```

```
delete interfaces bonding dpFbondx ip rip receive
```

```
show interfaces bonding
```

Both versions of RIP packets are received.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

version

The version of RIP packets that are received (either 1 or 2 but not both)

Configuration mode

```
interfaces {
```

```

bonding dpFbondx {
    ip {
        rip {
            receive version
        }
    }
}

```

Use this command to limit the RIP packets that are received to a specific version (1 or 2).

Use the `set` form of this command to limit the receipt of RIP packets to a specific version.

Use the `delete` form of this command to receive both versions of RIP packets on an interface.

Use the `show` form of this command to view the configuration of RIP packets that are received on an interface.

interfaces bonding ip rip send

Limits RIP packets that are sent on an Ethernet link bonding interface to the specific version.

```
set interfaces bonding dpFbondx ip rip send [ version ]
```

```
delete interfaces bonding dpFbondx ip rip send
```

```
show interfaces bonding dpFbondx
```

Both versions of RIP packets are sent.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

version

The version of RIP packets that are sent (either 1 or 2 but not both)

Configuration mode

```

interfaces {
    bonding dpFbondx {
        ip {
            rip {
                send version
            }
        }
    }
}

```

Use this command to limit the RIP packets that are sent to a specific version (1 or 2).

Use the `set` form of this command to limit the sending of RIP packets to a specific version.

Use the `delete` form of this command to send both versions of RIP packets on an interface.

Use the `show` form of this command to view the configuration of RIP packets that are sent on an interface.

interfaces bonding ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ip tcp-mss limit clamp-value
```

```
delete interfaces bonding dpFbondx ip tcp-mss limit clamp-value
```

```
show interfaces bonding dpFbondx ip tcp-mss limit
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  bonding {
    ip {
      tcp-mss {
        limit
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding ip tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ip tcp-mss mtu
```

```
delete interfaces bonding dpFbondx ip tcp-mss mtu
```

```
show interfaces bonding dpFbondx ip tcp-mss mtu
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

Configuration mode

```
interfaces {
  bonding {
    ip {
      tcp-mss {
        mtu
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding ip tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ip tcp-mss mtu-minus subtrahend
```

```
delete interfaces bonding dpFbondx ip tcp-mss mtu-minus subtrahend
```

```
show interfaces bonding dpFbondx ip tcp-mss mtu-minus
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode


```

interfaces {
  bonding {
    ip {
      tcp-mss {
        mtu-minus
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding ipv6 address

Assigns an IPv6 address to an Ethernet link bonding interface.

```

set interfaces bonding interface-name ipv6 address [ autoconf | eui64
ipv6prefix ]

```

```

delete interfaces bonding interface-name ipv6 address [ autoconf | eui64
ipv6prefix ]

```

```

show interfaces bonding interface-name ipv6 address [ autoconf | eui64 ]

```

interface-name

The name of a bonding interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

autoconf

Generates an IPv6 address using the SLAAC protocol. Use this keyword if the interface is performing a “host” function rather than a “router” function. You can specify this keyword in addition to static IPv6, static IPv4, or IPv4 DHCP addresses on the interface.

eui64 *ipv6prefix*

Specifies the 64-bit IPv6 address prefix that is used to configure an IPv6 address in EUI-64 format. The system concatenates this prefix with a 64-bit EUI-64 value that is derived from the 48-bit MAC address of the interface.

Configuration mode

```

interfaces {
  bonding interface-name {
    ipv6 {
      address {
        autoconf

```

```

    eui64 ipv6prefix
  }
}
}

```

Use this command to assign an IPv6 address to a bonding interface.

Use the **autoconf** keyword to direct the system to automatically configure (autoconfigure) the address, using the Stateless Address Autoconfiguration (SLAAC) protocol defined in RFC 4862. Alternatively, you can provide an EUI-64 IPv6 address prefix so that the system constructs the IPv6 address.

If you want the system to use SLAAC to acquire an address on the interface, then in addition to setting this parameter, you must also disable IPv6 forwarding, either globally (by using the `system ipv6 disable-forwarding` command) or specifically on the interface.

Use the `set` form of this command to assign an IPv6 address to a bonding interface.

Use the `delete` form of this command to delete an IPv6 address from a bonding interface.

Use the `show` form of this command to display IPv6 address configuration settings.

interfaces bonding ipv6 ospfv3 area

Assigns an OSPFv3 area to a IPv6 Ethernet link bonding interface.

```
set interfaces bonding dpFbondx ipv6 ospfv3 area area-id
```

```
delete interfaces bonding dpFbondx ipv6 ospfv3 area
```

```
show interfaces bonding dpFbondx ipv6 ospfv3 area
```

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

area-id

Specifies an OSPFv3 area as an IP address or decimal value.

Configuration mode

```

interfaces {
  bonding dp0bondx {
    ipv6 {
      ospfv3 {
        area area-id
      }
    }
  }
}

```

Use this command to assign an OSPFv3 area to a bonding interface.

Use the `set` form of this command to assign an OSPFv3 area to a bonding interface.

Use the `delete` form of this command to delete an OSPFv3 area from a bonding interface.

Use the `show` form of this command to display OSPFv3 configuration settings.

interfaces bonding ipv6 ospfv3 process instance-id

Defines an OSPFv3 area for an Ethernet bonding interface.

```
set interfaces bonding dpFbondx ipv6 ospfv3 process process-id instance-id
instance-id
```

```
delete interfaces bonding dpFbondx ipv6 ospfv3
```

```
show interfaces bonding dpFbondx ipv6 ospfv3
```

area-id

The ID of an OSPFv3 area being configured, expressed as an IP address or a decimal value. The decimal value ranges from 0 to 4294967.

process-id

The OSPFv3 process ID. Enter alphanumeric characters.

Configuration mode

```
interfaces {
  bonding dp0bondx {
    ipv6 {
      ospfv3 {
        process process-id {
          area area-id
        }
      }
    }
  }
}
```

Use this command to define an area within an OSPFv3 AS.

Use the **set** form of this command to create an OSPFv3 area or define the parameters for a specified OSPFv3 area.

Use the **delete** form of this command to remove an OSPFv3 area.

Use the **show** form of this command to display the OSPFv3 area configuration.

interfaces bonding ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ipv6 tcp-mss limit clamp-value
```

```
delete interfaces bonding dpFbondx ipv6 tcp-mss limit clamp-value
```

```
show interfaces bonding dpFbondx ipv6 tcp-mss limit
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  bonding {
    ipv6 {
      tcp-mss {
        limit
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding ipv6 tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ipv6 tcp-mss mtu
```

```
delete interfaces bonding dpFbondx ipv6 tcp-mss mtu
```

```
show interfaces bonding dpFbondx ipv6 tcp-mss mtu
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

Configuration mode

```

interfaces {
  bonding {
    ipv6 {
      tcp-mss {
        mtu
      }
    }
  }
}

```

Use the `set` form of this command to clamp the MSS to the MTU for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx ipv6 tcp-mss mtu-minus subtrahend
```

```
delete interfaces bonding dpFbondx ipv6 tcp-mss mtu-minus subtrahend
```

```
show interfaces bonding dpFbondx ipv6 tcp-mss mtu-minus
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  bonding {
    ipv6 {
      tcp-mss {
        mtu-minus
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding lacp-options activity

Sets the LACP operational mode for an Ethernet link bond group.

```
set interfaces bonding dpFbondx lacp-options activity { active | passive }
delete interfaces bonding dpFbondx lacp-options
show interfaces bonding dpFbondx lacp-options
```

active

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

active

Specifies active participation in the exchange of LACP Data Unit (LACPDU).

passive

Specifies passive (default) participation in the exchange of LACPDU.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    lacp-options {
      activity {
        active
        passive
      }
    }
  }
}
```

Use this command to set the LACP operational mode for an Ethernet link bond group.

Use the `set` form of this command to set the default LACP operational mode for a bond group.

Use the `delete` form of this command to restore the default (passive) LACPDU participation for a bond group.

Use the `show` form of this command to view the configuration of LACPDU participation.

interfaces bonding lacp-options key

Sets the LACP aggregation (operational) key for an Ethernet link bond group.

```
set interfaces bonding dpFbondx lACP-options key value
```

```
delete interfaces bonding dpFbondx lACP-options key
```

```
show interfaces bonding dpFbondx lACP-options key
```

active

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

value

Specifies an LACP aggregation (operational) key.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    lACP-options {
      key value
    }
  }
}
```

Use this command to set the LACP aggregation (operational) key for an Ethernet link bond group.

Use the `set` form of this command to set is used to set the LACP aggregation (operational) key.

Use the `delete` form of this command is used to restore the LACP aggregation (operational) key.

Use the `show` form of this command to view the LACP aggregation (operational) key setting.

interfaces bonding mac

Sets the MAC address of an Ethernet link bond group.

```
set interfaces bonding dpFbondx mac mac-addr
```

```
delete interfaces bonding dpFbondx mac
```

```
show interfaces bonding dpFbondx mac
```

The MAC address used is the MAC address of the first interface added to the bond group.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

mac-addr

The MAC address of an Ethernet link bond group. The format of the address must be appropriate for the interface type. For an Ethernet interface, the format is six colon-separated 8-bit numbers in hexadecimal; for example, 00:0a:59:9a:f2:ba.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    mac mac-addr
  }
}
```

Use this command to set the media access control (MAC) address of an Ethernet link bond group.

Use the `set` form of this command to set the MAC address of a bond group.

Use the `delete` form of this command to remove the MAC address of an Ethernet link bond group.

Use the `show` form of this command to view MAC address of a Ethernet link bond group.

interfaces bonding mode

Sets the bonding mode for an Ethernet link bond group.

```
set interfaces bonding dpFbondx mode { active-backup | lacp | balanced }
```

```
delete interfaces bonding dpFbondx mode
```

```
show interfaces bonding dpFbondx mode
```

lacp is the default mode.

dpFbondx

The identifier a bond group. The identifier ranges from **dp0bond0** through **dp0bond99**.

active-backup

Uses an interfaces of a Link Aggregation Group as the active (primary) interface. A different interface becomes active if the primary Ethernet interface fails. The Link Aggregation Group interface MAC address is externally visible only on the active interface.

lacp

Uses LACP to dynamically create aggregation groups that share the same link speed and duplex settings.

balanced

Uses a hash to distribute the transmit load across member links. This mode exhibits the same behavior as the LACP mode, but it transmits without the control protocol.

Configuration mode

```

interfaces {
  bonding dpFbondx {
    mode {
      active-backup
      lacp
      balanced
    }
  }
}

```

Use this command to set the bonding mode for an Ethernet link bond group.

Use the `set` form of this command to set the bonding mode of a bond group.

Use the `delete` form of this command to restore the default bonding mode (`lacp`) for the bond group.

Use the `show` form of this command to view the current bonding mode for a bond group.

interfaces bonding mtu

Specifies the MTU for an Ethernet link bond group.

```
set interfaces bonding dpFbondx mtu mtu
```

```
delete interfaces bonding dpFbondx mtu
```

```
show interfaces bonding dpFbondx mtu
```

The MTU of the first Ethernet link added to the group is used.

dpFbondx

The identifier for a bond group. The identifier ranges from **`dp0bond0`** through **`dp0bond99`**.

mtu

The MTU, in octets, for the interface as a whole, including any logical interfaces that are configured for it. The MTU ranges from 68 to 9000.

Configuration mode

```

interfaces {
  bonding dpFbondx {
    mtu mtu
  }
}

```

Use this command to set the maximum transmission unit (MTU) for an Ethernet link bond group. The MTU is also applied to any vifs that are defined for the bonding interface.

Changing the MTU changes the MTU on the Ethernet links within the bond. Also, explicitly changing the MTU of the Ethernet links within the bond (by configuring the individual links) is not allowed.

During forwarding, IPv4 packets larger than the MTU are fragmented unless the DF bit is set. In that case, the packets are dropped and an ICMP “Packet too big” message is returned to the sender.

Use the `set` form of this command to set the MTU of a bond group.

Use the `delete` form of this command to restore the default MTU, which is the MTU of the first Ethernet link added to the group, and disable fragmentation.

Use the `show` form of this command to display the MTU for a bond group.

interfaces bonding primary

Sets the primary Ethernet interface within the Ethernet link bonding interface.

```
set interfaces bonding dpFbondx primary ifx
```

```
delete interfaces bonding dpFbondx primary
```

```
show interfaces bonding dpFbondx primary
```

No primary link is configured.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

ifx

The identifier of the primary Ethernet interface within the bond group. The identifier ranges from **d0p0p1** through **d0p2p3**.

Configuration mode

```
interfaces {
  bonding dpFbondx {
    primary ifx
  }
}
```

Use this command to specify the primary interface within the Ethernet link bond group

This option is available only when the bonding mode is Active Backup.

When the bonding mode is Active Backup and an interface is identified as the primary, the primary interface is always the only active member of the bonding interface so long as it is available. Only when the primary is offline are alternates used.

This option is useful when one member link is to be preferred over another; for example, when one member link has higher throughput than another.

Use the `set` form of this command to designate an Ethernet interface as the primary interface for Ethernet link bonding.

Use the `delete` form of this command to remove the primary Ethernet interface as the primary interface for Ethernet link bonding.

Use the `show` form of this command to view the primary Ethernet link bonding configuration for the Ethernet link bonding.

show interfaces bonding

Shows information about Ethernet link bond groups

```
show interfaces bonding dpFbondx brief
show interfaces bonding dpFbondx slaves
show interfaces bonding dpFbondx [ vif value ]
```

Information is displayed for all Ethernet link bond groups.

dpFbondx

The identifier for the bond group. Supported values are **dp0bond0** through **dp0bond99**.

brief

Displays information about the bonding slaves.

slaves

Displays information about the bonding slaves in brief format.

Operational mode

Use this command to show bond group information.

The following example shows the output for `show interfaces bonding` for all bonding interfaces.

```
vyatta@vyatta:~$ show interfaces bonding
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface          IP Address          S/L  Description
-----
dp0bond0           10.2.0.1/30         u/u
dp0bond0.200       10.200.0.1/24       u/u
```

The following example shows the output for a specific bond group.

```
vyatta@vyatta:~$ show interfaces bonding dp0bond0
  <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
  group default qlen 500
  link/ether 52:54:00:5a:b2:86 brd ff:ff:ff:ff:ff:ff
  inet 10.2.0.1/30 brd 10.2.0.3 scope global dp0bond0 valid_lft forever
  preferred_lft forever
  inet6 fe80::dc63:72ff:fe07:2c6/64 scope link valid_lft forever
  preferred_lft forever
  uptime: 45 seconds transitions: 1 last-change: 2015-04-09T18:31:45+0000
  RX: bytes packets errors ignored overrun mcast
      3126          27           0           0           0           0
  TX: bytes packets errors dropped carrier collisions
      3242          28           0           0           0
      0
```

The following example shows the output for `show interfaces bonding slaves`.

```
vyatta@vyatta:~$ show interfaces bonding slaves
  Interface          Mode          State Link Slaves
  dp0bond0           802.3ad       up    up    2
```

The following example shows the output for `show interfaces bonding dpFbondx slaves`.

```
vyatta@vyatta:~$ show interfaces bonding dp0bond0 slaves
  Interface      RX: bytes  packets      TX: bytes  packets      slctd LACP flags
  dp0bond0      15348      148          12184     129
  dp0s5         3784       35           8976      98           yes
  DISTRIBUTING
  dp0s6         11564      113          3208      31           yes
  DISTRIBUTING
```

The following example shows the output for `show interfaces bonding dpFbondx vif value`.

```
vyatta@vyatta:~$ show interfaces bonding dp0bond0 vif 200
  dp0bond0.200@dp0bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
  noqueue state UP group default
  link/ether 52:54:00:5a:b2:86 brd ff:ff:ff:ff:ff:ff
  inet 10.200.0.1/24 brd 10.200.0.255 scope global dp0bond0.200 valid_lft
  forever preferred_lft forever
  inet6 fe80::5054:ff:fe5a:b286/64 scope link valid_lft forever preferred_lft
  forever
```

```

uptime: 16 seconds transitions: 1 last-change: 2015-04-09T18:50:39+0000

RX:  bytes    packets    errors    ignored    overrun    mcast
    1018      11         0         0         0         8
TX:  bytes    packets    errors    dropped    carrier    collisions
    1018      11         0         0         0         0

```

Related commands documented elsewhere

The following documents describe other system features that can be used with bonded Ethernet link interfaces.

Related Commands Documented Elsewhere	
Bridging	Layer 2 bridging is supported on bonding interfaces. Commands for configuring bridge groups are described in <i>Bridging Configuration Guide</i> .
Firewall	Firewall is supported on bonding interfaces. Commands for configuring firewall are described in <i>Firewall Configuration Guide</i> .
OSPF	OSPF is supported on bonding interfaces. Commands for configuring OSPF are described in <i>OSPF Configuration Guide</i> .
OSPF	OSPFv3 is supported on bonding interfaces. Commands for configuring OSPFv3 are described in <i>OSPFv3 Configuration Guide</i> .
Policy Based Routing	Policy-based routing is supported on bonding interfaces. Commands for configuring Policy-based routing are described in <i>Policy-based Routing Configuration Guide</i> .
RIP	RIP is supported on bonding interfaces. Commands for configuring RIP are described in <i>RIP Configuration Guide</i> .
RIPng	RIPng is supported on bonding interfaces. Commands for configuring RIPng are described in <i>RIPng Configuration Guide</i> .
Virtual interfaces	802.1Q VLAN operation is supported on bonding interfaces. Commands for configuring virtual interfaces (vifs) are described in Ethernet Link Bonding Interface .
VRRP	VRRP is supported on bonding interfaces. Commands for configuring VRRP are described in <i>High Availability Configuration Guide</i> .

Chapter 10. VLAN Interfaces

VLAN interface overview

This section presents the following topics:

- [VLAN operation using virtual interfaces](#)
- [Interface types that support VLAN operation](#)
- [VLAN operation as opposed to multinetting](#)
- [Simultaneous Ethernet and 802.1q operation](#)
- [Referring to VLAN interfaces in commands](#)
- [IPv6 support](#)

VLAN operation using virtual interfaces

Some interface types can be configured for IEEE 802.1Q VLAN operation by using a virtual interface. On the router, virtual interfaces for VLANs are called vifs. As the identifier of a vif, you specify the VLAN to which the vif is connected.

Each physical interface can be configured with multiple vifs. Then, like a physical Ethernet interface, each vif can have multiple addresses assigned to it.

Interface types that support VLAN operation

VLAN interfaces can be configured on physical Ethernet interfaces.

VLAN operation as opposed to multinetting

VLANs are identified by a four-byte tag that is inserted at the front of the Layer 2 Ethernet header. Having this additional tag means that interfaces configured for 802.1Q are not compatible with standard Ethernet packets. When considering whether or not to use a VLAN interface, keep the following in mind.

- If you are using 802.1Q VLANs, create vif configuration nodes beneath the physical interface and assign the IP address to the vif.
- If you are not using 802.1Q but want to have multiple networks on the same physical interface (that is, you want to use multinetting but not VLANs), simply create multiple address configuration nodes directly under the physical interface without using vifs.

Simultaneous Ethernet and 802.1q operation

If your other network devices support it, an Ethernet interface may be used simultaneously as a standard port and an 802.1Q VLAN port. To do this, configure an IP address directly on the physical interface and then define a vif for the interface. Assign the VLAN ID as the vif identifier and configure an IP address for the vif. (This feature may not be compatible with all Ethernet switches; some switches require a physical Ethernet interface to be exclusively either an 802.1Q interface or a standard Ethernet interface.)

Referring to VLAN interfaces in commands

To refer to a vif within an `interfaces` command, such as `show interfaces` or `set interfaces`, specify the whole path to the vif configuration node, as in the following example.

```
show interfaces dataplane dp0p1p2 vif 40
```

When referring to the same vif within other commands—for example, enabling RIP on the interface—use the format of `if-x.vlan-id`, where `if-x` is the interface type plus the interface identifier (for example, `dp0p1p2`) and `vlan-id` is the VLAN ID (and also the identifier of the vif). The following example refers to the vif on VLAN 40 configured under the `dp0p1p2` interface.

```
set protocols rip interface dp0p1p2.40
```

IPv6 support

The router has extensive support for IPv6, including IPv6 interface addressing. This chapter describes the commands for configuring IPv6 on VLAN interfaces. *IPv6 Support Configuration Guide* fully describes router IPv6 support.

Examples of VLAN interface configuration

This section presents the following topics:

- [VLAN configuration](#)
- [IPv6 on VLAN interfaces](#)

VLAN configuration

This example shows how to configure the R1 router to have a vif for VLAN 40 on the `dp0p1p2` Ethernet interface. After configuring this VLAN, the R1 router has the following interfaces:

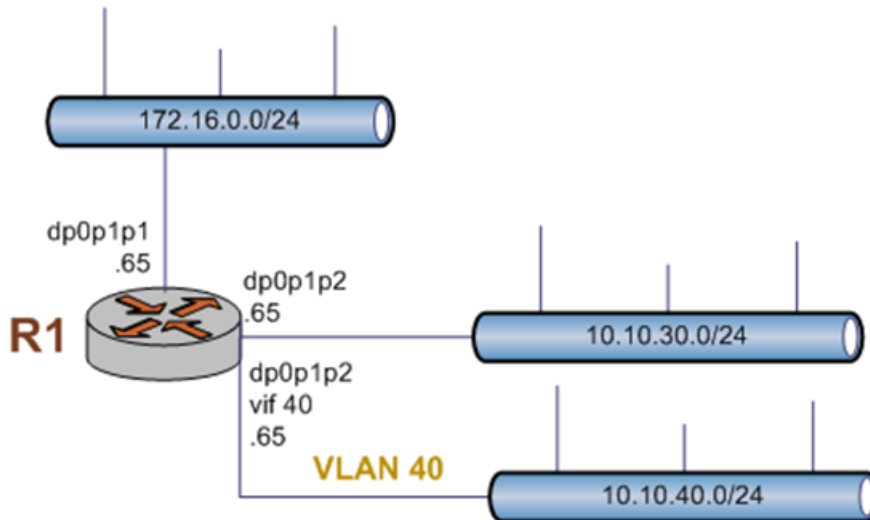
- One interface (`dp0p1p1`) that is configured as only a standard Ethernet interface. The IP address for this interface is `172.16.0.65`.

- One interface (dp0p1p2) that is configured as both a physical Ethernet interface and a VLAN interface. The IP address for the physical Ethernet interface is 10.10.30.65. The VLAN interface connects to VLAN 40 (the identifier of the vif) and has an IP address of 10.10.40.65.

Note: By default, the VLAN tag is the same as the vif. Only if `vlan-id` is defined does it override the tag value.

When you finish the example, the interfaces are configured as shown in the following figure.

Figure 4. VLAN configuration



To create and configure a VLAN interface, perform the following steps in configuration mode.

Table 9. Simultaneous Ethernet and VLAN operation

Step	Command
Assign an IP address directly to the dp0p1p1 untagged Ethernet interface.	<pre>vyatta@R1# set interfaces dataplane dp0p1p1 address 172.16.0.65/24</pre>
Assign an IP address directly to the dp0p1p2 untagged Ethernet interface.	<pre>vyatta@R1# set interfaces dataplane dp0p1p2 address 10.10.30.65/24</pre>
Create the configuration node for the vif. Assign the vif VLAN ID 40. Assign the IP address for the vif.	<pre>vyatta@R1# set interfaces dataplane dp0p1p2 vif 40 address 10.10.40.65/24</pre>
Commit the configuration.	<pre>vyatta@R1# commit</pre>
Commit and view the configuration.	<pre>vyatta@R1# show interfaces dataplane dataplane dp0p1p1 { address 172.16.0.65/24 } dataplane dp0p1p2 { address 10.10.30.65/24 vif 40 { address 10.10.40.65/24 } }</pre>

IPv6 on VLAN interfaces

IPv6 Support Configuration Guide provides examples of configuring IPv6 on interfaces.

Chapter 11. VLAN Interfaces Commands

interfaces bonding vif ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ip tcp-mss limit clamp-value
delete interfaces bonding dpFbondx vif vif-id ip tcp-mss limit clamp-value
show interfaces bonding dpFbondx vif vif-id ip tcp-mss limit
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  bonding {
    vif {
      ip {
        tcp-mss {
          limit
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding vif ip tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu
delete interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu
show interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```
interfaces {
  bonding {
    vif {
      ip {
        tcp-mss {
          mtu
        }
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding vif ip tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu-minus subtrahend
delete interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu-minus subtrahend
show interfaces bonding dpFbondx vif vif-id ip tcp-mss mtu-minus
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  bonding {
    vif {
      ip {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding vif ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss limit clamp-value
```

```
delete interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss limit clamp-value
```

```
show interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss limit
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```

interfaces {
  bonding {
    vif {
      ipv6 {
        tcp-mss {
          limit
        }
      }
    }
  }
}

```

```

    }
  }
}

```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding vif ipv6 tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu
```

```
delete interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu
```

```
show interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```

interfaces {
  bonding {
    vif {
      ipv6 {
        tcp-mss {
          mtu
        }
      }
    }
  }
}

```

Use the `set` form of this command to clamp the MSS to the MTU for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces bonding vif ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an Ethernet link bonding interface, also known as a bond group.

```
set interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu-minus subtrahend
```

```
delete interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu-minus subtrahend
```

```
show interfaces bonding dpFbondx vif vif-id ipv6 tcp-mss mtu-minus
```

dpFbondx

The identifier for the bond group: **dp0bond0** through **dp0bond99**.

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```
interfaces {
  bonding {
    vif {
      ipv6 {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a bond group.

Use the `delete` form of this command to remove the clamping for a bond group.

Use the `show` form of this command to show the clamping for a bond group.

interfaces dataplane vif

Assigns a vif ID.

```
set interfaces dataplane interface-name vif vif-id
```

```
delete interfaces dataplane interface-name vif vif-id
```

```
show interfaces dataplane interface-name vif vif-id
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id
  }
}
```

Use the `set` form of this command to assign an IP address and a network prefix to a vif.

Use the `delete` form of this command to delete a vif ID.

Use the `show` form of this command to display vif information.

interfaces dataplane vif address

Assigns an IP address and a network prefix to a vif.

```
set interfaces dataplane interface-name vif vif-id address { ipv4 | ipv6 | dhcp
| dhcpv6 }
```

```
delete interfaces dataplane interface-name vif vif-id address { ipv4 | ipv6 |
dhcp | dhcpv6 }
```

```
show interfaces dataplane interface-name vif vif-id address
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

ipv4

An IPv4 address and a network prefix for this vif. The format is *ip-address/prefix* (for example, 192.168.1.77/24).

You can define multiple IP addresses for a vif by creating multiple address configuration nodes.

ipv6

An IPv6 address and a network prefix for this vif. The format is *ipv6-address/prefix* (for example, 2001:db8:1234::/48).

You can define multiple IPv6 addresses for a vif by creating multiple address configuration nodes.

dhcp

Defines the interface as a DHCP client, which obtains its address and prefix from a DHCP server.

dhcpv6

Defines the interface as a DHCP for IPv6 client, which obtains its address and prefix from a DHCPv6 server.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      address [ipv4 | ipv6 | dhcp | dhcpv6]
    }
  }
}
```

Use this command to assign an IP address and a network prefix to a vif.

Use the `set` form of this command to assign an IP address and a network prefix to a vif.

Use the `delete` form of this command to delete an IP address and a network prefix on a vif.

Use the `show` form of this command to display an IP address and a network prefix for a vif.

interfaces dataplane vif bridge-group

Adds a vif to a bridge group.

```
set interfaces dataplane interface-name vif vif-id bridge-group { bridge
bridge-group-name | cost port-cost | priority port-priority | bpdu-guard |
root-block }
```

```
delete interfaces dataplane interface-name vif vif-id bridge-group [ bridge |
cost | priority ]
```

```
show interfaces dataplane interface-name vif vif-id bridge-group [ bridge |
cost | priority ]
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

bpdu-guard

Enables Spanning Tree Protocol PortFast Bridge Protocol Data Unit (BPDU) guard.

bridge *bridge-group-name*

Specifies the name of the bridge group.

cost *port-cost*

Specifies the numeric port cost. The range is 0 to 65535.

priority *port-priority*

Specifies the bridge port cost. The range is 0 to 63.

root-block

Enables Spanning Tree Protocol root guard.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      bridge-group
      bpdu-guard
      bridge bridge-group-name
      cost port-cost
      priority port-priority
      root-block
    }
  }
}

```

Use this command to add a vif to a bridge group.

Use the `set` form of this command to add a vif to a bridge group.

Use the `delete` form of this command to delete a vif from a bridge group.

Use the `show` form of this command to display a vif of a bridge group.

interfaces dataplane vif description

Describes a description for a vif.

```
set interfaces dataplane interface-name vif vif-id description description
```

```
delete interfaces dataplane interface-name vif vif-id description
```

```
show interfaces dataplane interface-name vif vif-id description
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

description

A description of the vif.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      description description
    }
  }
}
```

Use this command to describe a vif.

Use the `set` form of this command to describe a vif.

Use the `delete` form of this command to delete the description of a vif.

Use the `show` form of this command to display the description of a vif.

interfaces dataplane vif dhcpv6-options

Specifies the way in which a DHCPv6 client is to acquire an address, parameters, or both from the DHCPv6 server.

```
set interfaces dataplane interface-name vif vif-id dhcpv6-options [ parameters-only | temporary ]
```

```
delete interfaces dataplane interface-name vif vif-id dhcpv6-options [ parameters-only | temporary ]
```

```
show interfaces dataplane interface-name vif vif-id dhcpv6-options
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

parameters-only

Acquires only configuration parameters (and not an IPv6 address) from the DHCPv6 server.

Only one of the **parameters-only** and the **temporary** parameters can be specified.

temporary

Acquires a temporary IPv6 address as described for IPv6 privacy addressing in RFC 4941.

Only one of the **parameters-only** and the **temporary** parameters can be specified.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      dhcpv6-options [parameters-only | temporary]
    }
  }
}

```

Use this command to specify in what way the DHCPv6 client is to acquire an IPv6 address, parameters, or both from a DHCPv6 server.

Note that the parameters are relevant only if the **dhcpv6** option has been set for [interfaces dataplane vif](#).

The **parameters-only** parameter is typically used with Stateless Address Autoconfiguration (SLAAC) or static address configuration. The **parameters-only** and **temporary** parameters are mutually exclusive.

Use the `set` form of this command to specify the DHCPv6 options for a vif.

Use the `delete` form of this command to remove the DHCPv6 options from a vif.

Use the `show` form of this command to view DHCPv6 option configuration of a vif.

interfaces dataplane vif disable

Disables a vif without discarding configuration.

```
set interfaces dataplane interface-name vif vif-id disable
```

```
delete interfaces dataplane interface-name vif vif-id disable
```

```
show interfaces dataplane interface-name vif vif-id
```

The vif is enabled.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      disable
    }
  }
}

```

```
}
}
```

Use this command to disable a vif without discarding configuration for a vif.

Use the `set` form of this command to disable a vif.

Use the `delete` form of this command to enable a vif.

Use the `show` form of this command to display whether a vif is disabled or enabled.

interfaces dataplane vif disable-link-detect

Directs a vif not to detect physical link-state changes.

```
set interfaces dataplane interface-name vif vif-id disable-link-detect
delete interfaces dataplane interface-name vif vif-id disable-link-detect
show interfaces dataplane interface-name vif vif-id
```

A vif detects physical link-state changes.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      disable-link-detect
    }
  }
}
```

Use this command to direct a vif not to detect physical state change to the underlying Data plane link (for example, when a cable is unplugged).

Use the `set` form of this command to disable detection of physical link-state changes for a vif.

Use the `delete` form of this command to enable detection of physical link-state changes for a vif.

Use the `show` form of this command to display whether detection of physical link-state changes is disabled or enabled on a vif.

interfaces dataplane vif ip disable-forwarding

Disables IPv4 forwarding on a vif.

```
set interfaces dataplane interface-name vif vif-id ip disable-forwarding
delete interfaces dataplane interface-name vif vif-id ip disable-forwarding
show interfaces dataplane interface-name vif vif-id ip disable-forwarding
```

IPv4 packets are forwarded.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ip {
        disable-forwarding
      }
    }
  }
}
```

Use this command to disable IPv4 packet forwarding on a vif.

You can also disable IPv4 forwarding globally (that is, for all interfaces) by using the `system ipv4 disable-forwarding` command.

Use the `set` form of this command to disable IPv4 packet forwarding on a vif.

Use the `delete` form of this command to enable IPv4 packet forwarding on a vif.

Use the `show` form of this command to display whether IPv4 packet forwarding is disabled or enabled on a vif.

interfaces dataplane vif ip enable-proxy-arp

Enables proxy ARP on a vif.

```
set interfaces dataplane interface-name vif vif-id ip enable-proxy-arp
delete interfaces dataplane interface-name vif vif-id ip enable-proxy-arp
show interfaces dataplane interface-name vif vif-id ip enable-proxy-arp
```

Proxy ARP is not enabled on the vif.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ip {
        enable-proxy-arp
      }
    }
  }
}
```

Use this command to enable proxy Address Resolution Protocol (ARP) on a vif.

Proxy ARP allows a data plane interface to respond with its own media access control (MAC) address to ARP requests for destination IP addresses on subnets that are attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Use the `set` form of this command to enable proxy ARP on a vif.

Use the `delete` form of this command to return the system to its default behavior, that is, proxy ARP is not enabled on a data plane interface.

Use the `show` form of this command to display whether IPv4 packet forwarding is disabled or enabled on a vif.

interfaces dataplane vif ip rip receive

Configures receive options for RIP packets on a vif.

```
set interfaces dataplane interface-name vif vif-id ip rip receive [ version version ]
```

```
delete interfaces dataplane interface-name vif vif-id ip rip receive
```

```
show interfaces dataplane interface-name vif vif-id ip rip receive
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

version

Specifies a version of RIP packets. Legal values are 1, 2, or **both**.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ip {
        rip {
          receive [version version]
        }
      }
    }
  }
}
```

Use this command to configure receive options for RIP packets on a vif.

Use the `set` form of this command to configure the receive options for a vif, including an optional version.

Use the `delete` form of this command to restore the default configuration for a vif.

Use the `show` form of this command to display the configuration for a vif.

interfaces dataplane vif ip rip send

Configures send options for RIP packets on a vif.

```
set interfaces dataplane interface-name vif vif-id ip rip send [ version
version ]
```

```
delete interfaces dataplane interface-name vif vif-id ip rip send
```

```
show interfaces dataplane interface-name vif vif-id ip rip send
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

version

Specifies a version of RIP packets. Legal values are 1, 2, or **both**.

Configuration mode

```
interfaces {
```

```

dataplane interface-name {
    vif vif-id {
        ip {
            rip {
                send [version version]
            }
        }
    }
}

```

Use this command to configure send options for RIP packets on a vif.

Use the `set` form of this command to configure the send options for a vif, including an optional version.

Use the `delete` form of this command to restore the default configuration for a vif.

Use the `show` form of this command to display the configuration for a vif.

interfaces dataplane vif ip rpf-check

Specifies reverse path filter (RPF) policy (see RFC3704).

```

set interfaces dataplane interface-name vif vif-id ip rip rpf-check { disable
| loose | strict }

```

```

delete interfaces dataplane interface-name vif vif-id ip rip rpf-check

```

```

show interfaces dataplane interface-name vif vif-id ip rip rpf-check

```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

disable

Does not validate the source.

loose

Enables Loose Reverse Path Forwarding as defined in RFC3704

strict

Enable Strict Reverse Path Forwarding as defined in RFC3704.

Configuration mode

```

interfaces {
    dataplane interface-name {
        vif vif-id {

```



```

        ip {
            rpf-check {disable | loose | strict}
        }
    }
}

```

Use this command to specify reverse path filter (RPF) policy for a vif (see RFC3704).

Use the `set` form of this command to configure the policy for a vif.

Use the `delete` form of this command to restore the default configuration of a vif.

Use the `show` form of this command to display the configuration of a vif.

interfaces dataplane vif ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ip tcp-mss limit clamp-value
```

```
delete interfaces dataplane interface-name vif vif-id ip tcp-mss limit clamp-value
```

```
show interfaces dataplane interface-name vif vif-id ip tcp-mss limit
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```

interfaces {
    dataplane {
        vif {
            ip {
                tcp-mss {
                    limit
                }
            }
        }
    }
}

```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ip tcp-mss mtu

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU for a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ip tcp-mss mtu
```

```
delete interfaces dataplane interface-name vif vif-id ip tcp-mss mtu
```

```
show interfaces dataplane interface-name vif vif-id ip tcp-mss mtu
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```
interfaces {
  dataplane {
    vif {
      ip {
        tcp-mss {
          mtu
        }
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ip tcp-mss mtu-minus

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU less a given value a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ip tcp-mss mtu-minus
subtrahend
```

```
delete interfaces dataplane interface-name vif vif-id ip tcp-mss mtu-minus
subtrahend
```

```
show interfaces dataplane interface-name vif vif-id ip tcp-mss mtu-minus
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```
interfaces {
  dataplane {
    vif {
      ip {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ipv6 address

Assigns an IPv6 address to a vif.

```
set interfaces dataplane interface-name vif vif-id ipv6 address [ autoconf |
eui64 ipv6prefix | link-local ipv6-address ]
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 address [ autoconf |
eui64 ipv6prefix | link-local ]
```

```
show interfaces dataplane interface-name vif vif-id ipv6 address
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

autoconf

Generates an IPv6 address using the SLAAC protocol. Use this keyword if the interface is performing a “host” function rather than a “router” function. You can specify this keyword in addition to static IPv6, static IPv4, or IPv4 DHCP addresses on the vif.

eui64 ipv6prefix

Specifies the 64-bit IPv6 address prefix that is used to configure an IPv6 address, in EUI-64 format. The system concatenates this prefix with a 64-bit EUI-64 value that is derived from the 48-bit MAC address of the vif.

link-local ipv6-address

Specifies the 128-bit IPv6 address.

Configuration mode

```

interfaces {
    dataplane interface-name {
        vif vif-id {
            ipv6 {
                address [autoconf | eui64 ipv6prefix | link-local
                ipv6-address]
            }
        }
    }
}

```

Use this command to assign an IPv6 address to a vif.

Use the **autoconf** keyword to direct the system to automatically configure (autoconfigure) the address, using the Stateless Address Autoconfiguration (SLAAC) protocol defined in RFC 4862. Alternatively, you can provide an EUI-64 IPv6 address prefix so that the system constructs the IPv6 address.

If you want the system to use SLAAC to acquire addresses on the interface, then in addition to setting this parameter, you must also disable IPv6 forwarding, either globally (by using the `system ipv6 disable-forwarding` command) or specifically on the interface (by using [interfaces dataplane vif ipv6 disable-forwarding](#)).

Use the `set` form of this command to assign an IPv6 address to a vif.

Use the `delete` form of this command to delete an IPv6 address from a vif.

Use the `show` form of this command to view IPv6 address configuration settings for a vif.

interfaces dataplane vif ipv6 disable

Disables IPv6 on a vif.

```
set interfaces dataplane interface-name vif vif-id ipv6 disable
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 disable
```

```
show interfaces dataplane interface-name vif vif-id ipv6 disable
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vfi-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode.

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        disable
      }
    }
  }
}
```

By default, IPv6 is enabled on all interfaces. A global command exists which can disable IPv6, namely `set system ipv6 disable`, and this will take precedence over any of the existing per-interface based, IPv6 commands.

IPv6 Forwarding can be disabled via the `interfaces dataplane <interface-name> vif <vif-id> ipv6 disable-forwarding` command, but note that IPv6 traffic can still be terminated on this interface.

IPv6 configuration can be totally disabled via the `interfaces dataplane <interface-name> vif <vif-id> ipv6 disable` command.

Use the `set` form of this command to disable IPv6 on this interface.

Use the `delete` form of this command to enable IPv6 on this interface.

Use the `show` form of this command to display the current IPv6 configuration on this interface.

interfaces dataplane vif ipv6 disable-forwarding

Disables IPv6 forwarding on a vif.

```
set interfaces dataplane interface-name vif vif-id ipv6 disable-forwarding
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 disable-forwarding
show interfaces dataplane interface-name vif vif-id ipv6 disable-forwarding
```

IPv6 packets are forwarded.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        disable-forwarding
      }
    }
  }
}
```

Use this command to disable IPv6 packet forwarding on a vif.

You can also disable IPv6 forwarding globally (that is, for all interfaces) by using the `system ipv6 disable-forwarding` command.

Use the `set` form of this command to disable IPv6 packet forwarding on a vif.

Use the `delete` form of this command to enable IPv6 packet forwarding on a vif.

Use the `show` form of this command to display whether IPv6 packet forwarding is disabled or enabled on a vif.

interfaces dataplane vif ipv6 dup-addr-detect-transmits

Specifies the number of NS packets to transmit as part of the DAD process.

```
set interfaces dataplane interface-name vif vif-id ipv6 dup-addr-detect-transmits num
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 dup-addr-detect-transmits
```

```
show interfaces dataplane interface-name vif vif-id ipv6 dup-addr-detect-transmits
```

One NS packet is transmitted as part of the DAD process.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

num

The number of NS messages to transmit as part of the DAD process. The number ranges from 1 through 4294967295. The default number is 1.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        dup-addr-detect-transmits num
      }
    }
  }
}
```

Use this command to specify the number of Neighbor Solicitation (NS) packets to transmit as part of the Duplicate Address Detection (DAD) process on a vif.

Use the `set` form of this command to specify the number of NS packets to transmit on a vif.

Use the `delete` form of this command to delete the transmission number from a vif and transmit the default number of one NS packet.

Use the `show` form of this command to display the number of NS packets to transmit on a vif.

interfaces dataplane vif ipv6 mld

Enables MLD on an interface.

```
set interfaces dataplane interface-name vif vif-id ipv6 mld
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 mld
```

```
show interfaces dataplane interface-name vif vif-id ipv6 mld
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```


interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        mld
      }
    }
  }
}


```

Use this command to enable Multicast Listener Discovery (MLD) on a vif.

This command enables MLD operation in stand-alone mode, and can be used to learn local membership information prior to enabling a multicast routing protocol on the vif.

This command can only be issued on VLAN interfaces.

 **Note:** Enabling IP on a vif enables the host-side functionality of MLD by default. The `set interfaces dataplane interface-name vif vif-id ipv6 mld` command enables the router-side functionality of the MLD on the given vif.

 **Note:** To use MLD for multicast routing, multicast routing must be enabled on the router. For information about multicast routing in general, see the *Multicast Routing Configuration Guide*.

Use the `set` form of this command to enable MLD on a vif.

Use the `delete` form of this command to remove all MLD configuration and disable MLD on a vif.

Use the `show` form of this command to display MLD configuration for a vif.

interfaces dataplane vif ipv6 ospfv3

Enables OSPFv3 on a specified interface.

```

set interfaces dataplane interface-name vif vif-id ipv6 ospfv3 [ process
process-id [ instance-id instance-id ] ]

```

```

delete interfaces dataplane interface-name vif vif-id ipv6 ospfv3 [ [ process
process-id instance-id instance-id ] ]

```

```

show interfaces dataplane interface-name vif vif-id ipv6 ospfv3 [ process
process-id [ instance-id instance-id ] ]

```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

process-id

The OSPFv3 process ID. Enter alphanumeric characters.

instance-id

The OSPFv3 instance ID. The range is from 0 to 255.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        ospfv3 [process process-id [instance-id instance-id]]
      }
    }
  }
}
```

Use this command to enable the OSPFv3 routing protocol on a vif.

Use the `set` form of this command to enable OSPFv3 on a vif.

Use the `delete` form of this command to remove all OSPFv3 configuration and disable OSPFv3 on a vif.

Use the `show` form of this command to display OSPFv3 configuration for a vif.

interfaces dataplane vif ipv6 pim

Enables Protocol Independent Multicast for a vif.

```
set interfaces dataplane interface-name vif vif-id ipv6 pim
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 pim
```

```
show interfaces dataplane interface-name vif vif-id ipv6 pim
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

Configuration mode

```
interfaces {
  dataplane interface-name {
```

```

vif vif-id {
    ipv6 {
        pim
    }
}

```

Use this command to enable Protocol Independent Multicast (PIM) for a vif.

Use the `set` form of this command to restrict the flow of BSR messages through a vif.

Use the `delete` form of this command to restore the default behavior of a vif.

Use the `show` form of this command to display the current BSR border configuration for a vif.

interfaces dataplane vif ipv6 router-advert

Specifies the router advertisements to be sent from a vif.

```

set interfaces dataplane interface-name vif vif-id ipv6 router-advert [
cur-hop-limit limit ] [ default-lifetime lifetime ] [ default-preference
preference ] [ link-mtu mtu ] [ managed-flag state ] [ max-interval interval
] [ min-interval interval ] [ other-config-flag state ] [ prefix ipv6net [
autonomous-flag state | on-link-flag state | preferred-lifetime lifetime |
valid-lifetime lifetime ] ] [ reachable-time time ] [ retrans-timer time ] [
send-advert state ]

```

```

delete interfaces dataplane interface-name vif vif-id ipv6 router-advert [
cur-hop-limit ] [ default-lifetime ] [ default-preference ] [ link-mtu ]
[ managed-flag ] [ max-interval ] [ min-interval ] [ other-config-flag ] [
prefix ipv6net [ autonomous-flag | on-link-flag | preferred-lifetime | valid-
lifetime ] ] [ reachable-time ] [ retrans-timer [ send-advert ] ]

```

```

show interfaces dataplane interface-name vif vif-id ipv6 router-advert

```

Router advertisements are not sent on a vif.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

cur-hop-limit limit

Specifies the Hop Count field of the IP header for outgoing (unicast) IP packets. This value is placed in the Hop Count field of the IP header for outgoing (unicast) IP packets. The range is 0 to 255. The default is 64. A limit of 0 means unspecified by the router.

default-lifetime lifetime

Specifies the lifetime, in seconds, that is associated with the default router. A lifetime of 0 indicates that the router is not a default router. The lifetime ranges from the value configured

for the **max-interval** option to 9000 (18.2 hours). If the lifetime is not configured, the value for this timer is three times **max-interval**.

default-preference *preference*

The preference associated with the default router. Supported values are as follows:

low: The default router is low preference.

medium: The default router is medium preference.

high: The default router is high preference.

The default is **medium**.

link-mtu *mtu*

Specifies the MTU to be advertised for the link. The MTU is 0 or ranges from 1280 through the maximum MTU for the type of link, as defined in RFC 2464. The default MTU is 0, which means the MTU is not specified in the router advertisement message. That is because it is expected that the MTU is configured directly on the interface itself and not for routing advertisements. You can configure this option when the link MTU is not well known.

If the MTU that is set here does not match the MTU that is configured on the interface, the system issues a warning but does not fail.

managed-flag *state*

Whether to use the administered protocol for address autoconfiguration. The state is either of the following:

true: Hosts use the administered (stateful) protocol for address autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration.

false: Hosts use only stateless address autoconfiguration.

The default state is **false**.

max-interval *interval*

Specifies the maximum time, in seconds, that is allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 4 through 1800. The default is 600 (10 minutes).

min-interval *interval*

Specifies the minimum time, in seconds, that is allowed between sending unsolicited multicast router advertisements from the interface. The interval ranges from 3 through 0.75 times the **max-interval** option. The default interval is 0.33 times **max-interval**.

other-config-flag *state*

Specifies that the interface use the administered (stateful) protocol for autoconfiguration of nonaddress information, as defined in RFC 4862. The state is either of the following:

true: Hosts use the administered protocol for autoconfiguration of nonaddress information.

false: Hosts use stateless autoconfiguration of nonaddress information.

The default state is **false**.

prefix *ipv6net*

Multinode. Specifies the IPv6 prefix to be advertised on the IPv6 interface, in the format *ipv6-address/prefix*.

You can define more than one IPv6 prefix by configuring multiple prefix configuration nodes.

autonomous-flag *state*

Specifies whether the prefix can be used for autonomous address configuration as defined in RFC 4862. The state is either of the following:

true: The prefix can be used for autonomous address configuration.

false: The prefix cannot be used for autonomous address configuration.

The default state is **true**.

on-link-flag *state*

Specifies whether the prefix can be used for onlink determination, as defined in RFC 4862. The state is either of the following:

true: The prefix can be used for onlink determination.

false: The advertisement makes no statement about onlink or off-link properties of the prefix. For instance, the prefix might be used for address configuration with some addresses belonging to the prefix being onlink and others being off-link.

The default state is **true**.

preferred-lifetime *lifetime*

Specifies the length of time, in seconds, that the addresses generated from the prefix by Stateless Address Autoconfiguration (SLAAC) is to remain preferred, as defined in RFC 4862. The interval is with respect to the time the packet is sent. The lifetime ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0xFFFFFFFF.) The default lifetime is 604800 (7 days).

valid-lifetime *lifetime*

Specifies the length of time, in seconds, that the prefix is valid for onlink determination, as defined in RFC 4862. The interval is with respect to the time the packet is sent. The time ranges from 1 through 4294967296 plus the **infinity** keyword, which represents forever. (The actual value of **infinity** is a byte in which all bits are set to 1s: 0xFFFFFFFF.) The default lifetime is 2592000 (30 days).

reachable-time *time*

Specifies the length of time, in milliseconds, for which the system assumes a neighbor is reachable after having received a reachability confirmation. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Section 7.3 of RFC 2461). The time ranges from 0 through 3600000, where a value of 0 means the reachable time is not specified in the router advertisement message. The default time is 0.

retrans-timer *time*

Specifies the length of time, in milliseconds, between retransmitted NS messages. This time is used by address resolution and the Neighbor Unreachability Detection algorithm (see Sections 7.2 and 7.3 of RFC 2461). The time ranges from 0 through 4294967295, where a value of 0 means the retransmit time is not specified in the router advertisement message. The default time is 0.

send-advert *state*

Specifies whether router advertisements are to be sent from this interface. The state is either of the following:

true: Sends router advertisements from this interface.

false: Does not send router advertisements from this interface. If a state is in effect, parameters in this configuration subtree are still used to configure the local implementation parameters.

The default state is **true**.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        router-advert {
          cur-hop-limit limit
          default-lifetime lifetime
          default-preference preference
          link-mtu mtu
          managed-flag state
          max-interval interval
          min-interval interval
          other-config-flag state
          prefix ipv6net {
            autonomous-flag state
            on-link-flag state
            preferred-lifetime lifetime
            valid-lifetime lifetime
          }
          reachable-time time
          retrans-timer time
          send-advert state
        }
      }
    }
  }
}

```

Use this command to specify the router advertisements to be sent from a vif.

Router advertisements are sent by IPv6 routers to advertise their existence to hosts on the network. IPv6 hosts do not send router advertisements.

If the **router-advert** node of the configuration tree is missing, router advertisements are not sent. In addition, if IPv6 forwarding is disabled either globally (by using the `system ipv6 disable-forwarding` command) or on the interface (by using [interfaces dataplane vif ipv6 disable-forwarding](#)), router advertisements are not sent.

Most router advertisement parameters are required by either the Neighbor Discovery (ND) protocol or the Stateless Address Autoconfiguration (SLAAC) protocol. These parameters are used both locally for the IPv6 implementation and become part of the RA messages sent to hosts on the network so that they can be configured appropriately.

Use the `set` form of this command to create the router-advert configuration node and begin to send router advertisements on a vif.

Use the `delete` form of this command to delete router-advert configuration node and stop sending router advertisements on a vif.

Use the `show` form of this command to view router advertisement configuration for a vif.

interfaces dataplane vif ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ipv6 tcp-mss limit clamp-value
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 tcp-mss limit clamp-value
```

```
show interfaces dataplane interface-name vif vif-id ipv6 tcp-mss limit
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  dataplane {
    vif {
      ipv6 {
        tcp-mss {
          limit
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ipv6 tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu
delete interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu
show interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```
interfaces {
  dataplane {
    vif {
      ipv6 {
        tcp-mss {
          mtu
        }
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for a data plane interface.

```
set interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu-minus
subtrahend
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu-minus
subtrahend
```

```
show interfaces dataplane interface-name vif vif-id ipv6 tcp-mss mtu-minus
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  dataplane {
    vif {
      ipv6 {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for a data plane interface.

Use the `delete` form of this command to remove the clamping for a data plane interface.

Use the `show` form of this command to show the clamping for a data plane interface.

interfaces dataplane vif ipv6 unnumbered donor-interface preferred-address

Configures the vif of the specified data plane interfaces as an IP unnumbered interface with the donor interface and preferred address.

```
set interfaces dataplane interface-name vif vif-id ipv6 unnumbered donor-
interface interface preferred-address ipv6-address
```

```
delete interfaces dataplane interface-name vif vif-id ipv6 unnumbered
```

```
show interfaces dataplane interface-name vif vif-id ipv6 unnumbered
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

interface

Specifies the interface.

ipv6-address

Specifies the 128-bit IPv6 address of another interface.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      ipv6 {
        unnumbered donor-interface interface preferred-address
      }
    }
  }
}

```

Configures IP processing on a vif with the requirement of an assigned IP address.

Use the `set` form of this command to configure the processing.

Use the `delete` form of this command to restore the default configuration.

Use the `show` form of this command to display the configuration.

interfaces dataplane vif mtu

Set the maximum transmission unit (MTU) for a vif.

```
set interfaces dataplane interface-name vif vif-id mtu mtu
```

```
delete interfaces dataplane interface-name vif vif-id mtu
```

```
show interfaces dataplane interface-name vif vif-id mtu
```

The default size of the MTU is 1500.

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a vif. The ID ranges from 1 through 4094.

mtu

The size of the MTU, in octets, for the vif as a whole. The size of the MTU ranges from 68 through 9000.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif vif-id {
      mtu mtu
    }
  }
}

```

```
}
}
```

Use this command to set the size of the maximum transmission unit (MTU) for a vif.

During forwarding, IPv4 packets larger than the MTU are fragmented unless the “Don't Fragment” (DF) bit is set in the IP header. In that case, the packets are dropped and an ICMP “fragmentation needed” message is returned to the sender.

Note that MTU sizes larger than 1500 cause the system to generate “jumbo frames,” which are not compatible with some data plane interface cards and devices.

If the data plane interface is part of an Ethernet link bond, MTU settings for the bonded link override MTU settings for the physical interface.

Use the `set` form of this command to specify the size of the MTU.

Use the `delete` form of this command to delete the size of the MTU and restore the default size of 1500.

Use the `show` form of this command to display the current size of the MTU.

interfaces dataplane vif vlan

Creates a vif.

```
set interfaces dataplane interface-name vif vif-id [ vlan vlan-id ]
```

```
delete interfaces dataplane interface-name vif vif-id [ vlan vlan-id ]
```

```
show interfaces dataplane interface-name vif vif-id [ vlan vlan-id ]
```

dataplane *interface-name*

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif *vif-id*

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

vlan *vlan-id*

Multinode. The VLAN ID of a vif, for use with 802.1Q VLAN tagging. The ID ranges from 1 through 4094. Note that only 802.1Q-tagged packets are accepted on Ethernet vifs.

You can define more than one vif for an interface by creating multiple vif configuration nodes.

Configuration mode


```
interfaces {
  dataplane interface-name {
    vif vif-id {
      vlan vlan-id
    }
  }
}
```

```

}
}
}

```

Use this command to create a vif. The vifs function as VLAN interfaces, and only 802.1Q tagged packets are accepted.

 **Note:** The interface or bonding group must be defined before a vif can be added. For Ethernet bonded links, group members must also be assigned.

Use the `set` form of this command to create a vif.

Use the `delete` form of this command to delete a vif.

Use the `show` form of this command to display a vif.

interfaces dataplane vif inner-vlan

Add an inner-VLAN to a virtual interface.

```
set interfaces dataplane interface-name vif vif-id inner-vlan inner-vlan-id
```

```
delete interfaces dataplane interface-name vif vif-id inner-vlan inner-vlan-id
```

```
show interfaces dataplane interface-name vif vif-id inner-vlan
```

vif *vif-id*

The ID of a virtual interface. The ID ranges from 1 through 4094.

inner-vlan *inner-vlan-id*

The inner-VLAN ID of a virtual interface, for use with 802.1Q VLAN tagging. The ID ranges from 1 through 4094. Note that only 802.1Q-tagged packets are accepted on Ethernet vifs.

You can define more than one vif for an interface by creating multiple vif configuration nodes.

Configuration mode

```

interfaces {
    dataplane
        interface-name {
            vif vif-id {
                vlan inner-vlan-id
            }
        }
}

```

Use the `set` form of this command to add an inner-VLAN to a virtual interface.

Use the `delete` form of this command to delete an inner-VLAN to a virtual interface.

Use the `show` form of this command to display the inner-VLAN of a virtual interface.

interfaces switch default-port-parameters mode

Specifies the default mode of switch ports.

```
set interfaces switch interface-name default-port-parameters mode { access | trunk }
```

```
delete interfaces switch interface-name default-port-parameters mode { access | trunk }
```

```
show interfaces switch interface-name default-port-parameters mode { access | trunk }
```

interface-name

The name of an interface.

access

Only allow untagged VLANs on this port.

trunk

Allow multiple VLANs on this port

Configuration mode

```
switch {
    default-port-parameters {
        mode
    }
}
```

Use this command to specify the the default mode of switch ports.

Use the `set` form of this command to set the mode.

Use the `delete` form of this command to delete the mode.

Use the `show` form of this command to display the mode.

interfaces switch default-port-parameters vlan-parameters vlans

Specifies the VLANs allowed by default to ingress port on a switch.

```
set interfaces switch interface-name default-port-parameters vlan-parameters vlans vlan-ids
```

```
delete interfaces switch interface-name default-port-parameters vlan-parameters vlans vlan-ids
```

```
show interfaces switch interface-name default-port-parameters vlan-parameters
vlangs vlan-ids
```

interface-name

The name of a switch.

vlan-ids

VLANs allowed to ingress to this port.

Configuration mode

```
interfaces {
  switch {
    default-port-parameters {
      vlan-parameters {
        vlangs
      }
    }
  }
}
```

Use the `set` form of this command to set the VLANs allowed by default to ingress port on a switch.

Use the `delete` form of this command to delete the VLANs allowed by default to ingress port on a switch.

Use the `show` form of this command to display the VLANs allowed by default to ingress port on a switch..

interfaces switch physical-switch

Specifies a physical switch contained in this switch.

```
set interfaces switch interface-name physical-switch switch_id
set interfaces switch interface-name physical-switch switch_id
show interfaces switch interface-name physical-switch switch_id
```

interface-name

The name of a switch.

switch_id

Physical switch contained in this switch in the range 0-4294967295.

Configuration mode

```
interfaces {
  switch {
    physical-switch
```

```
}
}
```

Use the `set` form of this command to specify a physical switch contained in this switch.

Use the `delete` form of this command to delete a physical switch contained in this switch.

Use the `show` form of this command to display a physical switch contained in this switch.

interfaces dataplane interface duplex

Sets the LAN instances to a dataplane interface.

```
set interfaces dataplane interface-name duplex [auto | full | half]
```

```
delete interfaces dataplane interface-name duplex [auto | full | half]
```

```
show interfaces dataplane interface-name duplex [auto | full | half]
```

interface-name

Name of the interfaces.

Configuration mode

```
interfaces {
  dataplane dp0p1p1 {
    duplex {
      auto
      full
      half
    }
  }
}
```

Use this command to define the parameters for redistribution of LAN.

Use the `set interfaces dataplane interface-name duplex [auto | full | half]` form of this command to set LAN parameters.

Use the `delete interfaces dataplane interface-name duplex [auto | full | half]` form of this command to remove LAN parameters.

Use the `show interfaces dataplane interface-name duplex [auto | full | half]` form of this command to display LAN configuration.

interfaces loopback interface ipv6 disable

Sets the IPv6 ospfv3 instances to a loopback interface.

```
set interfaces loopback interface-name ipv6 disable
delete interfaces loopback interface-name ipv6 disable
show interfaces loopback interface-name ipv6 disable
```

interface-name

Name of the interfaces.

Configuration mode

```
interfaces {
  loopback dp0p1p1 {
    ipv6 {
      disable
    }
  }
}
```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

Use the `set interfaces loopback interface-name ipv6 disable` form of this command to set loopback parameters.

Use the `delete interfaces loopback interface-name ipv6 disable` form of this command to remove loopback parameters.

Use the `show interfaces loopback interface-name ipv6 disable` form of this command to display loopback configuration.

interfaces loopback interface ipv6 ospfv3 instance

Sets the IPv6 ospfv3 instances to a loopback interface.

```
set interfaces loopback interface-name ipv6 ospfv3 instance-id instance-id [
AREA value | cost value | dead-interval value | hello-intervalvaluelink-lsa-
suppressionvaluemtu-ignoreneighboraddressnetworktypepriorityvalueretransmit-
intervalvaluetransmit-delayvalue]
```

```
delete interfaces loopback interface-name ipv6 ospfv3 instance-id instance-id [
AREA value | cost value | dead-interval value | hello-intervalvaluelink-lsa-
suppressionvaluemtu-ignoreneighboraddressnetworktypepriorityvalueretransmit-
intervalvaluetransmit-delayvalue]
```

```
show interfaces loopback interface-name ipv6 ospfv3 instance-id instance-id [
AREA value | cost value | dead-interval value | hello-intervalvaluelink-lsa-
suppressionvaluemtu-ignoreneighboraddressnetworktypepriorityvalueretransmit-
intervalvaluetransmit-delayvalue]
```

interface-name

Name of the interfaces.

instance-id

Id of the instance

area

A route map for which routes are redistributed.

interval

An interval in seconds during which the virtual link should wait to detect Hello packets from neighboring routers before declaring the neighbor down. The range is from 1 to 65535. The default is 40 seconds.

transmit-delay

The delay, in seconds, between link-state transmits. This value must be the same for all nodes on the network. The range is 1 to 65535. The default is 1.

Configuration mode

```

interfaces {
  loopback dp0p1p1 {
    ipv6 {
      ospfv3 {
        instance-id 1 {
          area area-value
          cost cost-value
          dead-interval interval-value
          hello-interval interval-value
          link-lsa-suppression value
          mtu-ignore
          neighbor address
          priority priority-value
          retransmit-interval interval-value
          transmit-delay value
        }
      }
    }
  }
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

 **Note:** This command applies to IPv4 unicast routes only.

Use the `set interfaces loopback interface-name ipv6 ospfv3 instance-id <instance-id> [area <value> | cost <value> | dead-interval <value> | hello-interval <value> | link-`

lsa-suppression <value> | mtu-ignore | neighbor <address> | network <type> | priority <value> | retransmit-interval <value> | transmit-delay <value>] form of this command to set loopback interface parameters.

Use the `delete interfaces loopback interface-name ipv6 ospfv3 intance-id <instance-id> [area <value> | cost <value> | dead-interval <value> | hello-interval <value> | link-lsa-suppression <value> | mtu-ignore | neighbor <address> | network <type> | priority <value> | retransmit-interval <value> | transmit-delay <value>]` form of this command to remove loopback interface parameters.

Use the `show interfaces loopback interface-name ipv6 ospfv3 intance-id <instance-id> [area <value> | cost <value> | dead-interval <value> | hello-interval <value> | link-lsa-suppression <value> | mtu-ignore | neighbor <address> | network <type> | priority <value> | retransmit-interval <value> | transmit-delay <value>]` form of this command to display loopback interface configuration.

interfaces dataplane bridge-group

Sets the instances to a dataplane interface.

```
set interfaces dataplane interface-name bridge-group [pvst-filter | pvst-guard]
```

```
delete interfaces dataplane interface-name bridge-group [pvst-filter | pvst-guard]
```

```
show interfaces dataplane interface-name bridge-group [pvst-filter | pvst-guard]
```

interface-name

Name of the interfaces.

Configuration mode

```
interfaces {
    dataplane <interface-name> {
        bridge-group {
            pvst-filter
            pvst-guard
        }
    }
}
```

Use this command to define the parameters for redistribution of BGP routes into vlan.

Use the `set interfaces dataplane <interface-name> bridge-group [pvst-filter | pvst-guard]` form of this command to set vlan route redistribution parameters.

Use the `delete interfaces dataplane <interface-name> bridge-group [pvst-filter | pvst-guard]` form of this command to remove vlan route redistribution parameters.

Use the `show interfaces dataplane <interface-name> bridge-group [pvst-filter | pvst-guard]` form of this command to display vlan route redistribution configuration.

interfaces dataplane interface duplex

Sets the LAN instances to a dataplane interface.

```
set interfaces dataplane interface-name duplex [auto | full | half]
```

```
delete interfaces dataplane interface-name duplex [auto | full | half]
```

```
show interfaces dataplane interface-name duplex [auto | full | half]
```

interface-name

Name of the interfaces.

Configuration mode

```
interfaces {
  dataplane dp0p1p1 {
    duplex {
      auto
      full
      half
    }
  }
}
```

Use this command to define the parameters for redistribution of LAN.

Use the `set interfaces dataplane interface-name duplex [auto | full | half]` form of this command to set LAN parameters.

Use the `delete interfaces dataplane interface-name duplex [auto | full | half]` form of this command to remove LAN parameters.

Use the `show interfaces dataplane interface-name duplex [auto | full | half]` form of this command to display LAN configuration.

interfaces dataplane switch-group

Sets the instances to a dataplane interface.

```
set interfaces dataplane interface-name switch-group [pvst-filter | pvst-guard]
```

```
delete interfaces dataplane interface-name switch-group [pvst-filter | pvst-guard]
```

```
show interfaces dataplane interface-name switch-group [pvst-filter | pvst-guard]
```

interface-name

Name of the interfaces.

Configuration mode

```

interfaces {
    dataplane <interface-name> {
        switch-group {
            pvst-filter
            pvst-guard
        }
    }
}

```

Use this command to define the parameters for redistribution of BGP routes into vlan.

Use the `set interfaces dataplane <interface-name> switch-group [pvst-filter | pvst-guard]` form of this command to set vlan route redistribution parameters.

Use the `delete interfaces dataplane <interface-name> switch-group [pvst-filter | pvst-guard]` form of this command to remove vlan route redistribution parameters.

Use the `show interfaces dataplane <interface-name> switch-group [pvst-filter | pvst-guard]` form of this command to display vlan route redistribution configuration.

interfaces dataplane tagnode switch-group port-parameters vlan-parameters

Sets the interface instances to a vlan.

```

set interfacesdataplanetagnodeswitch-groupport-parametersvlan-parametersqos-
parametersvlanvlan-idpolicyqos value

```

```

delete interfacesdataplanetagnodeswitch-groupport-parametersvlan-parametersqos-
parametersvlanvlan-idpolicyqos value

```

```

show interfacesdataplanetagnodeswitch-groupport-parametersvlan-parametersqos-
parametersvlanvlan-idpolicyqos value

```

interface-name

Name of the interfaces.

Configuration mode

```

interfaces {
    data-plane <tagnode> {
        switch-group {
            port-parameters {

```

```

vlan-parameters
{
  qos-parameters {
    vlan <vlan-id> {
      policy {
        qos <value>
      }
      qos <value>
    }
  }
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

 **Note:** This command applies to IPv4 unicast routes only.

Use the `set interfaces dataplane <tagnode> switch-group port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to set vlan interface parameters.

Use the `delete interfaces dataplane <tagnode> switch-group port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to remove vlan interface parameters.

Use the `show interfaces dataplane <tagnode> switch-group port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to display vlan interface configuration.

interfaces dataplane tagnode switch-group port-parameters policy

Sets the interface instances to a vlan.

```
set interfaces dataplane tagnode switch-group port-parameters policy qosvalue vlan-
modifyname
```

```
delete interfaces dataplane tagnode switch-group port-parameters policy qosvalue vlan-
modifyname
```

```
show interfaces dataplane tagnode switch-group port-parameters policy qos value vlan-
modify name
```

interface-name

Name of the interfaces.

Configuration mode

```
interfaces {
  data-plane <tagnode> {
    switch-group {
      port-parameters {
        policy {
          qos
        }
      }
    }
  }
  <value>
  vlan-modify <name>
  qos <value>
}
```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

 **Note:** This command applies to IPv4 unicast routes only.

Use the `set interfaces dataplane <tagnode> switch-group port-parameters policy qos <value> vlan-modify <name>` form of this command to set vlan interface parameters.

Use the `delete interfaces dataplane <tagnode> switch-group port-parameters policy qos <value> vlan-modify <name>` form of this command to remove vlan interface parameters.

Use the `show interfaces dataplane <tagnode> switch-group port-parameters policy qos <value> vlan-modify <name>` form of this command to display vlan interface configuration.

interfaces dataplane vif bridge-group

Sets the instances to a dataplane interface.

```
set interfaces dataplane interface-name vifnumber bridge-group [pvst-filter |
pvst-guard]
```

```
delete interfaces dataplane interface-name vifnumber bridge-group [pvst-filter |
pvst-guard]
```

```
show interfaces dataplane interface-name vifnumber bridge-group [pvst-filter |
pvst-guard]
```

interface-name

Name of the interfaces.

Configuration mode

```

interfaces {
    dataplane <interface-name> {
        vif <number> {
            bridge-group {
                pvst-filter
                pvst-guard
            }
        }
    }
}

```

Use this command to define the parameters for redistribution of BGP routes into vlan.

Use the `set interfaces dataplane <interface-name> vif <number> bridge-group [pvst-filter | pvst-guard]` form of this command to set vlan route redistribution parameters.

Use the `delete interfaces dataplane <interface-name> vif <number> bridge-group [pvst-filter | pvst-guard]` form of this command to remove vlan route redistribution parameters.

Use the `show interfaces dataplane <interface-name> vif <number> bridge-group [pvst-filter | pvst-guard]` form of this command to display vlan route redistribution configuration.

interfaces l2tpeth vif ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN vif vif-id ip tcp-mss limit clamp-value
```

```
delete interfaces l2tpeth ltpN vif vif-id ip tcp-mss limit clamp-value
```

```
show interfaces l2tpeth ltpN vif vif-id ip tcp-mss limit
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```

interfaces {
  l2tpeth {
    vif {
      ip {
        tcp-mss {
          limit
        }
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth vif ip tcp-mss mtu

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu
```

```
delete interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu
```

```
show interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```

interfaces {
  l2tpeth {
    vif {
      ip {
        tcp-mss {
          mtu
        }
      }
    }
  }
}

```

Use the `set` form of this command to clamp the MSS to the MTU for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth vif ip tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value on an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu-minus subtrahend
```

```
delete interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu-minus subtrahend
```

```
show interfaces l2tpeth ltpN vif vif-id ip tcp-mss mtu-minus
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```
interfaces {
  l2tpeth {
    vif {
      ip {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth vif ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss limit clamp-value
```

```
delete interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss limit clamp-value
```

```
show interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss limit
```

lttpN

L2TPv3 static tunnel interface identifier (*lttp0* through *lttpN*).

vif-id

Virtual interface identifier (1-4094).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  l2tpeth {
    vif {
      ipv6 {
        tcp-mss {
          limit
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth vif ipv6 tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu
```

```
delete interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu
```

```
show interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu
```

lttpN

L2TPv3 static tunnel interface identifier (*lttp0* through *lttpN*).

vif-id

Virtual interface identifier (1-4094).

Configuration mode

```
interfaces {
  l2tpeth {
    vif {
      ipv6 {
        tcp-mss {
          mtu
        }
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth vif ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu-minus subtrahend
```

```
delete interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu-minus subtrahend
```

```
show interfaces l2tpeth lttpN vif vif-id ipv6 tcp-mss mtu-minus
```

lttpN

L2TPv3 static tunnel interface identifier (*lttp0* through *lttpN*).

vif-id

Virtual interface identifier (1-4094).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  l2tpeth {
    vif {
      ipv6 {
        tcp-mss {
          mtu-minus
        }
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces switch default-port-parameters vlan-parameters

Sets the interface instances to a vlan.

```

set interfaces switch name default-port-parameters vlan-parameters qos-parameters vlan vlan-id policy qos value

```

```

delete interfaces switch name default-port-parameters vlan-parameters qos-parameters vlan vlan-id policy qos value

```

```

show interfaces switch name default-port-parameters vlan-parameters qos-parameters vlan vlan-id policy qos value

```

interface-name

Name of the interfaces.

Configuration mode

```

interfaces {
  switch <name> {
    default-port-parameters {
      vlan-parameters {

    qos-parameters {

      vlan <vlan-id> {

        policy {

```

```

      qos <value>
    }
  }
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

 **Note:** This command applies to IPv4 unicast routes only.

Use the `set interfaces switch <name> default-port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to set vlan interface parameters.

Use the `delete interfaces switch <name> default-port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to remove vlan interface parameters.

Use the `show interfaces switch <name> default-port-parameters vlan-parameters qos-parameters vlan <vlan-id> policy qos <value>` form of this command to display vlan interface configuration.

Related commands documented elsewhere

Commands for using other system features with VLAN interfaces are located in the following guides.

Related Commands Documented Elsewhere	
IGMP and MLD	IGMP and MLD are supported on IPv4 VLAN interfaces. Commands for configuring IGMP and MLD are described in <i>IGMP Configuration Guide</i> .
Multicast	Multicast is supported on IPv4 VLAN interfaces. Commands for configuring multicast are described in <i>Multicast Routing Configuration Guide</i> .
OSPF	OSPF is supported on IPv4 VLAN interfaces. Commands for configuring OSPF are described in <i>OSPF Configuration Guide</i> .
OSPFv3	OSPFv3 is supported on IPv4 VLAN interfaces. Commands for configuring OSPFv3 are described in <i>OSPFv3 Configuration Guide</i> .
PIM	PIM is supported on IPv4 and IPv6 VLAN interfaces. Commands for configuring PIM are described in <i>PIM Configuration Guide</i> .
RIP	RIP is supported on IPv4 VLAN interfaces. Commands for configuring RIP are described in <i>RIP Configuration Guide</i> .
RIPng	RIPng is supported on IPv6 VLAN interfaces. Commands for configuring RIPng are described in <i>RIPng Configuration Guide</i> .

policy vlan-modify rule

Creates the vlan policy actions.

```
set policy vlan-modify name rule rule-numdirection[egress|ingress]
delete policy vlan-modify name rule rule-numdirection[egress|ingress]
show policy vlan-modify name rule rule-numdirection[egress|ingress]
```

vlan-modify

Name of the vlan.

Configuration mode

```
policy {
    vlan-modify <name> {
        rule <rule-num> {
            direction {
                egress
                ingress
            }
        }
    }
}
```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

Use the `set policy vlan-modify <name> rule <rule-num> direction [egress | ingress]` form of this command to set policy action parameters.

Use the `delete policy vlan-modify <name> rule <rule-num> direction [egress | ingress]` form of this command to delete policy action parameters.

Use the `show policy vlan-modify <name> rule <rule-num> direction [egress | ingress]` form of this command to display policy action parameters.

policy vlan-modify rule action

Creates the vlan policy actions.

```
set policy vlan-modify name rule rule-numactionpop
delete policy vlan-modify name rule rule-numactionpop
show policy vlan-modify name rule rule-numactionpop
```

vlan-modify

Name of the vlan.

Configuration mode

```

policy {
    vlan-modify <name> {
        rule <rule-num> {
            action {
                pop
            }
        }
    }
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

Use the `set policy vlan-modify <name> rule <rule-num> action pop` form of this command to set policy action parameters.

Use the `delete policy vlan-modify <name> rule <rule-num> action pop` form of this command to delete policy action parameters.

Use the `show policy vlan-modify <name> rule <rule-num> action pop` form of this command to display policy action parameters.

policy vlan-modify rule action

Creates the vlan policy actions.

```

set policy vlan-modify name rule rule-numactionpush [pcpvalue|tag-protocol-
id[0x8100 |0x88A8|0x88a8]|vlan-idvalue

```

```

delete policy vlan-modify name rule rule-numactionpush [pcpvalue|tag-protocol-
id[0x8100 |0x88A8|0x88a8]|vlan-idvalue

```

```

show policy vlan-modify name rule rule-numactionpush [pcpvalue|tag-protocol-
id[0x8100 |0x88A8|0x88a8]|vlan-idvalue

```

vlan-modify

Name of the vlan.

Configuration mode

```

policy {
    vlan-modify <name> {
        rule <rule-num> {
            action {
                push {
                    pcp <value>
                }
            }
        }
    }
}

```

```

tag-protocol-id
0x8100 tag-protocol-id
tag-protocol-id
0x88A8 tag-protocol-id
tag-protocol-id
0x88a8 vlan-id <value>
}
}
}
}
}
}
}
}
}
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

Use the `set policy vlan-modify <name> rule <rule-num> action push [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to set policy action parameters.

Use the `delete policy vlan-modify <name> rule <rule-num> action push [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to delete policy action parameters.

Use the `show policy vlan-modify <name> rule <rule-num> action push [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to display policy action parameters.

policy vlan-modify rule action swap

Creates the vlan policy actions.

```
set policy vlan-modify name rule rule-numactionsswap [pcpvalue|tag-protocol-id[0x8100 |0x88A8|0x88a8]|vlan-idvalue
```

```
delete policy vlan-modify name rule rule-numactionsswap [pcpvalue|tag-protocol-id[0x8100 |0x88A8|0x88a8]|vlan-idvalue
```

```
show policy vlan-modify name rule rule-numactionsswap [pcpvalue|tag-protocol-id[0x8100 |0x88A8|0x88a8]|vlan-idvalue
```

vlan-modify

Name of the vlan.

Configuration mode

```
policy {
    vlan-modify <name> {
        rule <rule-num> {
            action {
                swap {
```

```

0x8100          pcp <value>
                tag-protocol-id
0x88A8          tag-protocol-id
0x88a8          tag-protocol-id
                vlan-id <value>
                }
            }
        }
    }
}

```

Use this command to define the parameters for redistribution of BGP routes into OSPFv3.

Use the `set policy vlan-modify <name> rule <rule-num> action swap [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to set policy action parameters.

Use the `delete policy vlan-modify <name> rule <rule-num> action swap [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to delete policy action parameters.

Use the `show policy vlan-modify <name> rule <rule-num> action swap [pcp <value> | tag-protocol-id 0x8100 | tag-protocol-id 0x88A8 | tag-protocol-id 0x88a8 | vlan-id <value>]` form of this command to display policy action parameters.

policy vlan-modify rule selector

Creates the vlan policy actions.

```
set policy vlan-modify name rule rule-numselectorvlan-idany
```

```
delete policy vlan-modify name rule rule-numselectorvlan-idany
```

```
show policy vlan-modify name rule rule-numselectorvlan-idany
```

vlan-modify

Name of the vlan.

Configuration mode

```

policy {
    vlan-modify <name> {
        rule <rule-num> {
            selector {
                vlan-id {
                    any
                }
            }
        }
    }
}

```


Operational mode

Use the `show policy vlan-modify <interface-name>` command to display information about the policy valn information.

Use this command to display information about configured vlan-modify.

When used with no option, this command displays information about all active vlan. When the identifier of a interface is provided, this command displays information for the specified interface.

Chapter 12. Switch Virtual Interfaces

interfaces switch vif address

Specifies the IP address and network prefix for a virtual interface.

```
set interfaces switch interface-name vif <1-4094> address { ipv4 | ipv6 | dhcp  
| dhcpv6 }
```

```
delete interfaces switch interface-name vif <1-4094>address [ ipv4 | ipv6 |  
dhcp | dhcpv6 ]
```

```
show interfaces switch interface-name vif <1-4094>address [ ipv4 | ipv6 | dhcp  
| dhcpv6 ]
```

interface-name

The name of a data plane interface.

vif

Virtual interface identifier <1-4094>.

ipv4

The IPv4 address of the interface. The format is *ip-address/prefix* (for example, 192.168.1.77/24).

ipv6

The IPv6 address of the interface. The format is *ipv6-address/prefix* (for example, 2001:db8:1234::/48).

dhcp

Interface gets its address and prefix from a DHCP server.

dhcpv6

Interface gets its address and prefix from a DHCPv6 server.

Configuration mode

```
interfaces {  
  switch {  
    vif {  
      address {  
      }  
    }  
  }  
}
```

Use the `set` form of this command to set an IP address and a network prefix for a virtual interface. You can set only one IP address for the interface.

Use the `delete` form of this command to delete an IP address and network prefix for a virtual interface.

Use the `show` form of this command to display an IP address and network prefix for a virtual interface.

interfaces switch vif description

Specifies a description for a virtual interface.

```
set interfaces switch interface-name vif vif-id description description
```

```
delete interfaces switch interface-name vif vif-id description description
```

```
show interfaces switch interface-name vif vif-id description description
```

interface-name

The name of a data plane interface.

vif-id

Multinode. The identifier of a virtual interface in the range 1-4094.

description

Virtual interface description.

Configuration mode

```
interfaces {
  switch {
    vif {
      address {
      }
    }
  }
}
```

Use the `set` form of this command to set a description.

Use the `delete` form of this command to delete a description.

Use the `show` form of this command to display a description.

interfaces switch vif ip ospf authentication key-chain

Specifies the authentication for an OSPF interface.

```
set interfaces switch interface-name vif vif-id ip ospf authentication key-chain key-chain-name
```

```
delete interfaces switch interface-name vif vif-id ip ospf authentication key-chain key-chain-name
```

```
show interfaces switch interface-name vif vif-id ip ospf authentication key-chain key-chain-name
```

interface-name

The name of a data plane interface.

vif-id

Multinode. The identifier of a virtual interface in the range 1-4094.

key-chain-name

Key chain name

Configuration mode

```
interfaces {
  switch {
    vif {
      ip {
        ospf {
          authentication {
            keychain {
            }
          }
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the authentication for an OSPF interface.

Use the `delete` form of this command to delete the authentication for an OSPF interface.

Use the `show` form of this command to display the authentication for an OSPF interface.

interfaces switch vif ip ospf fall-over bfd

Initiates a BFD session for all OSPFv2 neighbors on a virtual interface.

```
set interfaces switch interface-name vif vif-id ip ospf fall-over bfd
```

```
delete interfaces switch interface-name vif vif-id ip ospf fall-over bfd
```

```
show interfaces switch interface-name vif vif-id ip ospf fall-over bfd
```

interface-name

The name of a data plane interface.

vif-id

Multinode. The ID of a virtual interface. The ID ranges from 1 through 4094.

Configuration mode

```

interfaces {
  switch {
    vif {
      ip {
        ospf {
          fall-over {
            bfd
          }
        }
      }
    }
  }
}

```

Use the `set` form of this command to initiate a BFD session for all OSPFv2 neighbors on a virtual interface.

Use the `delete` form of this command to end a BFD session for all OSPFv2 neighbors on a virtual interface.

Use the `show` form of this command to display a BFD session for all OSPFv2 neighbors on a virtual interface.

interfaces switch vif ip ospf network

Sets the OSPF network type for a virtual interface.

```

set interfaces switch interface-name vif vif-id ip ospfnetwork [ broadcast |
non-broadcast | point-to-multipoint | point-to-point ]

```

```

delete interfaces switch interface-name vif vif-id ip ospfnetwork [ broadcast |
non-broadcast | point-to-multipoint | point-to-point ]

```

```

show interfaces switch interface-name vif vif-id ip ospfnetwork [ broadcast |
non-broadcast | point-to-multipoint | point-to-point ]

```

interface-name

The name of a data plane interface.

vif-id

Multinode. The identifier of a virtual interface in the range 1-4094.

network

The network type for this interface:

`broadcast`: The interface supports broadcast mode, such as on a LAN link.

`non-broadcast`: The interface does not support broadcast mode.

`point-to-point`: The interface supports point-to-point mode.

`point-to-multipoint`: The interface supports point-to-multipoint mode, such as a PPP interface or a point-to-point logical interface on Frame Relay.

The default type is `broadcast`.

Configuration mode

```

interfaces {
  switch {
    vif {
      ip {
        ospf {
          network
        }
      }
    }
  }
}

```

Use the `set` form of this command to set the OSPF network type for an interface.

Use the `delete` form of this command to delete the OSPF network type for an interface.

Use the `show` form of this command to display the OSPF network type for an interface.

interfaces switch vif ipv6 ospfv3 area

Assigns an OSPFv3 area to a IPv6 virtual interface.

```
set interfaces switch interface-namevif vif-id ipv6 ospfv3 area area-id
```

```
delete interfaces switch interface-namevif vif-id ipv6 ospfv3 area area-id
```

```
show interfaces switch interface-namevif vif-id ipv6 ospfv3 area area-id
interface-name ipv6 ospfv3 area area-id
```

interface-name

The name of an interface.

vif-id

Multinode. The identifier of a virtual interface in the range 1-4094.

area-id

OSPFv3 area identifier in the range 0-4294967295.

Configuration mode

```

interfaces {
  switch {
    vif {
      ipv6 {
        ospfv3 {
          area
        }
      }
    }
  }
}

```

```

    }
  }
}

```

Use the `set` form of this command to assign an OSPFv3 area to a IPv6 virtual interface.

Use the `delete` form of this command to delete an OSPFv3 area from a IPv6 virtual interface.

Use the `show` form of this command to display the OSPFv3 area for a IPv6 virtual interface.

interfaces switch vif ip ospfv3 fall-over bfd

Initiates a BFD session for all OSPFv3 neighbors on a virtual interface.

```
set interfaces switch interface-name vif vif-id ip ospfv3 fall-over bfd
```

```
delete interfaces switch interface-name vif vif-id ip ospfv3 fall-over bfd
```

```
show interfaces switch interface-name vif vif-id ip ospfv3 fall-over bfd
```

interface-name

The name of a data plane interface.

vif-id

Multinode. The ID of a virtual interface. The ID ranges from 1 through 4094.

Configuration mode

```

interfaces {
  switch {
    vif {
      ip {
        ospfv3 {
          fall-over {
            bfd
          }
        }
      }
    }
  }
}

```

Use the `set` form of this command to initiate a BFD session for all OSPFv3 neighbors on a virtual interface.

Use the `delete` form of this command to end a BFD session for all OSPFv3 neighbors on a virtual interface.

Use the `show` form of this command to display a BFD session for all OSPFv3 neighbors on a virtual interface.

interfaces switch vif ip ospfv3 network

Sets the OSPF network type for a virtual interface.

```
set interfaces switch interface-name vif vif-id ip ospfv3network [ broadcast |
non-broadcast | point-to-multipoint | point-to-point ]
```

```
delete interfaces switch interface-name vif vif-id ip ospfv3network [ broadcast
| non-broadcast | point-to-multipoint | point-to-point ]
```

```
show interfaces switch interface-name vif vif-id ip ospfv3network [ broadcast |
non-broadcast | point-to-multipoint | point-to-point ]
```

interface-name

The name of a data plane interface.

vif-id

Multinode. The ID of a virtual interface. The ID ranges from 1 through 4094.

network

The network type for this interface. The type is one of the following:

`broadcast`: The interface supports broadcast mode, such as on a LAN link.

`non-broadcast`: The interface does not support broadcast mode.

`point-to-point`: The interface supports point-to-point mode.

`point-to-multipoint`: The interface supports point-to-multipoint mode, such as a PPP interface or a point-to-point logical interface on Frame Relay.

The default type is `broadcast`.

Configuration mode

```
interfaces {
  switch {
    vif {
      ip {
        ospfv3 {
          network
        }
      }
    }
  }
}
```

Use the `set` form of this command to set the OSPF network type for an interface.

Use the `delete` form of this command to delete the OSPF network type for an interface.

Use the `show` form of this command to display the OSPF network type for an interface.

Chapter 13. Q-in-Q Interface

Q-in-Q interface overview

This section presents the following topics:

- [Q-in-Q interface features](#)
- [Packet processing using QoS schedulers](#)

With services such as metro Ethernet, the demand for VLAN IDs can increase beyond the original space limit of 4K. Q-in-Q enables the VLAN ID space to be extended from 4K to 16M, while also reducing the number of VLANs that must be managed.

The router Q-in-Q implementation is based on IEEE 802.1ad—multiple VLAN tags in an IP packet frame. The Q-in-Q interface is able to insert two VLAN tags into the Ethernet frame of a packet.

The Q-in-Q interface is supported on the data plane interface and must be configured as a virtual interface (vif). You configure the inner VLAN and the outer VLAN by using the VIF CLI commands, modify the data plane to be able to process packets with two VLAN tags, and then display the VLAN IDs of the VIF interface.

Only the outer VLAN tag is used when scheduling QoS packets. The outer VLAN tag can use EtherTypes 0x8100, 0x9100, 0x9200, 0x9300 and 0x88A8. The inner VLAN tag uses only 0x8100.

The outer VLAN is also referred to as “service VLAN” (S-VLAN) while the inner VLAN is called “customer VLAN” (C-VLAN).

When a Q-in-Q VIF is configured, two logical interfaces are created in the Linux kernel and the data plane—one representing the outer VLAN and the other representing the inner VLAN. In the data plane, the Q-in-Q logical interfaces are created after the corresponding netlink link messages are received.

Most forwarding features are performed based on the specifications of the interface regardless of whether the packet has a VLAN.

Q-in-Q interface features

The router implementation of the Q-in-Q interface supports the same set of features as that of the single VLAN VIF:

- IPv4 and IPv6
- Bridging
- QoS
- Firewall

- Routing protocols
- PBR policy

Packet processing using QoS schedulers

To manage traffic bandwidth, use the QoS scheduler to shape traffic based on Q-in-Q packets with VLAN IDs by using the following command:

- `set policy qos name policy-name shaper vlan outer-vid`

The VLAN ID is the outer VLAN ID of a Q-in-Q interface.

For more information on QoS, refer to *QoS Reference Guide*.

You can configure the aggregated scheduler to handle traffic that traverses the Q-in-Q interface.

- All inner VLANs share the same scheduler.
- Only the outer VLAN must be specified in the configuration.
- Use the following command to configure an aggregated scheduler:

```
set policy qos name policy-name shaper vlan outer-vid
```

- A subport is created for the outer VLAN and all inner VLANs under this outer VLAN share the same subport.

VLAN map

The three-bit priority of a VLAN tag in the frame can be used to map an incoming packet to a queue. For Q-in-Q packets, the priority applies only to the outer VLAN tag.

Configuring the Q-in-Q interface

To configure the VIF attributes and the QoS behavior, perform the following steps in configuration mode:

Step	Command
Create the VIF.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id vlan outer-vid vyatta@R1# set interfaces dataplane interface-name vif vif-id inner-vlan inner-vid</pre>
Assign the interface address.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id address address</pre>
Set the EtherType for VLAN packets.	<pre>vyatta@R1# set interfaces dataplane interface-name vlan-protocol ethertype</pre>

Step	Command
Add this interface to a bridge group.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id bridge-group ...</pre>
Define the IP routing parameters.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id ip ...</pre>
Define the IPv6 routing parameters.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id ipv6 ...</pre>
Define the firewall rules for this interface.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id firewall ...</pre>
Create an interface description.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id description ...</pre>
Define the interface MTU.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id mtu</pre>
Define the policy-based routing parameters.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id policy ...</pre>
Define the l2tp cross-connect interface parameters.	<pre>vyatta@R1# set interfaces dataplane interface-name vif vif-id xconnect ...</pre>
Configure the QoS behavior on the interface.	<pre>vyatta@R1# set policy qos name shaper vlan outer-vid</pre>

Disabling the interface

To disable the interface, use the following command:

```
vyatta@R1# set interfaces dataplane interface-name vif vif-id disable ...
```

Ignoring link state changes

To configure the system to ignore link state changes, use the following command:

```
vyatta@R1# set interfaces dataplane interface-name vif vif-id
disable-link-detect ...
```

Viewing interface queuing

To view the queuing of the logical interfaces, use the following command:

```
vyatta@R1# show queuing
```

Chapter 14. Q-in-Q Interface Commands

interfaces dataplane vif inner-vlan

Configures the ID of the inner VLAN for Q-in-Q.

```
set interfaces dataplane interface-name vif vif-id inner-vlan inner-vid
```

```
delete interfaces dataplane interface-name vif vif-id
```

```
show interfaces dataplane interface-name vif
```

dataplane *interface-name*

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

inner-vid

Inner VLAN ID.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      inner-vlan inner-vid
    }
  }
}
```

Use this command to configure the ID of the inner VLAN for Q-in-Q.

Use the `set` form of this command to configure the ID of the inner VLAN

Use the `delete` form of this command to delete the configuration.

Use the `show` form of this command to display the current settings.

interfaces dataplane vif vlan

Configures the ID of the outer VLAN for Q-in-Q.

```
set interfaces dataplane interface-name vif vif-id vlan outer-vid
```

```
delete interfaces dataplane interface-name vif vif-id
```

```
show interfaces dataplane interface-name vif vif-id
```

dataplane *interface-name*

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

Multinode. The ID of a vif. The ID ranges from 1 through 4094.

outer-vid

Outer VLAN ID.

Configuration mode

```
interfaces {
  dataplane interface-name {
    vif vif-id {
      vlan outer-vid
    }
  }
}
```

Use this command to configure the ID of the outer VLAN for Q-in-Q.

Use the `set` form of this command to configure the ID of the outer VLAN.

Use the `delete` form of this command to remove the ID settings.

Use the `show` form of this command to display the current settings.

interfaces dataplane vlan-protocol

Specifies the EtherType for VLAN frames.

```
set interfaces dataplane interface-name vlan-protocol ethertype
```

```
delete interfaces dataplane interface-name
```

```
show interfaces dataplane interface-name
```

The default EtherType is 0x8100.

dataplane *interface-name*

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

ethertype

Specifies the EtherType for the VLAN frames. The EtherType can be 0x88A8, 0x8100, 0x9100, 0x9200, or 0x9300.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vlan-protocol etherType
  }
}

```

Use this command to specify the EtherType for the VLAN frames.

Use the `set` form of this command to specify the EtherType.

Use the `delete` form of this command to restore the default EtherType.

Use the `show` form of this command to display the current EtherType.

policy qos name shaper vlan

Specifies the outer VLAN ID for a QoS policy.

```
set policy qos name policy-name shaper vlan outer-vid
```

```
delete policy qos name policy-name shaper vlan
```

```
show policy qos name policy-name shaper vlan
```

policy-name

Specifies the QoS policy.

outer-vid

Outer VLAN ID.

Configuration mode

```

policy {
  qos {
    name policy-name {
      shaper {
        vlan outer-vid
      }
    }
  }
}

```

Use this command to specify the outer VLAN ID for a QoS policy.

Use the `set` form of this command to specify the outer VLAN ID for the QoS policy.

Use the `delete` form of this command to return to the default VLAN IDs.

Use the `show` form of this command to display the current values.


Chapter 15. IP Unnumbered Interfaces

Overview

The IP unnumbered interfaces feature allows the use of the same IPv4 address on multiple interfaces, thus conserving IP addresses. The configuration of interfaces as unnumbered allows the enabling of IP processing on interfaces without assigning an IP address to them.

To configure an unnumbered interface, specify a donor interface from which the system can borrow an IP address. Then, configure a static-interface route to forward packets through the unnumbered interface.

Only data plane interfaces and virtual interfaces (vifs) can be configured as unnumbered interfaces. For IP packets that originate from routers, the unnumbered interfaces use IPv4 addresses from donor interfaces as preferred source addresses. The donor interfaces may be either data plane or loopback interfaces.

 **Note:** Numbered interfaces are interfaces that have their IP addresses configured with the `set interfaces` command, for example, `set interfaces dataplane dp0s192 address 1.1.1.1/24`.

Limitations of the feature

- The IP unnumbered interfaces feature does not support IPv6 unnumbered interfaces.
- Interfaces that are configured as unnumbered can neither take other addresses nor be part of a bridge.
- Only data plane interfaces and vifs can be configured as unnumbered interfaces.
- The preferred donor interfaces are loopback interfaces.
- The feature may affect Protocol Independent Multicast (PIM) and Internet Group Management Protocol (IGMP).

How unnumbered interfaces work

[Basic IP unnumbered interface configuration](#) illustrates how unnumbered interfaces work. A router called VR1 contains an interface named A that faces an interface named B on a router called VR2. A VLAN subinterface of A connects to a VLAN subinterface of B. The VLAN subinterface of A is configured as unnumbered.

On VR1, the administrator configures a static route through A toward the subnet in B. The static route makes the router take the manually configured unnumbered interface as the entry for routing, thereby, bypassing the need for a dynamic protocol to route traffic.

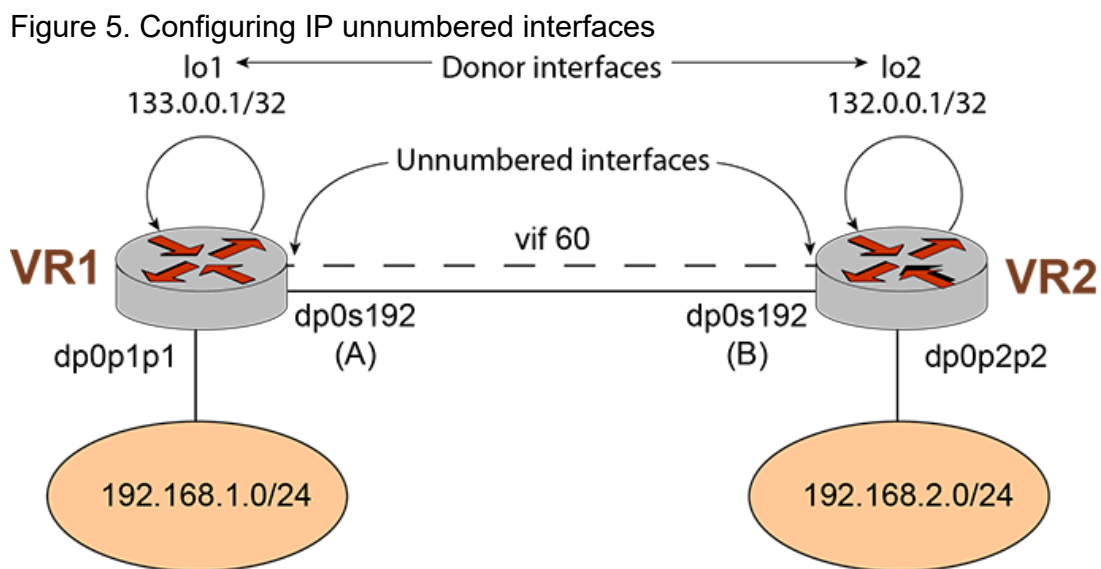
Unnumbered interface configuration example

This section includes the following topics:

- [Basic IP unnumbered interface configuration](#)

Basic IP unnumbered interface configuration

[Figure 5: Configuring IP unnumbered interfaces](#) presents a sample configuration of two routers with unnumbered interfaces and a static route.



To configure these interfaces, perform the following steps in configuration mode.

Table 10. Configuring IP unnumbered interfaces on VR1

Step	Command
Create a loopback interface, and configure the IP address.	<pre>vyatta@Router# set interfaces loopback lo1 address 133.0.0.1/32</pre>
Create the configuration node for the vif; assign the vif VLAN ID 60; and then assign the IP address for the vif.	<pre>vyatta@Router# set interfaces dataplane dp0s192 vif 60 vlan '60'</pre>
Borrow the IP address from a loopback interface.	<pre>vyatta@Router# set interfaces dataplane dp0s192 vif 60 ip unnumbered donor-interface lo1</pre>
Add a static-interface route to reach the other network.	<pre>vyatta@Router# set protocols static interface-route 132.0.0.1/32 next-hop-interface 'dp0s192.60'</pre>
Add a route to the destination network.	<pre>vyatta@Router# set protocol static route 192.168.2.0/0 nexthop 132.0.0.1</pre>
Commit the configuration.	<pre>vyatta@Router# commit</pre>
Verify whether the IP unnumbered support has been configured correctly.	<pre>vyatta@Router# show interfaces dataplane dp0s192 vif 60 ip { unnumbered { donor-interface lo1 } }</pre>

Table 10. Configuring IP unnumbered interfaces on VR1 (continued)

Step	Command
	} vlan 60

Table 11. Configuring IP unnumbered interfaces on VR2

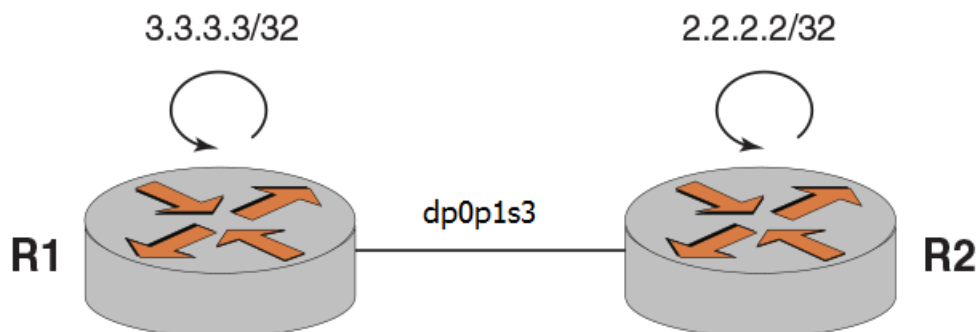
Step	Command
Create a loopback interface, and configure the IP address.	<pre>vyatta@Router# set interfaces loopback lo1 address 132.0.0.1/32</pre>
Create the configuration node for the vif; assign the vif VLAN ID 60; and then assign the IP address for the vif.	<pre>vyatta@Router# set interfaces dataplane dp0s192 vif 60 vlan '60'</pre>
Borrow the IP address from a loopback interface.	<pre>vyatta@Router# set interfaces dataplane dp0s192 vif 60 ip unnumbered donor-interface lo1</pre>
Add the static interface route to reach the other network.	<pre>vyatta@Router# set protocols static interface-route 133.0.0.1/32 next-hop-interface 'dp0s192.60'</pre>
Add a route to the destination network.	<pre>vyatta@Router# set protocol static route 192.168.1.0/0 nexthop 133.0.0.1</pre>
Commit the configuration.	<pre>vyatta@Router# commit</pre>
Verify whether the IP unnumbered support has been configured correctly.	<pre>vyatta@Router# show interfaces dataplane dp0s192 vif 60 ip { unnumbered { donor-interface lo1 } } vlan 60</pre>

Configuring BGP on an unnumbered interface

You can configure the BGP protocol on an unnumbered interface.

This section presents an example configuration of eBGP with an unnumbered Interface. The example is based on the following figure.

Figure 6. Configuring BGP on an unnumbered interface



To configure BGP on an unnumbered interface, perform the following steps in configuration mode.

Table 12. Configuring an IP unnumbered interface on router R1

Step	Command
Add an interface address.	<code>vyatta@R1# set interfaces dataplane dp0pls2 address 133.0.1.1/24</code>
Add an unnumbered interface address.	<code>vyatta@R1# set interfaces dataplane dp0pls3 ip unnumbered donor-interface dp0pls2</code>
Set the identifier of the BGP router to that of the router address.	<code>vyatta@R1# set protocols bgp parameters router-id 3.3.3.3</code>
Configure the eBGP neighbor peering.	<code>vyatta@R1# set protocols bgp 100 neighbor 133.0.1.2 address-family ipv4-unicast</code>
Set the eBGP remote AS.	<code>vyatta@R1# set protocols bgp 100 neighbor 133.0.1.2 remote-as 200</code>
Add a static interface route to reach the neighbor.	<code>vyatta@R1# set protocols static interface-route 133.0.1.2/32 next-hop-interface dp0pls3 bgp network point-to-point</code>
Verify that the BGP protocol is configured with an unnumbered interface on a point-to-point interface type only.	<pre>vyatta@R1# show interfaces interfaces { dataplane dp0pls2 { address 133.0.1.1/24 } dataplane dp0pls3 { ip { unnumbered { donor-interface dp0pls2 } } } }</pre>

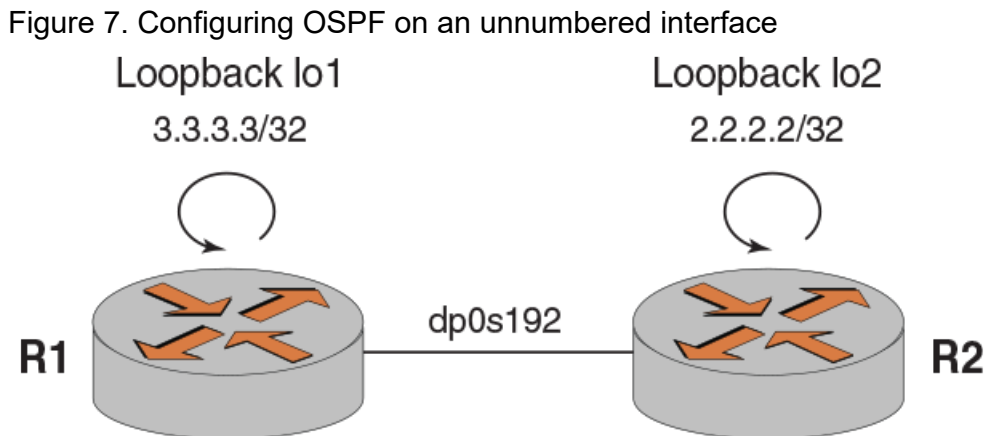
Table 13. Configuring an IP unnumbered interface on router R2

Step	Command
Set the interface address.	<code>vyatta@R1# set interfaces dataplane dp0pls2 address 133.0.1.2/24</code>
Set the unnumbered interface address.	<code>vyatta@R1# set interfaces dataplane dp0pls3 ip unnumbered donor-interface dp0pls2</code>
Set the identifier of the BGP router to that of the router address.	<code>vyatta@R1# set protocols bgp parameters router-id 3.3.3.3</code>
Set the eBGP neighbor peering.	<code>vyatta@R1# set protocols bgp 200 neighbor 133.0.1.1 address-family ipv4-unicast</code>
Set the eBGP remote AS.	<code>vyatta@R1# set protocols bgp 200 neighbor 133.0.1.1 remote-as 100</code>
Add a static interface route to reach the neighbor.	<code>vyatta@R1# set protocols static interface-route 133.0.1.1/32 next-hop-interface dp0pls3</code>
Verify that the BGP protocol is configured with an unnumbered interface on a point-to-point interface type only.	<pre>vyatta@R1# show interfaces interfaces { dataplane dp0pls2 { address 133.0.1.2/24 } dataplane dp0pls3 { ip { unnumbered { donor-interface dp0pls2 } } } }</pre>

Configuring OSPF on an unnumbered interface

You can configure the OSPF protocol on an unnumbered interface.

This section presents an example configuration of OSPF with an unnumbered Interface. The example is based on the following figure.



To configure OSPF on an unnumbered interface, perform the following steps in configuration mode.

Table 14. Configuring an IP unnumbered interface on router R1

Step	Command
Create a loopback interface for router R1.	<pre>vyatta@R1#set interfaces loopback lo1 address 3.3.3.3/32</pre>
Add a static interface route to reach the other network.	<pre>vyatta@R1# set protocols static interface-route 192.169.2.0/24 next-hop-interface dp0s192</pre>
Add a route to the destination network.	<pre>vyatta@R1# set protocols static route 192.169.2.0/24 next-hop 3.3.3.3</pre>
Set the identifier of the OSPF router to that of the loopback address.	<pre>vyatta@R1#set protocols ospf parameters router-id 3.3.3.3</pre>
Configure the donor interface.	<pre>vyatta@R1#set interfaces unnumbered donor-interface lo1</pre>
Advertise to the 3.3.3.0/24 network.	<pre>vyatta@R1#set protocols ospf area 0.0.0.0 network 3.3.3.0/24</pre>
Configure the point-to-point protocol on the OSPF network.	<pre>vyatta@R1#set interfaces dataplane dp0s192 ip ospf network point-to-point</pre>
Verify that the OSPF protocol is configured with an unnumbered interface on a point-to-point interface type only.	<pre>vyatta@R1## show interfaces interfaces { ospf { network point-to-point } unnumbered { donor-interface lo1 } } loopback lo1 { address 3.3.3.3/32 } }</pre>

Table 15. Configuring an IP unnumbered interface on router R2

Step	Command
Create a loopback interface for router R2.	<pre>vyatta@R1#set interfaces loopback lo1 address 2.2.2.2/32</pre>
Add a static interface route to reach the other network.	<pre>vyatta@R1# set protocols static interface-route 192.169.2.0/24 next-hop-interface dp0s192</pre>
Add a route to the destination network.	<pre>vyatta@R1# set protocols static route 192.169.2.0/24 next-hop 2.2.2.2</pre>
Set the identifier of the OSPF router to that of the loopback address.	<pre>vyatta@R1#set protocols ospf parameters router-id 3.3.3.3</pre>
Configure the donor interface.	<pre>vyatta@R1#set interfaces unnumbered donor-interface lo1</pre>
Advertise to the 2.2.2.0/24 network.	<pre>vyatta@R1# set protocols ospf area 0.0.0.0 network 2.2.2.0/24</pre>
Configure the point-to-point protocol on the OSPF network.	<pre>vyatta@R1#set interfaces dataplane dp0s192 ip ospf network point-to-point</pre>
Verify that the OSPF protocol is configured with an unnumbered interface on a point-to-point interface type only.	<pre>vyatta@R1## show interfaces interfaces { ospf { network point-to-point } unnumbered { donor-interface lo1 } } loopback lo1 { address 2.2.2.2/32 } }</pre>

Chapter 16. IP Unnumbered Interfaces Command

interfaces dataplane vif ip unnumbered donor-interface

Configures the virtual interface of the specified data plane as an IP unnumbered interface with the donor interface and preferred address.

```
set interfaces dataplane interface-name vif vif-id ip unnumbered donor-  
interface donor_interface-name [ preferred-address ip-address ]
```

```
delete interfaces dataplane interface-name vif vif-id ip unnumbered
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

vif-id

The ID of a virtual interface. The ID ranges from 1 through 4094.

donor-interface-name

The name of the donor data plane or loopback interface.

preferred-address *ip-address*

Specifies the borrowed preferred address to use from the donor interface.

Configuration mode

```
interfaces {  
    dataplane interface-name {  
        vif vif-id {  
            ip {  
                unnumbered {  
                    donor-interface donor-interface-name {  
                        preferred-address ip-address  
                    }  
                }  
            }  
        }  
    }  
}
```


Use this command to configure interfaces as unnumbered interfaces.

Use the `set` form of this command to configure a data plane interface as an unnumbered interface. The preferred address of the donor interface must be one of the configured addresses on the interface.

Use the `delete` form of this command to remove an unnumbered interface. To remove the preferred address of the donor interface, use the following command:


```
delete interfaces dataplane interface-name vif vif-id ip unnumbered donor-interface  
donor_interface-name preferred-address
```

This command has no `show` form. To display information about unnumbered data plane interfaces and donor interfaces, use [show interfaces dataplane](#) and [show interfaces loopback](#).

 **Note:** To change a donor interface, first set a new donor interface, then delete the one that you want to change.

Chapter 17. L2TPv3 Data Plane Interfaces

This chapter describes basic configuration for Layer 2 Tunnel Protocol Version 3 (L2TPv3) tunnel interfaces.

 **Note:** Layer 2 bridging is supported on data plane interfaces. Commands for configuring bridge groups are described in *Bridging Configuration Guide*.

Overview

L2TPv3 is a protocol for carrying Layer 2 (L2) frames (such as Ethernet, Frame Relay, ATM, and PPP) across an IP network. The L2TPv3 RFC specifies both control-plane and data plane functions. The L2TPv3 control plane is primarily responsible for the setup, management, and maintenance of individual L2TP sessions, while the data plane function deals with encapsulation, security, and forwarding procedures.

In L2TPv3, end points are of two types: L2TP Access Concentrator (LAC) and L2TP Network Server (LNS). In L2TP LAC, a one-to-one mapping exists between a L2 attachment circuit (such as an Ethernet port or a VLAN) and an L2TP session. A LAC is simply a cross-connect device that transparently stitches the L2 frames between an Ethernet port or a VLAN and an L2TP session.

L2TP LNS, on the other hand, terminates L2 services and places the L2TP session into a forwarding domain (such as an L2 bridge domain or an L3 routing instance). LNS makes forwarding decisions based on forwarding information base (FIB) lookups.

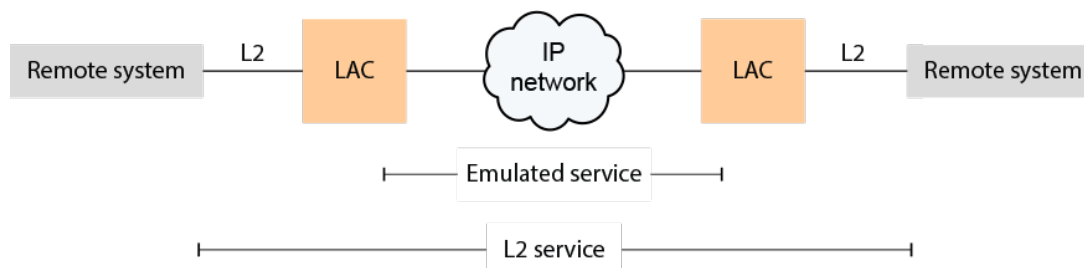
L2TP operates in three tunneling reference models:

- LAC-to-LAC
- LAC-to-LNS
- LNS-to-LNS

LAC-to-LAC tunneling reference model

In the LAC-to-LAC reference model, on both ends of the L2TP tunnel, LAC transparently stitches L2 frames between an Ethernet port or a VLAN and an L2TP session. The remote router performs the routing and bridging functions.

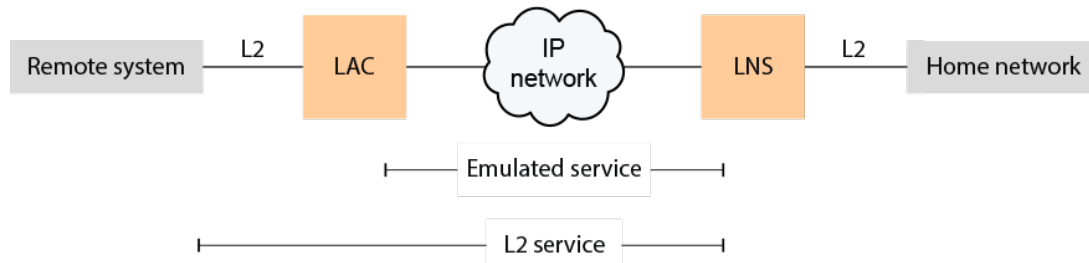
Figure 8. LAC-to-LAC tunneling reference model



LAC-to-LNS tunneling reference model

In the LAC-to-LNS reference model, on one side of the L2TP tunnel, LAC cross-connects L2 frames into an L2TP tunnel; while on the other side, LNS performs bridging or routing functions between the home network and the corresponding L2TP session.

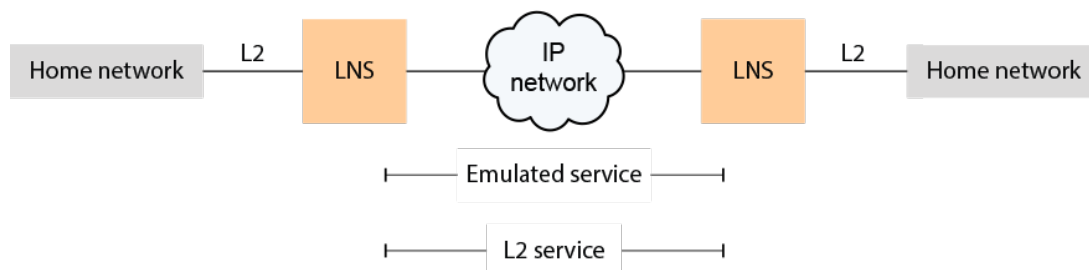
Figure 9. LAC-to-LNS tunneling reference model



LNS-to-LNS tunneling reference model

In the LNS-to-LNS reference model, on both ends of the L2TP tunnel, L2TP devices terminate the L2 service and perform routing or bridging functions between the home network and L2TP tunnel.

Figure 10. LNS-to-LNS tunneling reference model



Configuration commands

The [Commands](#) section describes the L2TPv3 tunnel and LAC configuration commands. The section also describes the L2TPv3 show commands.

For LNS commands, the L2TPv3 feature supports the following command options for the `set interfaces l2tpeth ltpN` command. You can use these command options to configure LNS tunnels. These command options are similar to the options for the `set interfaces dataplane dp_interface` command.

Table 16. LNS command options

Command option	Description
<code>address</code>	Specifies the IPv4 or IPv6 address of a tunnel interface.
<code>bridge-group</code>	Adds and configures a bridge group for a tunnel interface.

Table 16. LNS command options (continued)

Command option	Description
<code>description</code>	Provides a description of a tunnel interface.
<code>disable</code>	Disables a tunnel interface.
<code>firewall</code>	Specifies firewall options.
<code>ip</code>	Specifies IPv4 parameters.
<code>ipv6</code>	Specifies IPv6 parameters.
<code>mtu</code>	Specifies the maximum transmission unit (MTU).
<code>policy</code>	Specifies the policy settings for a tunnel interface.
<code>vif</code>	Specifies the identifier of a VIF interface for a tunnel.

Before configuring an L2TP tunnel

To configure an L2TP tunnel, a route to the remote peer must exist. Additionally, if the remote peer is in a different subnet, IP Infusion Inc. recommends adding a static route to that peer before configuring the L2TP tunnel.

L2TPv3 configuration examples

This section presents the following topics:

- [Before you begin](#)
- [Configuring a LAC-to-LAC IPv4 tunnel](#)
- [Configuring a LAC-to-LAC IPv6 tunnel](#)
- [Configuring an LNS-to-LNS IPv4 tunnel](#)
- [Configuring an LNS-to-LNS tunnel with BGP routes](#)
- [Configuring an LNS-to-LNS tunnel with OSPF routes](#)
- [Configuring an LNS-to-LNS tunnel with RIP routes](#)

Before you begin

It is assumed for the examples in the following sections that the data plane interfaces have already been configured.

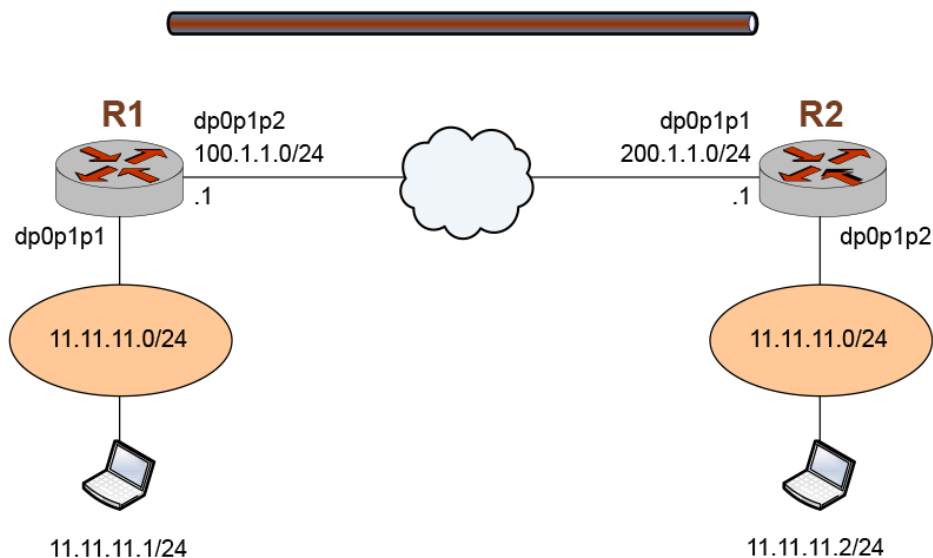
Configuring a LAC-to-LAC IPv4 tunnel

This section shows the topology of a sample L2TPv3 LAC-to-LAC IPv4 tunnel configuration. To configure this sample LAC-to-LAC tunnel, perform the following steps in configuration mode:

1. Configure the L2TPv3 tunnel and LAC cross-connect settings on the R1 router.

- a. Configure the L2TPv3 tunnel settings ([Table 17: Configuring the L2TPv3 settings for a LAC-to-LAC IPv4 tunnel on R1](#)).
 - b. Configure the LAC cross-connect settings ([Table 18: Configuring the LAC cross-connect settings on R1](#)).
2. Configure the L2TPv3 tunnel and LAC cross-connect settings on the R2 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 19: Configuring the L2TPv3 tunnel settings on R2](#)).
 - b. Configure the LAC cross-connect settings ([Table 20: Configuring the LAC cross-connect settings on R2](#)).

Figure 11. Sample LAC-to-LAC IPv4 tunnel
LAC-to-LAC Tunnel



[Table 17: Configuring the L2TPv3 settings for a LAC-to-LAC IPv4 tunnel on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 17. Configuring the L2TPv3 settings for a LAC-to-LAC IPv4 tunnel on R1

Step	Command
Specify IP as the encapsulation method to use for the ltp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp20 l2tp-session encapsulation 'ip'</pre>
Specify the cookie string used by R1 (the local router).	<pre>vyatta@vyatta# set interfaces l2tpeth ltp20 l2tp-session local-cookie '12345678'</pre>
Specify the local IP address of the ltp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp20 l2tp-session local-ip '100.1.1.0/24'</pre>
Specify the local session identifier of the ltp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp20 l2tp-session local-session-id '88776655'</pre>

Step	Command
Specify the local UDP port of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session local-udp-port '65535'</pre>
Specify the cookie string used by R2 (the remote router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-cookie '87654321'</pre>
Specify the remote IP address of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-ip '200.1.1.0/24'</pre>
Specify the remote session identifier of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-session-id '88776655'</pre>
Specify the remote UDP port of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-udp-port '65534'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the configuration.	<pre>vyatta@vyatta:~\$show interfaces interfaces { dataplane dp0p160p1 { address 10.18.179.216/21 } dataplane dp0p192p1 { address 100.1.1.1/24 } dataplane dp0p1p1 { } l2tpeth l2tp20 { l2tp-session { encapsulation udp local-cookie abcd1234 local-ip 100.1.1.1 local-session-id 11223344 local-udp-port 60000 remote-cookie 1234abcd remote-ip 200.1.1.1 remote-session-id 44332211 remote-udp-port 60000 } } loopback lo }</pre>

[Table 18: Configuring the LAC cross-connect settings on R1](#) shows how to configure the LAC cross-connect settings on R1.

Table 18. Configuring the LAC cross-connect settings on R1

Step	Command
Specify the cross-connect parameters of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces dataplane dp0p1p1 xconnect l2tpeth 'l2tp20'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the configuration.	<pre>vyatta@vyatta:~\$ show interfaces interfaces { dataplane dp0p160p1 { address 10.18.179.216/21 } dataplane dp0p192p1 { address 100.1.1.1/24 } dataplane dp0p1p1 { xconnect { l2tpeth l2tp20 } } }</pre>

Step	Command
	<pre> dataplane dp0p256p1 l2tpeth lttp20 { l2tp-session { encapsulation udp local-cookie abcd1234 local-ip 100.1.1.1 local-session-id 11223344 local-udp-port 60000 remote-cookie 1234abcd remote-ip 200.1.1.1 remote-session-id 44332211 remote-udp-port 60000 } } loopback lo </pre>

[Table 19: Configuring the L2TPv3 tunnel settings on R2](#) shows how to configure the L2TPv3 tunnel interface on R2.

Table 19. Configuring the L2TPv3 tunnel settings on R2

Step	Command
Specify IP as the encapsulation method to use for the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session encapsulation 'ip'</pre>
Specify the cookie string used by R2 (the local router).	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session local-cookie '87654321'</pre>
Specify the local IP address of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session local-ip '200.1.1.1'</pre>
Specify the local session identifier of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session local-session-id '88776655'</pre>
Specify the local UDP port of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session local-udp-port '65534'</pre>
Specify the cookie string used by R1 (the remote router).	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session remote-cookie '12345678'</pre>
Specify the remote IP address of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session remote-ip '100.1.1.1'</pre>
Specify the remote session identifier of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session remote-session-id '88776655'</pre>
Specify the remote UDP port of the lttp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth lttp20 l2tp-session remote-udp-port '65535'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the configuration.	<pre>vyatta@vyatta:~\$ show interfaces interfaces { dataplane dp0p160p1 { address 10.18.179.217/21 } dataplane dp0p192p1 { address 200.1.1.1/24 } dataplane dp0p1p1 { } l2tpeth lttp20 { l2tp-session { encapsulation udp local-cookie 1234abcd local-ip 200.1.1.1 local-session-id 44332211 local-udp-port 60000 } } } </pre>

Step	Command
	<pre> remote-cookie abcd1234 remote-ip 100.1.1.1 remote-session-id 11223344 remote-udp-port 60000 } } loopback lo } </pre>

[Table 20: Configuring the LAC cross-connect settings on R2](#) shows how to configure the LAC cross-connect settings on R2.

Table 20. Configuring the LAC cross-connect settings on R2

Step	Command
Specify the cross-connect parameters of the l2tp20 L2TPv3 tunnel interface.	<pre> vyatta@vyatta# set interfaces dataplane dp0p1p1 xconnect l2tpeth 'lttp20' </pre>
Commit the configuration.	<pre> vyatta@vyatta# commit </pre>
Save the configuration.	<pre> vyatta@vyatta# save </pre>
Show the configuration.	<pre> vyatta@vyatta:~\$ show interfaces interfaces { dataplane dp0p160p1 { address 10.18.179.217/21 } dataplane dp0p192p1 { address 200.1.1.1/24 } dataplane dp0p1p1 { xconnect { l2tpeth lttp20 } } l2tpeth lttp20 { l2tp-session { encapsulation udp local-cookie 1234abcd local-ip 200.1.1.1 local-session-id 44332211 local-udp-port 60000 remote-cookie abcd1234 remote-ip 100.1.1.1 remote-session-id 11223344 remote-udp-port 60000 } } } loopback lo } </pre>

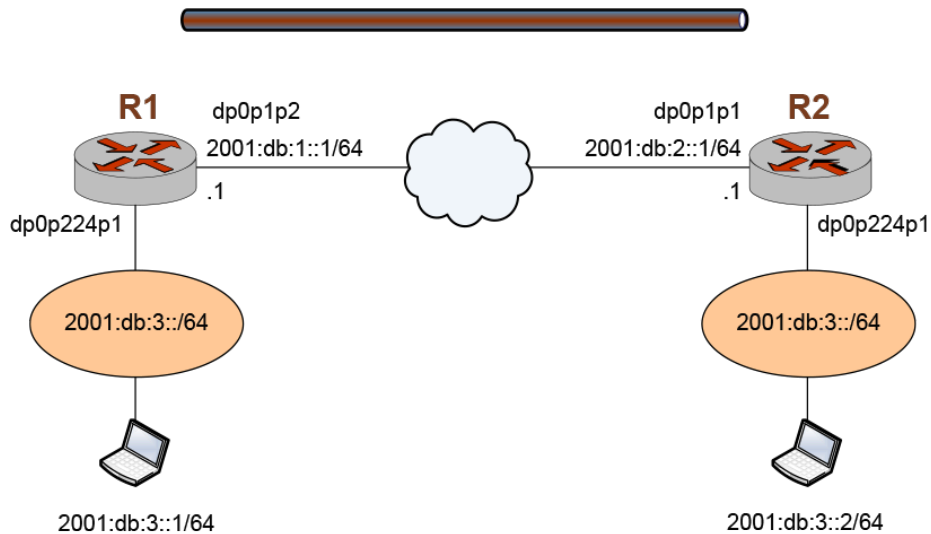
Configuring a LAC-to-LAC IPv6 tunnel

[Figure 12: Sample LAC-to-LAC IPv6 tunnel](#) shows the topology of a sample L2TPv3 LAC-to-LAC IPv6 tunnel configuration. To configure this sample LAC-to-LAC tunnel, perform the following steps in configuration mode:

1. Configure the L2TPv3 tunnel and LAC cross-connect settings on the R1 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 21: Configuring the L2TPv3 settings for a LAC-to-LAC IPv6 tunnel on R1](#)).
 - b. Configure the LAC cross-connect settings ([Table 22: Configuring the LAC cross-connect settings on R1](#)).
2. Configure the L2TPv3 tunnel and LAC cross-connect settings on the R2 router.

- a. Configure the L2TPv3 tunnel settings ([Table 23: Configuring the L2TPv3 tunnel settings on R2](#)).
- b. Configure the LAC cross-connect settings ([Table 24: Configuring the LAC cross-connect settings on R2](#)).

Figure 12. Sample LAC-to-LAC IPv6 tunnel
LAC-to-LAC Tunnel



[Table 21: Configuring the L2TPv3 settings for a LAC-to-LAC IPv6 tunnel on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 21. Configuring the L2TPv3 settings for a LAC-to-LAC IPv6 tunnel on R1

Step	Command
Specify IP as the encapsulation method to use for the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session encapsulation 'ip'</pre>
Specify the cookie string used by R1 (the local router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session local-cookie '12345678'</pre>
Specify the local IP address of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session local-ip '2001:db:1::1'</pre>
Specify the local session identifier of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session local-session-id '88776655'</pre>
Specify the local UDP port of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session local-udp-port '65535'</pre>
Specify the cookie string used by R2 (the remote router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-cookie '87654321'</pre>
Specify the remote IP address of the l2tp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp20 l2tp-session remote-ip '2001:db:2::1'</pre>

Step	Command
Specify the remote session identifier of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-session-id '88776655'</pre>
Specify the remote UDP port of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-udp-port '65534'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 22: Configuring the LAC cross-connect settings on R1](#) shows how to configure the LAC cross-connect settings on R1.

Table 22. Configuring the LAC cross-connect settings on R1

Step	Command
Specify the cross-connect parameters of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces dataplane dp0p224p1 xconnect l2tpeth 'l2tpv3'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 23: Configuring the L2TPv3 tunnel settings on R2](#) shows how to configure the LAC cross-connect settings on R2.

Table 23. Configuring the L2TPv3 tunnel settings on R2

Step	Command
Specify IP as the encapsulation method to use for the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session encapsulation 'ip'</pre>
Specify the cookie string used by R2 (the local router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session local-cookie '87654321'</pre>
Specify the local IP address of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session local-ip '2001:db:2::1'</pre>
Specify the local session identifier of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session local-session-id '88776655'</pre>
Specify the local UDP port of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session local-udp-port '65534'</pre>
Specify the cookie string used by R1 (the remote router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-cookie '12345678'</pre>
Specify the remote IP address of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-ip '2001:db:1::1'</pre>
Specify the remote session identifier of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-session-id '88776655'</pre>
Specify the remote UDP port of the l2tpv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tpv3 l2tp-session remote-udp-port '65534'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 24: Configuring the LAC cross-connect settings on R2](#) shows how to configure the LAC cross-connect settings on R2.

Table 24. Configuring the LAC cross-connect settings on R2

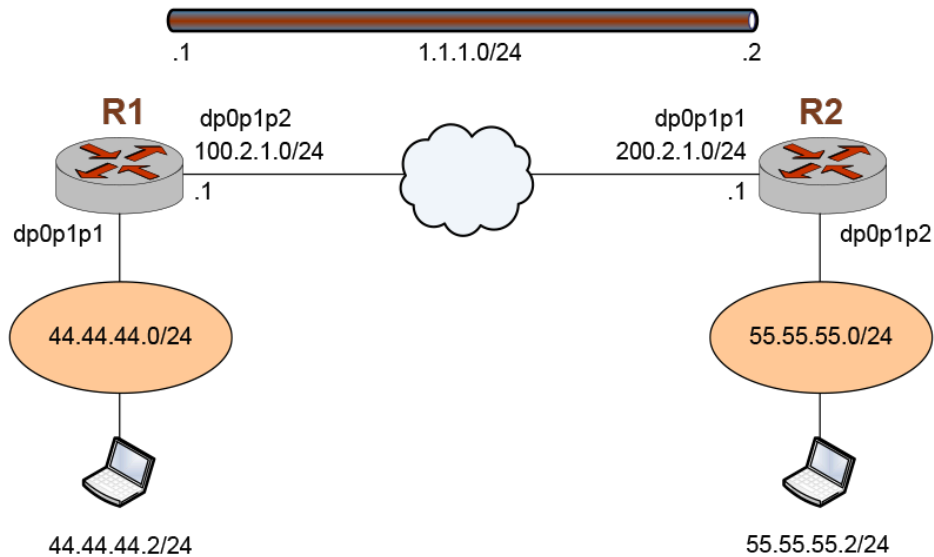
Step	Command
Specify the cross-connect parameters of the ltp20 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces dataplane dp0p224p1 xconnect l2tpeth 'ltp20'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

Configuring an LNS-to-LNS IPv4 tunnel

This section shows the topology of a sample L2TPv3 LNS-to-LNS IPv4 tunnel configuration. To configure this sample LNS-to-LNS tunnel, perform the following steps in configuration mode.

1. Configure the L2TPv3 tunnel settings and the LNS settings on the R1 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 25: Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R1](#)).
 - b. Configure the LNS settings ([Table 26: Configuring the next hop for 55.55.55.0/24 traffic on R1](#)).
2. Configure the L2TPv3 tunnel settings and LNS settings on the R2 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 27: Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R2](#)).
 - b. Configure the LNS settings ([Table 28: Configuring the next hop for 44.44.44.0/24 traffic on R2](#)).

Figure 13. Sample LNS-to-LNS IPv4 tunnel
**LNS-to-LNS Tunnel
 with Static Routes**



[Table 25: Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 25. Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R1

Step	Command
Specify IP as the encapsulation method to use for the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-ip '100.2.1.1'</pre>
Specify the local session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-session-id '4294967295'</pre>
Specify the remote IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-ip '200.2.1.1'</pre>
Specify the remote session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-session-id '4000000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 address '1.1.1.1/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 26: Configuring the next hop for 55.55.55.0/24 traffic on R1](#) shows how to configure on R1 the next hop for traffic that is destined for the 55.55.55.0/24 network.

Table 26. Configuring the next hop for 55.55.55.0/24 traffic on R1

Step	Command
Specify the IP address of the interface at the other end of the tunnel (1.1.1.2) as the next hop for traffic that is destined for the 55.55.55.0/24 network. For more information about this command, refer to <i>Basic Routing Configuration Guide</i> .	<pre>vyatta@vyatta# set protocols static route 55.55.55.0/24 next-hop '1.1.1.2'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the route.	<pre>vyatta@vyatta# run show ip route 55.55.55.2 Routing entry for 55.55.55.0/24 Known via "static", distance 1, metric 0, best * 1.1.1.2, via l2tp1111</pre>
Exit the configuration mode.	<pre>vyatta@vyatta# exit</pre>
Show the configuration.	<pre>vyatta@vyatta:~\$ show interfaces l2tpeth l2tp1111 address 1.1.1.1/24 l2tp-session { encapsulation ip local-ip 100.2.1.1 local-session-id 4294967295 remote-ip 200.2.1.1 remote-session-id 4000000000 }</pre>

[Table 27: Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R2](#) shows how to configure the L2TPv3 tunnel interface on R2.

Table 27. Configuring the L2TPv3 settings for an LNS-to-LNS IPv4 tunnel on R2

Step	Command
Specify IP as the encapsulation method to use for the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-ip '200.2.1.1'</pre>
Specify the local session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-session-id '4294967295'</pre>
Specify the remote IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-ip '100.2.1.1'</pre>
Specify the remote session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-session-id '4000000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 address '1.1.1.2/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 28: Configuring the next hop for 44.44.44.0/24 traffic on R2](#) shows how to configure on R2 the next hop for traffic that is destined for the 44.44.44.0/24 network.

Table 28. Configuring the next hop for 44.44.44.0/24 traffic on R2

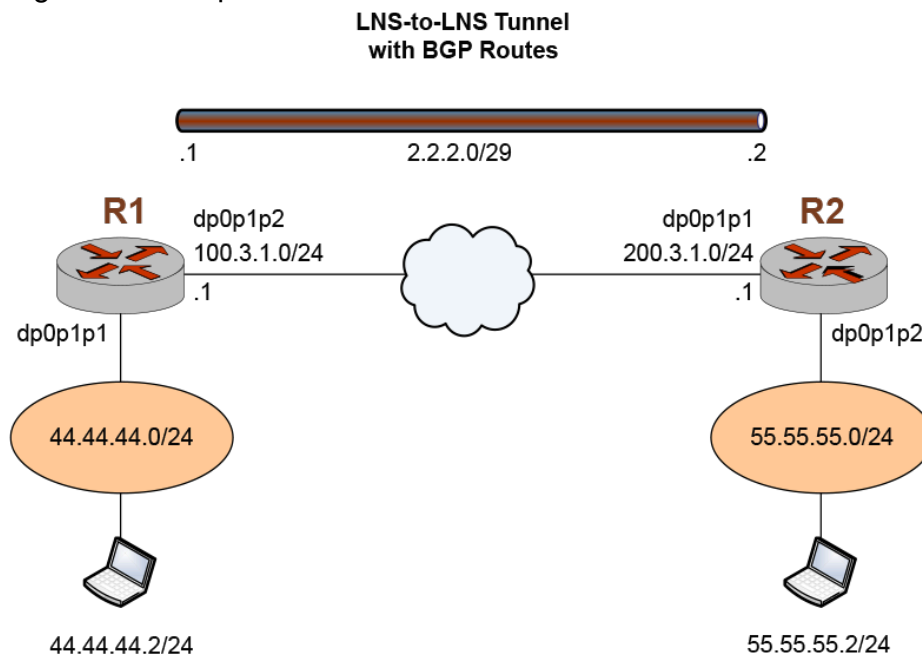
Step	Command
Specify the IP address of the interface at the other end of the tunnel (1.1.1.1) as the next hop for traffic that is destined for the 44.44.44.0/24 network. For more information about this command, refer to <i>Basic Routing Configuration Guide</i> .	<pre>vyatta@vyatta# set protocols static route 44.44.44.0/24 next-hop '1.1.1.1'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

Configuring an LNS-to-LNS tunnel with BGP routes

This section shows the topology of a sample L2TPv3 LNS-to-LNS IPv4 tunnel with Border Gateway Protocol (BGP) configuration. To configure this sample LNS-to-LNS tunnel, perform the following steps in configuration mode.

1. Configure the L2TPv3 tunnel settings and the LNS settings on the R1 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 29: Configuring the L2TPv3 tunnel interface settings on R1](#)).
 - b. Configure the BGP settings ([Table 30: Configuring the BGP settings of the LNS-to-LNS tunnel on R1](#)).
2. Configure the L2TPv3 tunnel settings and LNS settings on the R2 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 31: Configuring the L2TPv3 tunnel interface settings on R2](#)).
 - b. Configure the BGP settings ([Table 32: Configuring the BGP settings of the LNS-to-LNS tunnel on R2](#)).

Figure 14. Sample LNS-to-LNS IPv4 tunnel with BGP routes



[Table 29: Configuring the L2TPv3 tunnel interface settings on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 29. Configuring the L2TPv3 tunnel interface settings on R1

Step	Command
Specify IP as the encapsulation method to use for the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session local-ip '100.3.1.1'</pre>
Specify the local session identifier of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session local-session-id '1000000'</pre>
Specify the remote IP address of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session remote-ip '200.3.1.1'</pre>
Specify the remote session identifier of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session remote-session-id '1000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 address '2.2.2.1/29'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 30: Configuring the BGP settings of the LNS-to-LNS tunnel on R1](#) shows how to configure the BGP settings for the tunnel on R1. For more information about configuring BGP settings, refer to *BGP Reference Guide*.

Table 30. Configuring the BGP settings of the LNS-to-LNS tunnel on R1

Step	Command
Specify the 44.44.44.0/24 network as the network to be advertised by the BGP routing process.	<pre>vyatta@vyatta# set protocols bgp 100 address-family ipv4-unicast network '44.44.44.0/24'</pre>
Direct the router to store received routing updates.	<pre>vyatta@vyatta# set protocols bgp 100 neighbor 2.2.2.2 address-family ipv4-unicast soft-reconfiguration 'inbound'</pre>
Specify 200 as the autonomous system (AS) number of the neighbor.	<pre>vyatta@vyatta# set protocols bgp 100 neighbor 2.2.2.2 remote-as '200'</pre>
Set the fixed BGP router ID for the router to 2.2.2.1. This setting overrides the automatic ID selection process.	<pre>vyatta@vyatta# set protocols bgp 100 parameters router-id '2.2.2.1'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the l2tpeth configuration.	<pre>vyatta@vyatta# run show l2tpeth L2tpeth interface: lttp2222 Port: 0, ifIndex: 19 Mac: 96:89:f9:e7:6e:d1 Addresses: inet 2.2.2.1/29, broadcast 2.2.2.7 inet6 fe80::9489:f9ff:fee7:6ed1/64, scope Link L2TPv3: Encap IP Local Address: 100.3.1.1 Peer Address: 200.3.1.1 Local Session: 2000000 Peer Session: 2000000 Local Cookie: 00 Peer Cookie: 00 Statistics: Input bytes : 191229 Output bytes : 144752 Input packets : 2314 Output packets : 1143 Input multicast : 107</pre>
Show the configuration information for lttp2222.	<pre>vyatta@vyatta# run show interfaces lttp2222 lttp2222: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1488 qdisc pfifo_fast state UNKNOWN group default qlen 1000 link/ether 96:89:f9:e7:6e:d1 brd ff:ff:ff:ff:ff:ff inet 2.2.2.1/29 brd 2.2.2.7 scope global lttp2222 valid_lft forever preferred_lft forever inet6 fe80::9489:f9ff:fee7:6ed1/64 scope link valid_lft forever preferred_lft forever RX: bytes packets errors dropped overrun mcast 0 0 0 0 TX: bytes packets errors dropped carrier collisions 0 0 1118 0 0</pre>
Show a summary of the BGP settings.	<pre>vyatta@vyatta# run show ip bgp summary BGP router identifier 2.2.2.1, local AS number 100 BGP table version is 192 2 BGP AS-PATH entries 0 BGP community entries IPv4 Unicast address family: Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down Statd 2.2.2.2 4 200 383 383 192 0 0 03:10:16 1 Total number of neighbors 1</pre>
Show the BGP settings.	<pre>vyatta@vyatta# run show ip bgp BGP table version is 192, local router ID is 2.2.2.1 Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, l - labeled S Stale</pre>

Step	Command
	<pre>Origin codes: i - IGP, e - EGP, ? - incomplete Network Next Hop Metric LocPrf Weight Path *> 44.44.44.0/24 0.0.0.0 100 32768 i *> 55.55.55.0/24 2.2.2.2 0 0 200 i Total number of prefixes 2</pre>
Show the route information for the 55.55.55.2 system.	<pre>vyatta@vyatta# run show ip route 55.55.55.2 Routing entry for 55.55.55.0/24 Known via "bgp", distance 20, metric 0, best Last update 00:04:06 ago * 2.2.2.2, via l2tp2222</pre>

[Table 31: Configuring the L2TPv3 tunnel interface settings on R2](#) shows how to configure the L2TPv3 tunnel interface on R2.

Table 31. Configuring the L2TPv3 tunnel interface settings on R2

Step	Command
Specify IP as the encapsulation method to use for the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session local-ip '200.3.1.1'</pre>
Specify the local session identifier of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session local-session-id '2000000'</pre>
Specify the remote IP address of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session remote-ip '100.3.1.1'</pre>
Specify the remote session identifier of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session remote-session-id '2000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 address '2.2.2.2/29'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 32: Configuring the BGP settings of the LNS-to-LNS tunnel on R2](#) shows how to configure the BGP settings for the tunnel on R2. For more information about configuring BGP settings, refer to *BGP Reference Guide*.

Table 32. Configuring the BGP settings of the LNS-to-LNS tunnel on R2

Step	Command
Specify the 55.55.55.0/24 network as the network to be advertised by the BGP routing process.	<pre>vyatta@vyatta# set protocols bgp 100 address-family ipv4-unicast network '55.55.55.0/24'</pre>
Direct the router to store received routing updates.	<pre>vyatta@vyatta# set protocols bgp 100 neighbor 2.2.2.1 address-family ipv4-unicast soft-reconfiguration 'inbound'</pre>
Specify 100 as the AS number of the neighbor.	<pre>vyatta@vyatta# set protocols bgp 100 neighbor 2.2.2.1 remote-as '200'</pre>

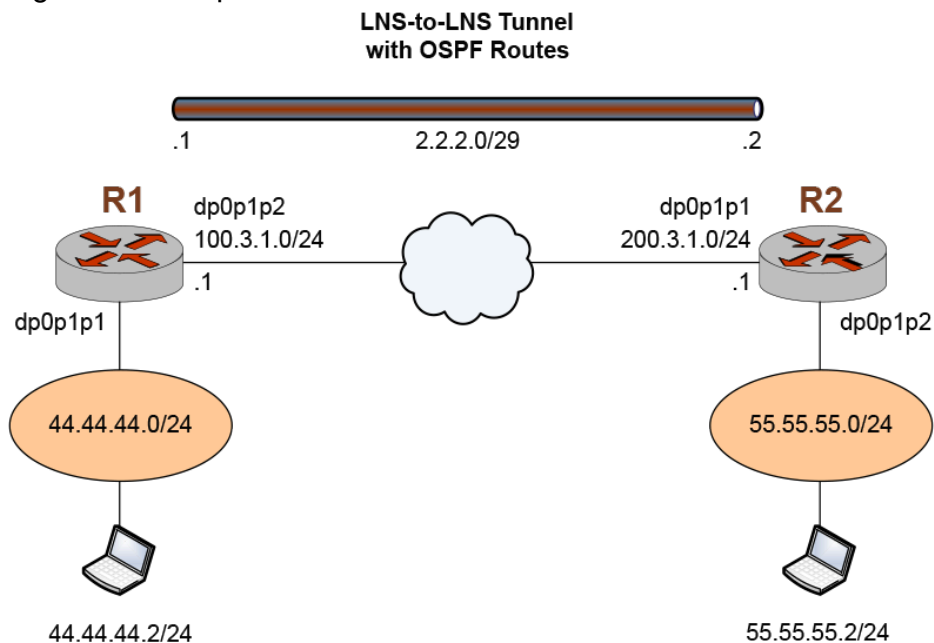
Step	Command
Set the fixed BGP router ID for the router to 2.2.2.2. This setting overrides the automatic ID selection process.	<pre>vyatta@vyatta# set protocols bgp 100 parameters router-id '2.2.2.2'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

Configuring an LNS-to-LNS tunnel with OSPF routes

This section shows the topology of a sample L2TPv3 LNS-to-LNS IPv4 tunnel with Open Shortest Path First (OSPF) configuration. To configure this sample LNS-to-LNS tunnel, perform the following steps in configuration mode.

- Configure the L2TPv3 tunnel settings and the LNS settings on the R1 router.
 - Configure the L2TPv3 tunnel settings ([Table 33: Configuring the L2TPv3 tunnel interface settings on R1](#)).
 - Configure the OSPF settings ([Table 34: Configuring the OSPF settings of the LNS-to-LNS tunnel on R1](#)).
- Configure the L2TPv3 tunnel settings and LNS settings on the R2 router.
 - Configure the L2TPv3 tunnel settings ([Table 35: Configuring the L2TPv3 tunnel interface settings on R2](#)).
 - Configure the OSPF settings ([Table 36: Configuring the OSPF settings of the LNS-to-LNS tunnel on R2](#)).

Figure 15. Sample LNS-to-LNS IPv4 tunnel with OSPF routes



[Table 33: Configuring the L2TPv3 tunnel interface settings on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 33. Configuring the L2TPv3 tunnel interface settings on R1

Step	Command
Specify IP as the encapsulation method to use for the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-ip '100.3.1.1'</pre>
Specify the local session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-session-id '1000000'</pre>
Specify the remote IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-ip '200.3.1.1'</pre>
Specify the remote session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-session-id '1000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 address '2.2.2.1/29'</pre>

[Table 34: Configuring the OSPF settings of the LNS-to-LNS tunnel on R1](#) shows how to configure the OSPF settings for the tunnel on R1. For more information about configuring OSPF settings, refer to *OSPF Reference Guide*.

Table 34. Configuring the OSPF settings of the LNS-to-LNS tunnel on R1

Step	Command
Specify the network address of the OSPF area.	<pre>vyatta@vyatta# set protocols ospf area 0 network '2.2.2.0/29'</pre>
Specify the network address of the OSPF area.	<pre>vyatta@vyatta# set protocols ospf area 0 network '44.44.44.0/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the OSPF route information.	<pre>vyatta@vyatta# run show ip ospf route OSPF process 0: Codes: C - connected, D - Discard, O - OSPF, IA - OSPF inter area N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2 C 2.2.2.0/29 [10] is directly connected, ltp2222, Area 0.0.0.0 C 44.44.44.0/24 [10] is directly connected, dp0p224p1, Area 0.0.0.0 O 55.55.55.0/24 [20] via 2.2.2.2, ltp2222, Area 0.0.0.0</pre>
Show the OSPF database information.	<pre>vyatta@vyatta# run show ip ospf database OSPF Router with ID (100.3.1.1) (Process ID 0 VRF default) Router Link States (Area 0.0.0.0) Link ID ADV Router Age Seq# CkSum Link count 100.3.1.1 100.3.1.1 309 0x80000006 0x0d98 2</pre>

Step	Command
	<pre> 200.3.1.1 200.3.1.1 288 0x80000004 0xf9c2 2 Net Link States (Area 0.0.0.0) Link ID ADV Router Age Seq# CkSum 2.2.2.1 100.3.1.1 325 0x80000001 0x4059 </pre>

[Table 35: Configuring the L2TPv3 tunnel interface settings on R2](#) shows how to configure the L2TPv3 tunnel interface on R2.

Table 35. Configuring the L2TPv3 tunnel interface settings on R2

Step	Command
Specify IP as the encapsulation method to use for the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-ip '200.3.1.1'</pre>
Specify the local session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-session-id '20000000'</pre>
Specify the remote IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-ip '100.3.1.1'</pre>
Specify the remote session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-session-id '20000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 address '2.2.2.2/29'</pre>

[Table 36: Configuring the OSPF settings of the LNS-to-LNS tunnel on R2](#) shows how to configure the OSPF settings for the tunnel on R2. For more information about configuring OSPF settings, refer to *OSPF Reference Guide*.

Table 36. Configuring the OSPF settings of the LNS-to-LNS tunnel on R2

Step	Command
Specify the network address of the OSPF area.	<pre>vyatta@vyatta# set protocols ospf area 0 network '2.2.2.0/29'</pre>
Specify the network address of the OSPF area.	<pre>vyatta@vyatta# set protocols ospf area 0 network '55.55.55.0/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Show the RIP settings.	<pre> vyatta@vyatta# run show ip ospf Codes: R - RIP, Rc - RIP connected, Rs - RIP static, K - Kernel, C - Connected, S - Static, O - OSPF, I - IS-IS, B - BGP Network Next Hop Metric From If Time Rc 2.2.2.0/29 l2tp2222 Rc 44.44.44.0/24 dp0p224p1 R 55.55.55.0/24 2.2.2.2 2 2.2.2.2 l2tp2222 02:40 </pre>
	<pre>vyatta@vyatta# run show ip rip status</pre>

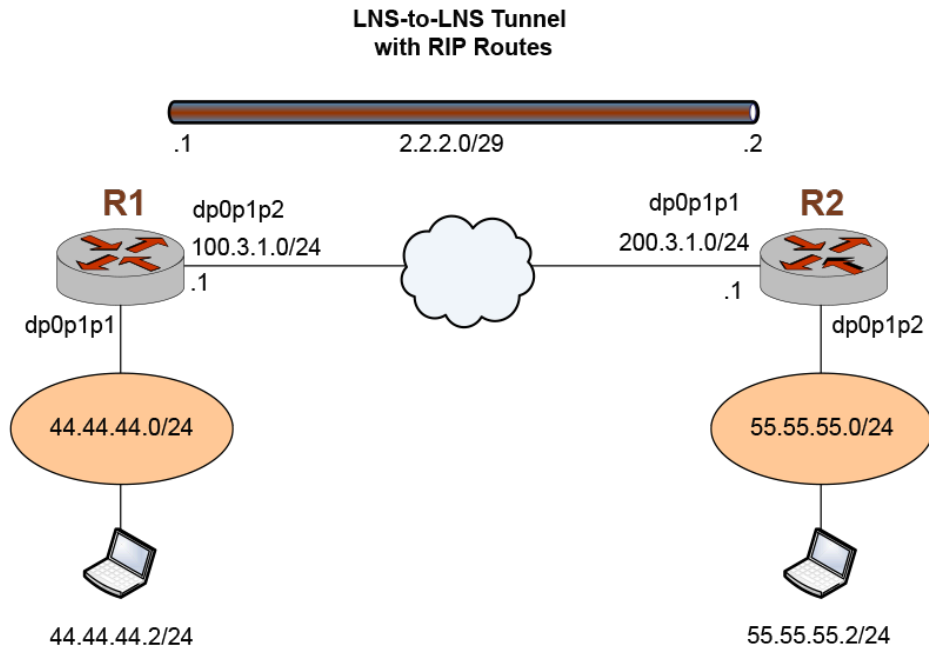
Step	Command
	<pre> RIP Database for VRF (default) Routing Protocol is "rip" Sending updates every 30 seconds with +/-50%, next due in 17 seconds Timeout after 180 seconds, garbage collect after 120 seconds Outgoing update filter list for all interface is not set Incoming update filter list for all interface is not set Default redistribution metric is 1 Redistributing: Default version control: send version 2, receive version 2 Interface Send Recv Key-chain dp0p224p1 2 2 lttp2222 2 2 Split horizon: dp0p224p1 Enabled with Poisoned Reversed lttp2222 Enabled with Poisoned Reversed Routing for Networks: 2.2.2.0/29 44.44.44.0/24 Routing Information Sources: Gateway Distance Last Update Bad Packets Bad Routes 2.2.2.2 120 00:00:12 0 0 Number of routes (including connected): 3 Distance: (default is 120) </pre>

Configuring an LNS-to-LNS tunnel with RIP routes

This section shows the topology of a sample L2TPv3 LNS-to-LNS IPv4 tunnel with Routing Information Protocol (RIP) routes. To configure this sample LNS-to-LNS tunnel, perform the following steps in configuration mode.

1. Configure the L2TPv3 tunnel settings and the LNS settings on the R1 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 37: Configuring the L2TPv3 tunnel settings on R1](#)).
 - b. Configure the RIP settings ([Table 38: Configuring the BGP settings of the LNS-to-LNS tunnel on R1](#)).
2. Configure the L2TPv3 tunnel settings and LNS settings on the R2 router.
 - a. Configure the L2TPv3 tunnel settings ([Table 39: Configuring the L2TPv3 tunnel interface settings on R2](#)).
 - b. Configure the RIP settings ([Table 40: Configuring the RIP settings of the LNS-to-LNS tunnel on R2](#)).

Figure 16. Sample LNS-to-LNS IPv4 tunnel with RIP routes



[Table 37: Configuring the L2TPv3 tunnel settings on R1](#) shows how to configure the L2TPv3 tunnel interface on R1.

Table 37. Configuring the L2TPv3 tunnel settings on R1

Step	Command
Specify IP as the encapsulation method to use for the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-ip '100.3.1.1'</pre>
Specify the local session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session local-session-id '4000000000'</pre>
Specify the remote IP address of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-ip '200.3.1.1'</pre>
Specify the remote session identifier of the ltp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 l2tp-session remote-session-id '4000000000'</pre>
Specify the IP address of the tunnel interface to which the R1 router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp1111 address '2.2.2.1/29'</pre>
Specify IP as the encapsulation method to use for the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session local-ip '100.3.1.1'</pre>
Specify the local session identifier of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session local-session-id '4000000000'</pre>
Specify the remote IP address of the ltp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth ltp2222 l2tp-session remote-ip '200.3.1.1'</pre>

Step	Command
Specify the remote session identifier of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session remote-session-id '4000000000'</pre>
Specify the IP address of R1 (the local router).	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 address '2.2.2.2/29'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 38: Configuring the BGP settings of the LNS-to-LNS tunnel on R1](#) shows how to configure the RIP settings for the tunnel on R1. For more information about configuring these settings, refer to *RIP Installing and Upgrading*.

Table 38. Configuring the BGP settings of the LNS-to-LNS tunnel on R1

Step	Command
Specify a network for RIP.	<pre>vyatta@vyatta# set protocols rip network '2.2.2.0/29'</pre>
Specify a network for RIP.	<pre>vyatta@vyatta# set protocols rip network '44.44.44.0/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>
Exit the configuration mode.	<pre>vyatta@vyatta# exit</pre>
Show the configuration information for the l2tp2222 interface.	<pre>vyatta@vyatta:~\$ show interfaces l2tpeth l2tp2222 address 2.2.2.1/29 l2tp-session { encapsulation ip local-ip 100.3.1.1 local-session-id 2000000 remote-ip 200.3.1.1 remote-session-id 2000000 }</pre>
Show the configured RIP protocols.	<pre>vyatta@vyatta:~\$ show protocols rip rip { network 2.2.2.0/29 network 44.44.44.0/24 }</pre>

[Table 39: Configuring the L2TPv3 tunnel interface settings on R2](#) shows how to configure the L2TPv3 tunnel interface on R2.

Table 39. Configuring the L2TPv3 tunnel interface settings on R2

Step	Command
Specify IP as the encapsulation method to use for the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-ip '200.3.1.1'</pre>
Specify the local session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session local-session-id '4000000000'</pre>

Step	Command
Specify the remote IP address of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-ip '100.3.1.1'</pre>
Specify the remote session identifier of the l2tp1111 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 l2tp-session remote-session-id '4000000000'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp1111 address '1.1.1.2/24'</pre>
Specify IP as the encapsulation method to use for the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session encapsulation 'ip'</pre>
Specify the local IP address of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session local-ip '200.3.1.1'</pre>
Specify the local session identifier of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session local-session-id '4000000000'</pre>
Specify the remote IP address of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session remote-ip '100.3.1.1'</pre>
Specify the IP address of the tunnel interface to which the router is connected.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 address '2.2.2.2/29'</pre>
Specify the remote session identifier of the l2tp2222 L2TPv3 tunnel interface.	<pre>vyatta@vyatta# set interfaces l2tpeth l2tp2222 l2tp-session remote-session-id '4000000000'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>vyatta@vyatta# save</pre>

[Table 40: Configuring the RIP settings of the LNS-to-LNS tunnel on R2](#) shows how to configure the RIP settings for the tunnel on R2. For more information about configuring these settings, refer to *RIP Installing and Upgrading*.

Table 40. Configuring the RIP settings of the LNS-to-LNS tunnel on R2

Step	Command
Specify a network for RIP.	<pre>vyatta@vyatta# set protocols rip network '2.2.2.0/29'</pre>
Specify a network for RIP.	<pre>vyatta@vyatta# set protocols rip network '55.55.55.0/24'</pre>
Commit the configuration.	<pre>vyatta@vyatta# commit</pre>
Save the configuration.	<pre>save</pre>

Chapter 18. L2TPv3 Data Plane Interfaces Commands

interfaces l2tpeth ip tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN ip tcp-mss limit clamp-value
```

```
delete interfaces l2tpeth ltpN ip tcp-mss limit clamp-value
```

```
show interfaces l2tpeth ltpN ip tcp-mss limit
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```
interfaces {
  l2tpeth {
    ip {
      tcp-mss {
        limit
      }
    }
  }
}
```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth ip tcp-mss mtu

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN ip tcp-mss mtu
```

```
delete interfaces l2tpeth ltpN ip tcp-mss mtu
```

```
show interfaces l2tpeth lttpN ip tcp-mss mtu
```

lttpN

L2TPv3 static tunnel interface identifier (*lttp0* through *lttpN*).

Configuration mode

```
interfaces {
  l2tpeth {
    ip {
      tcp-mss {
        mtu
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth ip tcp-mss mtu-minus

Clamps the TCP Maximum Segment Size (MSS) to the interface MTU less a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth lttpN ip tcp-mss mtu-minus subtrahend
```

```
delete interfaces l2tpeth lttpN ip tcp-mss mtu-minus subtrahend
```

```
show interfaces l2tpeth lttpN ip tcp-mss mtu-minus
```

lttpN

L2TPv3 static tunnel interface identifier (*lttp0* through *lttpN*).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```
interfaces {
  l2tpeth {
    ip {
      tcp-mss {
```

```

    mtu-minus
  }
}
}

```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth ipv6 tcp-mss limit

Clamps the TCP Maximum Segment Size (MSS) to a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth l2tpN ipv6 tcp-mss limit clamp-value
```

```
delete interfaces l2tpeth l2tpN ipv6 tcp-mss limit clamp-value
```

```
show interfaces l2tpeth l2tpN ipv6 tcp-mss limit
```

l2tpN

L2TPv3 static tunnel interface identifier (*l2tp0* through *l2tpN*).

clamp-value

Maximum segment size, in bytes (1-65535).

Configuration mode

```

interfaces {
  l2tpeth {
    ipv6 {
      tcp-mss {
        limit
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth ipv6 tcp-mss mtu

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN ipv6 tcp-mss mtu
```

```
delete interfaces l2tpeth ltpN ipv6 tcp-mss mtu
```

```
show interfaces l2tpeth ltpN ipv6 tcp-mss mtu
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

Configuration mode

```
interfaces {
  l2tpeth {
    ipv6 {
      tcp-mss {
        mtu
      }
    }
  }
}
```

Use the `set` form of this command to clamp the MSS to the MTU for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth ipv6 tcp-mss mtu-minus

Clamps the TCP Maximum Segement Size (MSS) to the interface MTU less a given value for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN ipv6 tcp-mss mtu-minus subtrahend
```

```
delete interfaces l2tpeth ltpN ipv6 tcp-mss mtu-minus subtrahend
```

```
show interfaces l2tpeth ltpN ipv6 tcp-mss mtu-minus
```

ltpN

L2TPv3 static tunnel interface identifier (*ltp0* through *ltpN*).

subtrahend

Value to subtract from the MTU in bytes (1-65535).

Configuration mode

```

interfaces {
  l2tpeth {
    ipv6 {
      tcp-mss {
        mtu-minus
      }
    }
  }
}

```

Use the `set` form of this command to set the clamping for an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the clamping for an L2TPv3 tunnel interface.

Use the `show` form of this command to show the clamping for an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session encapsulation

Specifies either IP or UDP as the encapsulation method to use for an L2TPv3 tunnel interface.

```
set interfaces l2tpeth l2tpN l2tp-session encapsulation encapsulation-method
```

```
delete interfaces l2tpeth l2tpN l2tp-session encapsulation encapsulation-method
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

encapsulation-method

Encapsulation method (IP or UDP) to use for the interface.

Configuration mode

```

interfaces {
  l2tpeth l2tpN {
    l2tp-session {
      encapsulation encapsulation-method
    }
  }
}

```

Use the `set` form of this command to specify either IP or UDP as the encapsulation method to use for an L2TPv3 interface.

Use the `delete` form of this command to remove the encapsulation method from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session local-cookie

Specifies a local cookie string.

```
set interfaces l2tpeth l2tpN l2tp-session local-cookie cookie-string
```

```
delete interfaces l2tpeth l2tpN l2tp-session local-cookie cookie-string
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

cookie-string

Cookie string. An 8- or 16-byte hexadecimal string (for example, abcdef01).

Configuration mode

```
interfaces {
  l2tpeth l2tpN {
    l2tp-session {
      local-cookie cookie-string
    }
  }
}
```

Use the `set` form of this command to specify a local cookie string.

Use the `delete` form of this command to remove a local cookie string.

interfaces l2tpeth l2tp-session local-ip

Specifies a local IP address of an L2TPv3 tunnel interface.

```
set interfaces l2tpeth l2tpN l2tp-session local-ip local-ip-address
```

```
delete interfaces l2tpeth l2tpN l2tp-session local-ip local-ip-address
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

local-ip-address

IPv4 or IPv6 local IP address of the tunnel interface.

Configuration mode

```

interfaces {
    l2tpeth lttpN {
        l2tp-session {
            local-ip local-ip-address
        }
    }
}

```

Use the `set` form of this command to specify a local IP address of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a local IP address from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session local-session-id

Specifies a local session identifier of an L2TPv3 tunnel interface.

```
set interfaces l2tpeth lttpN l2tp-session local-session-id id
```

```
delete interfaces l2tpeth lttpN l2tp-session local-session-id id
```

lttpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **lttp0** through **lttpN**, where *N* is a non-negative integer.

id

Local session identifier of the interface. The identifier ranges from 0 through 4294967295.

Configuration mode

```

interfaces {
    l2tpeth lttpN {
        l2tp-session {
            local-session-id id
        }
    }
}

```

Use the `set` form of this command to specify a local session identifier of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a local session identifier from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session local-udp-port

Specifies a local UDP port of an L2TPv3 tunnel interface.

```
set interfaces l2tpeth l2tpN l2tp-session local-udp-port udp-port
```

```
delete interfaces l2tpeth l2tpN l2tp-session local-udp-port udp-port
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

udp-port

Local UDP port for the tunnel interface. The port ranges from 1 through 65535.

Configuration mode

```
interfaces {
  l2tpeth l2tpN {
    l2tp-session {
      local-udp-port udp-port
    }
  }
}
```

Use the `set` form of this command to specify a local UDP port of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a local UDP port from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session remote-cookie

Specifies a remote cookie string.

```
set interfaces l2tpeth l2tpN l2tp-session remote-cookie cookie-string
```

```
delete interfaces l2tpeth l2tpN l2tp-session remote-cookie cookie-string
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

cookie-string

Cookie string. An 8- or 16-byte hexadecimal string (for example, abcdef01).

Configuration mode

```
interfaces {
```



```

l2tpeth ltpN {
    l2tp-session {
        remote-cookie cookie-string
    }
}

```

Use the `set` form of this command to specify a remote cookie string.

Use the `delete` form of this command to remove a remote cookie string.

interfaces l2tpeth l2tp-session remote-ip

Specifies a remote IP address of an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN l2tp-session remote-ip remote-ip-address
```

```
delete interfaces l2tpeth ltpN l2tp-session remote-ip remote-ip-address
```

ltpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **ltp0** through **ltpN**, where *N* is a non-negative integer.

remote-ip-address

Encapsulation method (IP or UDP) to use for the interface.

Configuration mode

```

interfaces {
    l2tpeth ltpN {
        l2tp-session {
            remote-ip remote-ip-address
        }
    }
}

```

Use the `set` form of this command to specify a remote IP address of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a remote IP address from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session remote-session-id

Specifies a remote session identifier of an L2TPv3 tunnel interface.

```
set interfaces l2tpeth ltpN l2tp-session remote-session-id id
```

```
delete interfaces l2tpeth ltpN l2tp-session remote-session-id id
```

lttpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **lttp0** through **lttpN**, where *N* is a non-negative integer.

id

Remote session identifier of the interface. The identifier ranges from 0 through 4294967295.

Configuration mode

```
interfaces {
  l2tpeth lttpN {
    l2tp-session {
      remote-session-id id
    }
  }
}
```

Use the `set` form of this command to specify a remote session identifier of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a remote session identifier from an L2TPv3 tunnel interface.

interfaces l2tpeth l2tp-session remote-udp-port

Specifies a remote UDP port of a tunnel interface.

```
set interfaces l2tpeth lttpN l2tp-session remote-udp-port udp-port
```

```
delete interfaces l2tpeth lttpN l2tp-session remote-udp-port udp-port
```

lttpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **lttp0** through **lttpN**, where *N* is a non-negative integer.

udp-port

Remote UDP port of the interface. The port ranges from 1 through 65535.

Configuration mode

```
interfaces {
  l2tpeth lttpN {
    l2tp-session {
      remote-udp-port udp-port
    }
  }
}
```

Use the `set` form of this command to specify a remote UDP port of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a remote UDP port from an L2TPv3 tunnel interface.

interfaces dataplane xconnect l2tpeth

Specifies the cross-connect parameters of an L2TPv3 tunnel interface.

```
set interfaces dataplane interface-name xconnect l2tpeth tunnel-interface-name
```

```
delete interfaces dataplane interface-name xconnect l2tpeth tunnel-interface-name
```

interface-name

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

tunnel-interface-name

L2TPv3 tunnel interface name.

Configuration mode

```
interfaces {
  dataplane interface-name {
    xconnect {
      l2tpeth tunnel-interface-name
    }
  }
}
```

Use the `set` form of this command to specify the cross-connect parameters of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove the cross-connect parameters of an L2TPv3 tunnel interface.

interfaces dataplane vif xconnect l2tpeth

Specifies a virtual interface (VIF) identifier and the cross-connect parameters of an L2TPv3 tunnel interface.

```
set interfaces dataplane interface-name vif id xconnect l2tpeth tunnel-interface-name
```

```
delete interfaces dataplane interface-name vif id xconnect l2tpeth tunnel-interface-name
```

interface-name

Data plane interface to which the L2TPv3 tunnel is connected.

The name of a data plane interface. For more information about the supported interface name formats, refer to [Data Plane Interface Names](#).

id

Virtual interface identifier. The identifier ranges from 1 through 4094.

tunnel-interface-name

L2TPv3 tunnel interface name.

Configuration mode

```

interfaces {
  dataplane interface-name {
    vif id {
      xconnect {
        l2tpeth tunnel-interface-name
      }
    }
  }
}

```

Use the `set` form of this command to specify a VIF identifier and the cross-connect parameters of an L2TPv3 tunnel interface.

Use the `delete` form of this command to remove a VIF identifier and the cross-connect parameters of an L2TPv3 tunnel interface.

show interfaces

Displays configuration information for an L2TPv3 tunnel interface.

```
show interfaces l2tpN
```

l2tpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **l2tp0** through **l2tpN**, where *N* is a non-negative integer.

Operational mode

The following example shows how to display configuration information for the `l2tp3333` L2TPv3 tunnel interface.

```
vyatta@vyatta# show interfaces l2tp3333
```

```

l2tp3333: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1488 qdisc pfifo_fast state
UNKNOWN group default qlen 1000
  link/ether 7a:fe:0c:b7:88:16 brd ff:ff:ff:ff:ff:ff
  inet 3.3.3.2/29 brd 3.3.3.7 scope global l2tp3333
    valid_lft forever preferred_lft forever
  inet6 fe80::78fe:cff:feb7:8816/64 scope link
    valid_lft forever preferred_lft forever
  RX:  bytes    packets   errors   dropped   overrun    mcast
      1436         14         0         0         0          0
  TX:  bytes    packets   errors   dropped   carrier collisions
      1018         11         0         0         0          0

```

show l2tpeth

Displays a list of configured L2TPv3 tunnel interfaces.

```
show l2tpeth
```

Operational mode

The following example shows how to display a list of configured L2TPv3 tunnel interfaces.

```

vyatta@vyatta# show l2tpeth
L2tpeth interface: l2tp3333
  Port: 0, ifIndex: 13
  Mac: 7a:fe:c:b7:88:16
  Addresses:
    inet 3.3.3.2/29, broadcast 3.3.3.7
    inet6 fe80::78fe:cff:feb7:8816/64, scope Link
  L2TPv3: Encap IP
    Local Address: 173.1.1.1      Peer Address: 172.1.1.1
    Local Session: 3333          Peer Session: 3333
    Local Cookie: 00             Peer Cookie: 00
  Statistics:
    Input  bytes      :           4150
    Output bytes      :           160
    Input  packets    :             41
    Output packets    :             2
    Input  multicast  :             41

```

show l2tpeth

Displays configuration information for a configured L2TPv3 tunnel interface.

```
show l2tpeth l2tpN
```

lttpN

L2TPv3 static L2TPv3 tunnel interface. The interface ranges from **lttp0** through **lttpN**, where *N* is a non-negative integer.

Operational mode

The following example shows how to display configuration information for the **lttp3333** L2TPv3 tunnel interface.

```
vyatta@vyatta:~$ show l2tpeth lttp3333
L2tpeth interface: lttp3333
  Port: 0, ifIndex: 13
  Mac: 7a:fe:c:b7:88:16
  Addresses:
    inet 3.3.3.2/29, broadcast 3.3.3.7
    inet6 fe80::78fe:cff:feb7:8816/64, scope Link
  L2TPv3: Encap IP
    Local Address: 173.1.1.1      Peer Address: 172.1.1.1
    Local Session: 3333         Peer Session: 3333
    Local Cookie: 00           Peer Cookie: 00
  Statistics:
    Input bytes      :                4150
    ...
```

Chapter 19. Data Plane Interface Names

The supported formats of interface names are:

- $dpXpYpZ$ —The name of a data plane interface, where
 - dpX specifies the data plane identifier (ID). Currently, only $dp0$ is supported.
 - pY specifies a physical or virtual PCI slot index (for example, $p129$).
 - pZ specifies a port index (for example, $p1$). For example, $dp0p1p2$, $dp0p160p1$, and $dp0p192p1$.
- $dpXemY$ —The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where emY specifies an embedded network interface number (typically, a small number). For example, $dp0em3$.
- $dpXsY$ —The name of a data plane interface in a system in which the BIOS identifies the network interface card to reside in a particular physical or virtual slot Y , where Y is typically a small number. For example, for the $dp0s2$ interface, the BIOS identifies slot 2 in the system to contain this interface.
- $dpXPnYpZ$ —The name of a data plane interface on a device that is installed on a secondary PCI bus, where Pn specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network interface cards installed on different buses with these cards having the same slot ID. The value of n must be an integer greater than 0. For example, $dp0P1p162p1$ and $dp0P2p162p1$.

Chapter 20. TCP Maximum Segment Size

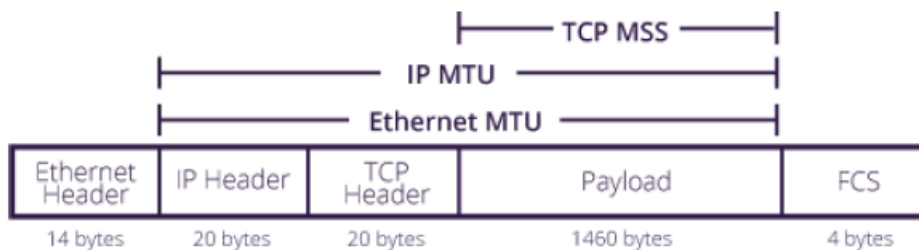
Overview

The TCP protocol uses the concept of Maximum Segment Size (MSS). MSS is simply the amount of useful data in a packet, or the maximum transmission unit (MTU) minus the IP and TCP headers.

A maximum transmission unit (MTU) is the largest packet that can be sent on a network. The standard Ethernet MTU is 1500 bytes. However, higher level protocols may want to create packets larger than an interface's MTU. To get around this issue, IPv4 allows fragmentation which divides a datagram into pieces.

Fragmentation is not supported by all devices and application, and therefore, it is best avoided. The best way to avoid fragmentation is to adjust the TCP maximum segment size (MSS).

Figure 17. TCP Maximum Segment Size



Encapsulation overhead

Fragmentation issues have become prevalent because tunnelling techniques are widely deployed and tunnel encapsulation adds overhead to the size of a packet. Here are some examples:

- GRE adds 24 bytes (20 byte IPv4, 4 byte GRE)
- 6in4 encapsulation adds 20 bytes
- 4in6 encapsulation adds 20 bytes
- MPLS adds 4 bytes for each label in the stack
- 802.1Q adds 4 bytes (Q-in-Q adds 8 bytes)
- VXLAN adds 50 bytes
- PPPoE adds 8 bytes

Advantages compared to Path MTU Discovery

TCP MSS has advantages over Path MTU Discovery (PMTUD), which relies on transmitting multiple ICMP "Fragmentation Needed" (Type 3, Code 4) messages. Notably, many network devices block all ICMP messages.

Characteristics of TCP MSS

TCP MSS has these characteristics:

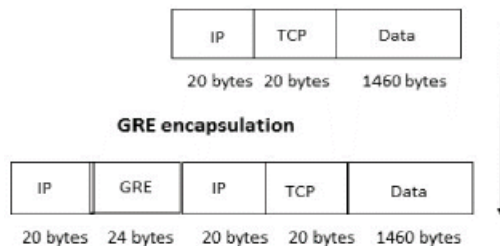
- The MSS is a parameter in the TCP options field, which comes between the TCP header and the payload (data) of the packet.
- The MSS specifies the largest amount of data that a device can receive in a single TCP segment.
- The MSS does not include the IP header and TCP header.
- The MSS is an announcement, not a negotiation.
- The MSS option is sent during the three-way handshake by both sides.
- The MSS can be different in each direction.

TCP MSS Clamping

TCP MSS clamping is a process where network devices rewrite the MSS value in TCP handshakes.

The example below shows a 24-byte GRE header added to a packet. In this case, TCP MSS should be clamped to 1436.

Figure 18. TCP MSS clamping



Configuring TCP MSS Clamping

When configured on *DANOS-Vyatta edition*, clamping is performed on TCP MSS SYN and SYN-ACK packets entering and leaving the interface.

Note: The TCP MSS option is not mandatory in TCP packets. *DANOS-Vyatta edition* can adjust the TCP MSS option value only if the MSS option is present in the packet. However, most TCP devices *do include* the MSS option.

Clamping can be configured:

- Per interface
- Independently for IPv4 and IPv6

DANOS-Vyatta edition supports the following options:

Table 41. Configuring TCP MSS

Option	Example	Command	Use
MTU maximum: Based on the interface MTU (MTU less default IP and TCP header sizes).	If the interface MTU was 1500, MSS is clamped to 1460	<code>tcp-mss mtu</code>	If there is no additional encapsulation on a packet's path, use this option.
MTU minus: Based on the interface MTU (MTU less default IP and TCP header sizes), and with an additional value subtracted.	If the MTU was 1500, MSS is clamped to 1436	<code>ip tcp-mss mtu-minus 24</code>	If PPPoE is used, use this option. PPPoE requires an additional 8 bytes and truncates the Ethernet MTU to 1492 (1500-8). PPPoE is often used with an Asymmetric Digital Subscriber Line (ADSL).
Specific value: you configure a specific value	MSS clamped to 1436	<code>ip tcp-mss limit 1436</code>	

Supported interfaces

You can configure TCP MSS clamping on these types of interfaces:

- Dataplane: See [Data Plane Interfaces Commands](#) and [VLAN Interfaces Commands](#).
- Bonding: See [Ethernet Link Bonding Interface Commands](#) and [VLAN Interfaces Commands](#).
- L2TPv3: See [L2TPv3 Data Plane Interfaces Commands](#) and [VLAN Interfaces Commands](#).
- Bridge: See *Bridging Configuration Guide*
- Tunnel: See *Tunnels Configuration Guide*
- Virtual tunnel: See *IPsec Site-to-Site VPN Configuration Guide*

Chapter 21. List of Acronyms

Acronym	Description
ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AH	Authentication Header
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DLCI	data-link connection identifier
DMI	desktop management interface
DMVPN	dynamic multipoint VPN
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol

Acronym	Description
GRE	Generic Routing Encapsulation
HDLCL	High-Level Data Link Control
I/O	Input/Output
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM
IPsec	IP Security
IPv4	IP Version 4
IPv6	IP Version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISM	Internet Standard Multicast
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
mGRE	multipoint GRE
MIB	Management Information Base
MLD	Multicast Listener Discovery
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
ND	Neighbor Discovery
NHRP	Next Hop Resolution Protocol
NIC	network interface card

Acronym	Description
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RSA	Rivest, Shamir, and Adleman
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SPT	Shortest Path Tree
SSH	Secure Shell
SSID	Service Set Identifier

Acronym	Description
SSM	Source-Specific Multicast
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
vif	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access