Ansible  
VyOS supports configuration via ansible. Need to install ansible and python3-paramiko module Structure of files   
.

├── ansible.cfg  
├── files│ └── id\_rsa\_docker.pub  
└── main.yml├── hosts   
File contents  
ansible.cfg   
[defaults]   
host\_key\_checking = no   
retry\_files\_enabled = False   
ANSIBLE\_INVENTORY\_UNPARSED\_FAILED = true   
id\_rsa\_docker.pub. Needs to declare only public key exactly.

AAAAB3NzaC1yc2EAAAADAQABAAABAQCoDgfhQJuJRFWJijHn7ZinZ3NWp4hWVrt7HFcvn0kgtP/ 5PeCtMt   
hosts   
[vyos\_hosts]   
r11 ansible\_ssh\_host=192.0.2.11

[vyos\_hosts:vars]   
ansible\_python\_interpreter=/usr/bin/python3   
ansible\_user=vyos   
ansible\_ssh\_pass=vyos   
ansible\_network\_os=vyos   
ansible\_connection=network\_cli   
main.yml  
---

- hosts: r11

connection: network\_cli  
 gather\_facts: 'no'

tasks:  
 - name: Configure remote r11  
 vyos\_config:  
 lines:  
 - set system host-name r11  
 - set system name-server 203.0.113.254  
 - set service ssh disable-host-validation

- set system login user vyos authentication public-keys docker@work type ssh-rsa - set system login user vyos authentication public-keys docker@work key "{{ lookup('file', 'id\_rsa\_docker.pub') }}"  
 - set system time-zone America/Los\_Angeles  
 - set interfaces ethernet eth0 description WAN   
Run ansible  
$ ansible-playbook -i hosts main.yml

PLAY [r11]   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Configure remote r11]   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
changed: [r11]

PLAY RECAP   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
r11 : ok=1 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0   
ARP  
ARP is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address. This mapping is a critical function in the Internet protocol suite. ARP was defined in 1982 by RFC 826 which is Internet Standard STD 37.

In Internet Protocol Version 6 (IPv6) networks, the functionality of ARP is provided by the Neighbor Discovery Protocol (NDP).

To manipulate or display ARP table entries, the following commands are implemented.

Configure  
 set protocols static arp interface <interface> address <host> mac <mac>   
This will configure a static ARP entry always resolving <address> to <mac> for interface <interface>.

Example:   
set protocols static arp interface eth0 address 192.0.2.1 mac 01:23:45:67:89:01 Operation  
 show protocols static arp   
Display all known ARP table entries spanning across all interfaces   
vyos@vyos:~$ show protocols static arp   
Address HWtype HWaddress Flags Mask Iface

10.1.1.1 ether 00:53:00:de:23:2e C eth1   
10.1.1.100 ether 00:53:00:de:23:aa CM eth1 show protocols static arp interface eth1   
Display all known ARP table entries on a given interface only (eth1): vyos@vyos:~$ show protocols static arp interface eth1   
Address HWtype HWaddress Flags Mask Iface 10.1.1.1 ether 00:53:00:de:23:2e C eth1   
10.1.1.100 ether 00:53:00:de:23:aa CM eth1 PreviousNext

Babel  
Babel is a modern routing protocol designed to be robust and efficient both in ordinary wired networks and in wireless mesh networks. By default, it uses hop-count on wired networks and a variant of ETX on wireless links, It can be configured to take radio diversity into account and to automatically compute a link’s latency and include it in the metric. It is defined in RFC 8966. Babel a dual stack protocol. A single Babel instance is able to perform routing for both IPv4 and IPv6.

General Configuration  
VyOS does not have a special command to start the Babel process. The Babel process starts when the first Babel enabled interface is configured.

set protocols babel interface <interface>   
This command specifies a Babel enabled interface by interface name. Both the sending and receiving of Babel packets will be enabled on the interface specified in this command.

Optional Configuration  
 set protocols babel parameters diversity   
This command enables routing using radio frequency diversity. This is highly recommended in networks with many wireless nodes.

Note   
If you enable this, you will probably want to set diversity-factor and channel below.

set protocols babel parameters diversity-factor <1-256>   
This command sets the multiplicative factor used for diversity routing, in units of 1/256; lower values cause diversity to play a more important role in route selection. The default it 256, which means that diversity plays no role in route selection; you will probably want to set that to 128 or less on nodes with multiple independent radios.

set protocols babel parameters resend-delay <milliseconds>   
This command specifies the time in milliseconds after which an ‘important’ request or update will be resent. The default is 2000 ms.

set protocols babel parameters smoothing-half-life <seconds>   
This command specifies the time constant, in seconds, of the smoothing algorithm used for implementing hysteresis. Larger values reduce route oscillation at the cost of very slightly increasing convergence time. The value 0 disables hysteresis, and is suitable for wired networks. The default is 4 s.

Interfaces Configuration  
 set protocols babel interface <interface> type <auto|wired|wireless>

This command sets the interface type:   
auto – automatically determines the interface type. wired – enables optimisations for wired interfaces. wireless – disables a number of optimisations that are only correct on wired interfaces. Specifying wireless is always correct, but may cause slower convergence and extra routing traffic.

set protocols babel interface <interface> split-horizon <default|disable|enable>   
This command specifies whether to perform split-horizon on the interface. Specifying no babel split-horizon is always correct, while babel split-horizon is an optimisation that should only be used on symmetric and transitive (wired) networks.

default – enable split-horizon on wired interfaces, and disable split-horizon on wireless interfaces. enable – enable split-horizon on this interfaces. disable – disable split-horizon on this interfaces.

set protocols babel interface <interface> hello-interval <milliseconds>   
This command specifies the time in milliseconds between two scheduled hellos. On wired links, Babel notices a link failure within two hello intervals; on wireless links, the link quality value is reestimated at every hello interval. The default is 4000 ms.

set protocols babel interface <interface> update-interval <milliseconds>   
This command specifies the time in milliseconds between two scheduled updates. Since Babel makes extensive use of triggered updates, this can be set to fairly high values on links with little packet loss. The default is 20000 ms.

set protocols babel interface <interface> rxcost <1-65534>   
This command specifies the base receive cost for this interface. For wireless interfaces, it specifies the multiplier used for computing the ETX reception cost (default 256); for wired interfaces, it specifies the cost that will be advertised to neighbours.

set protocols babel interface <interface> rtt-decay <1-256>   
This command specifies the decay factor for the exponential moving average of RTT samples, in units of 1/256. Higher values discard old samples faster. The default is 42.

set protocols babel interface <interface> rtt-min <milliseconds>   
This command specifies the minimum RTT, in milliseconds, starting from which we increase the cost to a neighbour. The additional cost is linear in (rtt - rtt-min). The default is 10 ms.

set protocols babel interface <interface> rtt-max <milliseconds>   
This command specifies the maximum RTT, in milliseconds, above which we don’t increase the cost to a neighbour. The default is 120 ms.

set protocols babel interface <interface> max-rtt-penalty <milliseconds>   
This command specifies the maximum cost added to a neighbour because of RTT, i.e. when the RTT is higher or equal than rtt-max. The default is 150. Setting it to 0 effectively disables the use of a RTT-based cost.

set protocols babel interface <interface> enable-timestamps   
This command enables sending timestamps with each Hello and IHU message in order to compute RTT values. It is recommended to enable timestamps on tunnel interfaces.

set protocols babel interface <interface> channel <1-254|interfering|noninterfering> This command set the channel number that diversity routing uses for this interface (see diversity option above).

1-254 – interfaces with a channel number interfere with interfering interfaces and interfaces with the same channel number. interfering – interfering interfaces are assumed to interfere with all other channels except noninterfering channels. noninterfering – noninterfering interfaces are assumed to only interfere with themselves.

Redistribution Configuration  
 set protocols babel redistribute <ipv4|ipv6> <route source>   
This command redistributes routing information from the given route source to the Babel process.

IPv4 route source: bgp, connected, eigrp, isis, kernel, nhrp, ospf, rip, static.

IPv6 route source: bgp, connected, eigrp, isis, kernel, nhrp, ospfv3, ripng, static.

set protocols babel distribute-list <ipv4|ipv6> access-list <in|out> <number>   
This command can be used to filter the Babel routes using access lists. in and out this is the direction in which the access lists are applied.

set protocols babel distribute-list <ipv4|ipv6> interface <interface> access-list <in|out> <number>   
This command allows you apply access lists to a chosen interface to filter the Babel routes.

set protocols babel distribute-list <ipv4|ipv6> prefix-list <in|out> <name>   
This command can be used to filter the Babel routes using prefix lists. in and out this is the direction in which the prefix lists are applied.

set protocols babel distribute-list <ipv4|ipv6> interface <interface> prefix-list <in|out> <name> This command allows you apply prefix lists to a chosen interface to filter the Babel routes.

Configuration Example  
Simple Babel configuration using 2 nodes and redistributing connected interfaces.

Node 1:   
set interfaces loopback lo address 10.1.1.1/32   
set interfaces loopback lo address fd12:3456:dead:beef::1/128   
set protocols babel interface eth0 type wired   
set protocols babel redistribute ipv4 connected   
set protocols babel redistribute ipv6 connected   
Node 2:   
set interfaces loopback lo address 10.2.2.2/32   
set interfaces loopback lo address fd12:3456:beef:dead::2/128   
set protocols babel interface eth0 type wired   
set protocols babel redistribute ipv4 connected   
set protocols babel redistribute ipv6 connected   
BFD  
BFD is described and extended by the following RFCs: RFC 5880, RFC 5881 and RFC 5883. In the age of very fast networks, a second of unreachability may equal millions of lost packets.

The idea behind BFD is to detect very quickly when a peer is down and take action extremely fast.

BFD sends lots of small UDP packets very quickly to ensures that the peer is still alive.

This allows avoiding the timers defined in BGP and OSPF protocol to expires.

Configure BFD  
 set protocols bfd peer <address>

Set BFD peer IPv4 address or IPv6 address  
 set protocols bfd peer <address> echo-mode   
Enables the echo transmission mode  
 set protocols bfd peer <address> multihop   
Allow this BFD peer to not be directly connected  
 set protocols bfd peer <address> source [address <address> | interface <interface>] Bind listener to specific interface/address, mandatory for IPv6  
 set protocols bfd peer <address> interval echo-interval <10-60000>   
The minimal echo receive transmission interval that this system is capable of handling set protocols bfd peer <address> interval multiplier <2-255>   
Remote transmission interval will be multiplied by this value  
 set protocols bfd peer <address> interval [receive | transmit] <10-60000>   
Interval in milliseconds  
 set protocols bfd peer <address> shutdown   
Disable a BFD peer   
Enable BFD in BGP  
 set protocols bgp neighbor <neighbor> bfd   
Enable BFD on a single BGP neighbor  
 set protocols bgp peer-group <neighbor> bfd   
Enable BFD on a BGP peer group   
Enable BFD in OSPF  
 set protocols ospf interface <interface> bfd   
Enable BFD for OSPF on an interface  
 set protocols ospfv3 interface <interface> bfd   
Enable BFD for OSPFv3 on an interface   
Enable BFD in ISIS  
 set protocols isis <name> interface <interface> bfd   
Enable BFD for ISIS on an interface   
Operational Commands  
 show bfd peers   
Show all BFD peers   
BFD Peers:  
 peer 198.51.100.33 vrf default interface eth4.100  
 ID: 4182341893  
 Remote ID: 12678929647  
 Status: up  
 Uptime: 1 month(s), 16 hour(s), 29 minute(s), 38 second(s)  
 Diagnostics: ok  
 Remote diagnostics: ok  
 Local timers:  
 Receive interval: 300ms  
 Transmission interval: 300ms  
 Echo transmission interval: 50ms  
 Remote timers:

Receive interval: 300ms  
 Transmission interval: 300ms  
 Echo transmission interval: 0ms

peer 198.51.100.55 vrf default interface eth4.101  
 ID: 4618932327  
 Remote ID: 3312345688  
 Status: up  
 Uptime: 20 hour(s), 16 minute(s), 19 second(s)  
 Diagnostics: ok  
 Remote diagnostics: ok  
 Local timers:  
 Receive interval: 300ms  
 Transmission interval: 300ms  
 Echo transmission interval: 50ms  
 Remote timers:  
 Receive interval: 300ms  
 Transmission interval: 300ms  
 Echo transmission interval: 0ms   
BFD Static Route Monitoring  
A monitored static route conditions the installation to the RIB on the BFD session running state: when BFD session is up the route is installed to RIB, but when the BFD session is down it is removed from the RIB.

Configuration  
 set protocols static route <subnet> next-hop <address> bfd profile <profile>   
Configure a static route for <subnet> using gateway <address> and use the gateway address as BFD peer destination address.

set protocols static route <subnet> next-hop <address> bfd multi-hop source <address> profile <profile>   
Configure a static route for <subnet> using gateway <address> , use source address to indentify the peer when is multi-hop session and the gateway address as BFD peer destination address.

set protocols static route6 <subnet> next-hop <address> bfd profile <profile>   
Configure a static route for <subnet> using gateway <address> and use the gateway address as BFD peer destination address.

set protocols static route6 <subnet> next-hop <address> bfd multi-hop source <address> profile <profile>   
Configure a static route for <subnet> using gateway <address> , use source address to indentify the peer when is multi-hop session and the gateway address as BFD peer destination address.

Operational Commands  
 show bfd static routes   
Showing BFD monitored static routes   
Showing BFD monitored static routes:

Next hops:

VRF default IPv4 Unicast:  
 10.10.13.3/32 peer 192.168.2.3 (status: installed) 172.16.10.3/32 peer 192.168.10.1 (status: uninstalled)

VRF default IPv4 Multicast:

VRF default IPv6 Unicast:

BGP  
BGP is one of the Exterior Gateway Protocols and the de facto standard interdomain routing protocol. The latest BGP version is 4. BGP-4 is described in RFC 1771 and updated by RFC 4271. RFC 2858 adds multiprotocol support to BGP.

VyOS makes use of FRR and we would like to thank them for their effort!

Basic Concepts  
Autonomous Systems  
From RFC 1930:   
An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy.

Each AS has an identifying number associated with it called an ASN. This is a two octet value ranging in value from 1 to 65535. The AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not be advertised on the global Internet. The 2-byte AS number range has been exhausted. 4-byte AS numbers are specified in RFC 6793, and provide a pool of 4294967296 AS numbers.

The ASN is one of the essential elements of BGP. BGP is a distance vector routing protocol, and the AS-Path framework provides distance vector metric and loop detection to BGP.

set protocols bgp system-as <asn>   
Set local ASN that this router represents. This is a a mandatory option!

Address Families  
Multiprotocol extensions enable BGP to carry routing information for multiple network layer protocols. BGP supports an Address Family Identifier (AFI) for IPv4 and IPv6.

Route Selection  
The route selection process used by FRR’s BGP implementation uses the following decision criterion, starting at the top of the list and going towards the bottom until one of the factors can be used.

1. Weight check   
Prefer higher local weight routes to lower routes.

2. Local preference check   
Prefer higher local preference routes to lower.

3. Local route check   
Prefer local routes (statics, aggregates, redistributed) to received routes.

4. AS path length check   
Prefer shortest hop-count AS\_PATHs.

5. Origin check

Prefer the lowest origin type route. That is, prefer IGP origin routes to EGP, to Incomplete routes.

6. MED check   
Where routes with a MED were received from the same AS, prefer the route with the lowest MED.

7. External check   
Prefer the route received from an external, eBGP peer over routes received from other types of peers.

8. IGP cost check   
Prefer the route with the lower IGP cost.

9. Multi-path check   
If multi-pathing is enabled, then check whether the routes not yet distinguished in preference may be considered equal. If bgp bestpath as-path multipath-relax is set, all such routes are considered equal, otherwise routes received via iBGP with identical AS\_PATHs or routes received from eBGP neighbours in the same AS are considered equal.

10. Already-selected external check   
Where both routes were received from eBGP peers, then prefer the route which is already selected. Note that this check is not applied if bgp bestpath compare-routerid is configured. This check can prevent some cases of oscillation.

11. Router-ID check   
Prefer the route with the lowest router-ID. If the route has an ORIGINATOR\_ID attribute, through iBGP reflection, then that router ID is used, otherwise the router-ID of the peer the route was received from is used.

12. Cluster-List length check   
The route with the shortest cluster-list length is used. The cluster-list reflects the iBGP reflection path the route has taken.

13. Peer address   
Prefer the route received from the peer with the higher transport layer address, as a last-resort tie-breaker.

Capability Negotiation  
When adding IPv6 routing information exchange feature to BGP. There were some proposals.

IETF IDR adopted a proposal called Multiprotocol Extension for BGP. The specification is described in RFC 2283. The protocol does not define new protocols. It defines new attributes to existing BGP. When it is used exchanging IPv6 routing information it is called BGP-4+. When it is used for exchanging multicast routing information it is called MBGP.

bgpd supports Multiprotocol Extension for BGP. So if a remote peer supports the protocol, bgpd can exchange IPv6 and/or multicast routing information.

Traditional BGP did not have the feature to detect a remote peer’s capabilities, e.g. whether it can handle prefix types other than IPv4 unicast routes. This was a big problem using   
Multiprotocol Extension for BGP in an operational network. RFC 2842 adopted a feature called Capability Negotiation. bgpd use this Capability Negotiation to detect the remote peer’s capabilities. If a peer is only configured as an IPv4 unicast neighbor, bgpd does not send these Capability Negotiation packets (at least not unless other optional BGP features require capability negotiation).

By default, FRR will bring up peering with minimal common capability for the both sides. For example, if the local router has unicast and multicast capabilities and the remote router only has unicast capability the local router will establish the connection with unicast only capability.

When there are no common capabilities, FRR sends Unsupported Capability error and then resets the connection.

Configuration  
BGP Router Configuration  
First of all you must configure BGP router with the ASN. The AS number is an identifier for the autonomous system. The BGP protocol uses the AS number for detecting whether the BGP connection is internal or external. VyOS does not have a special command to start the BGP process. The BGP process starts when the first neighbor is configured.

set protocols bgp system-as <asn>   
Set local autonomous system number that this router represents. This is a mandatory option!

Peers Configuration  
Defining Peers  
 set protocols bgp neighbor <address|interface> remote-as <nasn>   
This command creates a new neighbor whose remote-as is <nasn>. The neighbor address can be an IPv4 address or an IPv6 address or an interface to use for the connection. The command is applicable for peer and peer group.

set protocols bgp neighbor <address|interface> remote-as internal   
Create a peer as you would when you specify an ASN, except that if the peers ASN is different than mine as specified under the protocols bgp <asn> command the connection will be denied.

set protocols bgp neighbor <address|interface> remote-as external   
Create a peer as you would when you specify an ASN, except that if the peers ASN is the same as mine as specified under the protocols bgp <asn> command the connection will be denied.

set protocols bgp neighbor <address|interface> local-role <role> [strict]   
BGP roles are defined in RFC RFC 9234 and provide an easy way to add route leak prevention, detection and mitigation. The local Role value is negotiated with the new BGP Role capability which has a built-in check of the corresponding value. In case of a mismatch the new OPEN Roles Mismatch Notification <2, 11> would be sent. The correct Role pairs are:   
Provider - Customer   
Peer - Peer   
RS-Server - RS-Client   
If strict is set the BGP session won’t become established until the BGP neighbor sets local Role on its side. This configuration parameter is defined in RFC RFC 9234 and is used to enforce the corresponding configuration at your counter-parts side.

Routes that are sent from provider, rs-server, or the peer local-role (or if received by customer, rs-client, or the peer local-role) will be marked with a new Only to Customer (OTC) attribute.

Routes with this attribute can only be sent to your neighbor if your local-role is provider or rs-server. Routes with this attribute can be received only if your local-role is customer or rs-client.

In case of peer-peer relationship routes can be received only if OTC value is equal to your neighbor AS number.

All these rules with OTC will help to detect and mitigate route leaks and happen automatically if local-role is set.

set protocols bgp neighbor <address|interface> shutdown   
This command disable the peer or peer group. To reenable the peer use the delete form of this command.

set protocols bgp neighbor <address|interface> description <text> Set description of the peer or peer group.

set protocols bgp neighbor <address|interface> update-source <address|interface>   
Specify the IPv4 source address to use for the BGP session to this neighbor, may be specified as either an IPv4 address directly or as an interface name.

Capability Negotiation  
 set protocols bgp neighbor <address|interface> capability dynamic   
This command would allow the dynamic update of capabilities over an established BGP session.

set protocols bgp neighbor <address|interface> capability extended-nexthop   
Allow bgp to negotiate the extended-nexthop capability with it’s peer. If you are peering over a IPv6 Link-Local address then this capability is turned on automatically. If you are peering over a IPv6 Global Address then turning on this command will allow BGP to install IPv4 routes with IPv6 nexthops if you do not have IPv4 configured on interfaces.

set protocols bgp neighbor <address|interface> disable-capability-negotiation   
Suppress sending Capability Negotiation as OPEN message optional parameter to the peer. This command only affects the peer is configured other than IPv4 unicast configuration.

When remote peer does not have capability negotiation feature, remote peer will not send any capabilities at all. In that case, bgp configures the peer with configured capabilities.

You may prefer locally configured capabilities more than the negotiated capabilities even though remote peer sends capabilities. If the peer is configured by override-capability, VyOS ignores received capabilities then override negotiated capabilities with configured values. Additionally you should keep in mind that this feature fundamentally disables the ability to use widely deployed BGP features. BGP unnumbered, hostname support, AS4, Addpath, Route Refresh, ORF, Dynamic Capabilities, and graceful restart.

set protocols bgp neighbor <address|interface> override-capability   
This command allow override the result of Capability Negotiation with local configuration. Ignore remote peer’s capability value.

set protocols bgp neighbor <address|interface> strict-capability-match   
This command forces strictly compare remote capabilities and local capabilities. If capabilities are different, send Unsupported Capability error then reset connection.

You may want to disable sending Capability Negotiation OPEN message optional parameter to the peer when remote peer does not implement Capability Negotiation. Please use disable-capability-negotiation command to disable the feature.

Peer Parameters  
 set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> allowas-in number <number>   
This command accept incoming routes with AS path containing AS number with the same value as the current system AS. This is used when you want to use the same AS number in your sites, but you can’t connect them directly.

The number parameter (1-10) configures the amount of accepted occurences of the system AS number in AS path.

This command is only allowed for eBGP peers. It is not applicable for peer groups.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> as-override   
This command override AS number of the originating router with the local AS number.

Usually this configuration is used in PEs (Provider Edge) to replace the incoming customer AS number so the connected CE ( Customer Edge) can use the same AS number as the other customer sites. This allows customers of the provider network to use the same AS number across their sites.

This command is only allowed for eBGP peers.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> attribute-unchanged <as-path|med|next-hop>   
This command specifies attributes to be left unchanged for advertisements sent to a peer or peer group.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> maximum-prefix <number>   
This command specifies a maximum number of prefixes we can receive from a given peer. If this number is exceeded, the BGP session will be destroyed. The number range is 1 to 4294967295.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> nexthop-self   
This command forces the BGP speaker to report itself as the next hop for an advertised route it advertised to a neighbor.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> remove-private-as   
This command removes the private ASN of routes that are advertised to the configured peer. It removes only private ASNs on routes advertised to EBGP peers.

If the AS-Path for the route has only private ASNs, the private ASNs are removed.

If the AS-Path for the route has a private ASN between public ASNs, it is assumed that this is a design choice, and the private ASN is not removed.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> soft-reconfiguration inbound   
Changes in BGP policies require the BGP session to be cleared. Clearing has a large negative impact on network operations. Soft reconfiguration enables you to generate inbound updates from a neighbor, change and activate BGP policies without clearing the BGP session.

This command specifies that route updates received from this neighbor will be stored unmodified, regardless of the inbound policy. When inbound soft reconfiguration is enabled, the stored updates are processed by the new policy configuration to create new inbound updates.

Note   
Storage of route updates uses memory. If you enable soft reconfiguration inbound for multiple neighbors, the amount of memory used can become significant.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> weight <number>   
This command specifies a default weight value for the neighbor’s routes. The number range is 1 to 65535.

set protocols bgp neighbor <address|interface> advertisement-interval <seconds>   
This command specifies the minimum route advertisement interval for the peer. The interval value is 0 to 600 seconds, with the default advertisement interval being 0.

set protocols bgp neighbor <address|interface> disable-connected-check   
This command allows peerings between directly connected eBGP peers using loopback addresses without adjusting the default TTL of 1.

set protocols bgp neighbor <address|interface> disable-send-community <extended|standard> This command specifies that the community attribute should not be sent in route updates to a peer. By default community attribute is sent.

set protocols bgp neighbor <address|interface> ebgp-multihop <number>   
This command allows sessions to be established with eBGP neighbors when they are multiple hops away. When the neighbor is not directly connected and this knob is not enabled, the session will not establish. The number of hops range is 1 to 255. This command is mutually exclusive with ttl-security hops.

set protocols bgp neighbor <address|interface> local-as <asn> [no-prepend] [replace-as] Specify an alternate AS for this BGP process when interacting with the specified peer or peer group. With no modifiers, the specified local-as is prepended to the received AS\_PATH when receiving routing updates from the peer, and prepended to the outgoing AS\_PATH (after the process local AS) when transmitting local routes to the peer.

If the no-prepend attribute is specified, then the supplied local-as is not prepended to the received AS\_PATH.

If the replace-as attribute is specified, then only the supplied local-as is prepended to the AS\_PATH when transmitting local-route updates to this peer.

Note   
This command is only allowed for eBGP peers.

set protocols bgp neighbor <address|interface> passive   
Configures the BGP speaker so that it only accepts inbound connections from, but does not initiate outbound connections to the peer or peer group.

set protocols bgp neighbor <address|interface> password <text>   
This command specifies a MD5 password to be used with the tcp socket that is being used to connect to the remote peer.

set protocols bgp neighbor <address|interface> ttl-security hops <number>   
This command enforces Generalized TTL Security Mechanism (GTSM), as specified in RFC 5082. With this command, only neighbors that are specified number of hops away will be allowed to become neighbors. The number of hops range is 1 to 254. This command is mutually exclusive with ebgp-multihop.

Peer Groups  
Peer groups are used to help improve scaling by generating the same update information to all members of a peer group. Note that this means that the routes generated by a member of a peer group will be sent back to that originating peer with the originator identifier attribute set to indicated the originating peer. All peers not associated with a specific peer group are treated as belonging to a default peer group, and will share updates.

set protocols bgp peer-group <name>

This command defines a new peer group. You can specify to the group the same parameters that you can specify for specific neighbors.

Note   
If you apply a parameter to an individual neighbor IP address, you override the action defined for a peer group that includes that IP address.

set protocols bgp neighbor <address|interface> peer-group <name> This command bind specific peer to peer group with a given name.

Network Advertisement Configuration  
 set protocols bgp address-family <ipv4-unicast|ipv6-unicast> network <prefix> This command is used for advertising IPv4 or IPv6 networks.

Note   
By default, the BGP prefix is advertised even if it’s not present in the routing table. This behaviour differs from the implementation of some vendors.

set protocols bgp parameters network-import-check   
This configuration modifies the behavior of the network statement. If you have this configured the underlying network must exist in the routing table.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> default-originate [route-map <name>]   
By default, VyOS does not advertise a default route (0.0.0.0/0) even if it is in routing table.

When you want to announce default routes to the peer, use this command. Using optional argument route-map you can inject the default route to given neighbor only if the conditions in the route map are met.

Route Aggregation Configuration  
 set protocols bgp address-family <ipv4-unicast|ipv6-unicast> aggregate-address <prefix> This command specifies an aggregate address. The router will also announce longer-prefixes inside of the aggregate address.

set protocols bgp address-family <ipv4-unicast|ipv6-unicast> aggregate-address <prefix> as-set This command specifies an aggregate address with a mathematical set of autonomous systems. This command summarizes the AS\_PATH attributes of all the individual routes.

set protocols bgp address-family <ipv4-unicast|ipv6-unicast> aggregate-address <prefix> summary-only   
This command specifies an aggregate address and provides that longer-prefixes inside of the aggregate address are suppressed before sending BGP updates out to peers.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> unsuppress-map <name>   
This command applies route-map to selectively unsuppress prefixes suppressed by   
summarisation.

Redistribution Configuration  
 set protocols bgp address-family <ipv4-unicast|ipv6-unicast> redistribute <route source> This command redistributes routing information from the given route source to the BGP process. There are six modes available for route source: connected, kernel, ospf, rip, static, table.

set protocols bgp address-family <ipv4-unicast|ipv6-unicast> redistribute <route source> metric <number>

This command specifies metric (MED) for redistributed routes. The metric range is 0 to 4294967295. There are six modes available for route source: connected, kernel, ospf, rip, static, table.

set protocols bgp address-family <ipv4-unicast|ipv6-unicast> redistribute <route source> route-map <name>   
This command allows to use route map to filter redistributed routes. There are six modes available for route source: connected, kernel, ospf, rip, static, table.

General Configuration  
Common parameters  
 set protocols bgp parameters router-id <id>   
This command specifies the router-ID. If router ID is not specified it will use the highest interface IP address.

set protocols bgp address-family <ipv4-unicast|ipv6-unicast> maximum-paths <ebgp|ibgp> <number>   
This command defines the maximum number of parallel routes that the BGP can support. In order for BGP to use the second path, the following attributes have to match: Weight, Local Preference, AS Path (both AS number and AS path length), Origin code, MED, IGP metric. Also, the next hop address for each path must be different.

set protocols bgp parameters log-neighbor-changes   
This command enable logging neighbor up/down changes and reset reason.

set protocols bgp parameters no-client-to-client-reflection   
This command disables route reflection between route reflector clients. By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, route reflection is not required. In this case, use the no-client-to-client-reflection command to disable client-to-client reflection.

set protocols bgp parameters no-fast-external-failover   
Disable immediate session reset if peer’s connected link goes down.

set protocols bgp listen range <prefix> peer-group <name>   
This command is useful if one desires to loosen the requirement for BGP to have strictly defined neighbors. Specifically what is allowed is for the local router to listen to a range of IPv4 or IPv6 addresses defined by a prefix and to accept BGP open messages. When a TCP connection (and subsequently a BGP open message) from within this range tries to connect the local router then the local router will respond and connect with the parameters that are defined within the peer group. One must define a peer-group for each range that is listed. If no peer-group is defined then an error will keep you from committing the configuration.

set protocols bgp listen limit <number>   
This command goes hand in hand with the listen range command to limit the amount of BGP neighbors that are allowed to connect to the local router. The limit range is 1 to 5000.

set protocols bgp parameters ebgp-requires-policy   
This command changes the eBGP behavior of FRR. By default FRR enables RFC 8212   
functionality which affects how eBGP routes are advertised, namely no routes are advertised across eBGP sessions without some sort of egress route-map/policy in place. In VyOS however we have this RFC functionality disabled by default so that we can preserve backwards

compatibility with older versions of VyOS. With this option one can enable RFC 8212 functionality to operate.

Administrative Distance  
 set protocols bgp parameters distance global <external|internal|local> <distance>   
This command change distance value of BGP. The arguments are the distance values for external routes, internal routes and local routes respectively. The distance range is 1 to 255.

set protocols bgp parameters distance prefix <subnet> distance <distance>   
This command sets the administrative distance for a particular route. The distance range is 1 to 255.

Note   
Routes with a distance of 255 are effectively disabled and not installed into the kernel.

Timers  
 set protocols bgp timers holdtime <seconds>   
This command specifies hold-time in seconds. The timer range is 4 to 65535. The default value is 180 second. If you set value to 0 VyOS will not hold routes.

set protocols bgp timers keepalive <seconds>   
This command specifies keep-alive time in seconds. The timer can range from 4 to 65535. The default value is 60 second.

Route Dampening  
When a route fails, a routing update is sent to withdraw the route from the network’s routing tables. When the route is re-enabled, the change in availability is also advertised. A route that continually fails and returns requires a great deal of network traffic to update the network about the route’s status.

Route dampening wich described in RFC 2439 enables you to identify routes that repeatedly fail and return. If route dampening is enabled, an unstable route accumulates penalties each time the route fails and returns. If the accumulated penalties exceed a threshold, the route is no longer advertised. This is route suppression. Routes that have been suppressed are re-entered into the routing table only when the amount of their penalty falls below a threshold.

A penalty of 1000 is assessed each time the route fails. When the penalties reach a predefined threshold (suppress-value), the router stops advertising the route.

Once a route is assessed a penalty, the penalty is decreased by half each time a predefined amount of time elapses (half-life-time). When the accumulated penalties fall below a predefined threshold (reuse-value), the route is unsuppressed and added back into the BGP routing table.

No route is suppressed indefinitely. Maximum-suppress-time defines the maximum time a route can be suppressed before it is re-advertised.

set protocols bgp parameters dampening half-life <minutes>   
This command defines the amount of time in minutes after which a penalty is reduced by half. The timer range is 10 to 45 minutes.

set protocols bgp parameters dampening re-use <seconds>   
This command defines the accumulated penalty amount at which the route is re-advertised. The penalty range is 1 to 20000.

set protocols bgp parameters dampening start-suppress-time <seconds>

This command defines the accumulated penalty amount at which the route is suppressed. The penalty range is 1 to 20000.

set protocols bgp parameters dampening max-suppress-time <seconds>   
This command defines the maximum time in minutes that a route is suppressed. The timer range is 1 to 255 minutes.

Route Selection Configuration  
 set protocols bgp parameters always-compare-med   
This command provides to compare the MED on routes, even when they were received from different neighbouring ASes. Setting this option makes the order of preference of routes more defined, and should eliminate MED induced oscillations.

set protocols bgp parameters bestpath as-path confed   
This command specifies that the length of confederation path sets and sequences should be taken into account during the BGP best path decision process.

set protocols bgp parameters bestpath as-path multipath-relax   
This command specifies that BGP decision process should consider paths of equal AS\_PATH length candidates for multipath computation. Without the knob, the entire AS\_PATH must match for multipath computation.

set protocols bgp parameters bestpath as-path ignore   
Ignore AS\_PATH length when selecting a route  
 set protocols bgp parameters bestpath compare-routerid   
Ensure that when comparing routes where both are equal on most metrics, including local-pref, AS\_PATH length, IGP cost, MED, that the tie is broken based on router-ID.

If this option is enabled, then the already-selected check, where already selected eBGP routes are preferred, is skipped.

If a route has an ORIGINATOR\_ID attribute because it has been reflected, that ORIGINATOR\_ID will be used. Otherwise, the router-ID of the peer the route was received from will be used. The advantage of this is that the route-selection (at this point) will be more deterministic. The disadvantage is that a few or even one lowest-ID router may attract all traffic to otherwise-equal paths because of this check. It may increase the possibility of MED or IGP oscillation, unless other measures were taken to avoid these. The exact behaviour will be sensitive to the iBGP and reflection topology.

set protocols bgp parameters bestpath med confed   
This command specifies that BGP considers the MED when comparing routes originated from different sub-ASs within the confederation to which this BGP speaker belongs. The default state, where the MED attribute is not considered.

set protocols bgp parameters bestpath med missing-as-worst   
This command specifies that a route with a MED is always considered to be better than a route without a MED by causing the missing MED attribute to have a value of infinity. The default state, where the missing MED attribute is considered to have a value of zero.

set protocols bgp parameters default local-pref <local-pref value>   
This command specifies the default local preference value. The local preference range is 0 to 4294967295.

set protocols bgp parameters deterministic-med

This command provides to compare different MED values that advertised by neighbours in the same AS for routes selection. When this command is enabled, routes from the same   
autonomous system are grouped together, and the best entries of each group are compared.

set protocols bgp address-family ipv4-unicast network <prefix> backdoor   
This command allows the router to prefer route to specified prefix learned via IGP through backdoor link instead of a route to the same prefix learned via EBGP.

Route Filtering Configuration  
In order to control and modify routing information that is exchanged between peers you can use route-map, filter-list, prefix-list, distribute-list.

For inbound updates the order of preference is:  
• route-map  
• filter-list  
• prefix-list, distribute-list   
For outbound updates the order of preference is:  
• prefix-list, distribute-list  
• filter-list  
• route-map   
Note   
The attributes prefix-list and distribute-list are mutually exclusive, and only one command (distribute-list or prefix-list) can be applied to each inbound or outbound direction for a particular neighbor.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> distribute-list <export|import> <number>   
This command applies the access list filters named in <number> to the specified BGP neighbor to restrict the routing information that BGP learns and/or advertises. The arguments export and import specify the direction in which the access list are applied.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> prefix-list <export|import> <name>   
This command applies the prfefix list filters named in <name> to the specified BGP neighbor to restrict the routing information that BGP learns and/or advertises. The arguments export and import specify the direction in which the prefix list are applied.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> route-map <export|import> <name>   
This command applies the route map named in <name> to the specified BGP neighbor to control and modify routing information that is exchanged between peers. The arguments export and import specify the direction in which the route map are applied.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> filter-list <export|import> <name>   
This command applies the AS path access list filters named in <name> to the specified BGP neighbor to restrict the routing information that BGP learns and/or advertises. The arguments export and import specify the direction in which the AS path access list are applied.

set protocols bgp neighbor <address|interface> address-family <ipv4-unicast|ipv6-unicast> capability orf <receive|send>

This command enables the ORF capability (described in RFC 5291) on the local router, and enables ORF capability advertisement to the specified BGP peer. The receive keyword configures a router to advertise ORF receive capabilities. The send keyword configures a router to advertise ORF send capabilities. To advertise a filter from a sender, you must create an IP prefix list for the specified BGP peer applied in inbound derection.

set protocols bgp neighbor <address|interface> solo   
This command prevents from sending back prefixes learned from the neighbor.

BGP Scaling Configuration  
BGP routers connected inside the same AS through BGP belong to an internal BGP session, or IBGP. In order to prevent routing table loops, IBGP speaker does not advertise IBGP-learned routes to other IBGP speaker (Split Horizon mechanism). As such, IBGP requires a full mesh of all peers. For large networks, this quickly becomes unscalable.

There are two ways that help us to mitigate the BGPs full-mesh requirement in a network:• Using BGP route-reflectors   
 Using BGP confederation•  
Route Reflector Configuration  
Introducing route reflectors removes the need for the full-mesh. When you configure a route reflector you have to tell the router whether the other IBGP router is a client or non-client. A client is an IBGP router that the route reflector will “reflect” routes to, the non-client is just a regular IBGP neighbor. Route reflectors mechanism is described in RFC 4456 and updated by RFC 7606.

set protocols bgp neighbor <address> address-family <ipv4-unicast|ipv6-unicast> route-reflector-client   
This command specifies the given neighbor as route reflector client.

set protocols bgp parameters cluster-id <id>   
This command specifies cluster ID which identifies a collection of route reflectors and their clients, and is used by route reflectors to avoid looping. By default cluster ID is set to the BGP router id value, but can be set to an arbitrary 32-bit value.

Confederation Configuration  
A BGP confederation divides our AS into sub-ASes to reduce the number of required IBGP peerings. Within a sub-AS we still require full-mesh IBGP but between these sub-ASes we use something that looks like EBGP but behaves like IBGP (called confederation BGP). Confederation mechanism is described in RFC 5065  
 set protocols bgp parameters confederation identifier <asn>   
This command specifies a BGP confederation identifier. <asn> is the number of the autonomous system that internally includes multiple sub-autonomous systems (a confederation).

set protocols bgp parameters confederation peers <nsubasn>   
This command sets other confederations <nsubasn> as members of autonomous system specified by confederation identifier <asn>.

Operational Mode Commands  
Show  
 show <ip|ipv6> bgp   
This command displays all entries in BGP routing table.

BGP table version is 10, local router ID is 10.0.35.3, vrf id 0

Default local pref 100, local AS 65000   
Status codes: s suppressed, d damped, h history, \* valid, > best, = multipath, i internal, r RIB-failure, S Stale, R Removed   
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self   
Origin codes: i - IGP, e - EGP, ? - incomplete

Network Next Hop Metric LocPrf Weight Path \*> 198.51.100.0/24 10.0.34.4 0 0 65004 i \*> 203.0.113.0/24 10.0.35.5 0 0 65005 i

Displayed 2 routes and 2 total paths  
 show <ip|ipv6> bgp <address|prefix>   
This command displays information about the particular entry in the BGP routing table.

BGP routing table entry for 198.51.100.0/24   
Paths: (1 available, best #1, table default)  
 Advertised to non peer-group peers:  
 10.0.13.1 10.0.23.2 10.0.34.4 10.0.35.5  
 65004  
 10.0.34.4 from 10.0.34.4 (10.0.34.4)  
 Origin IGP, metric 0, valid, external, best (First path received)  
 Last update: Wed Jan 6 12:18:53 2021  
 show ip bgp cidr-only   
This command displays routes with classless interdomain routing (CIDR).

show <ip|ipv6> bgp community <value>   
This command displays routes that belong to specified BGP communities. Valid value is a community number in the range from 1 to 4294967200, or AA:NN (autonomous system-community number/2-byte number), no-export, local-as, or no-advertise.

show <ip|ipv6> bgp community-list <name>   
This command displays routes that are permitted by the BGP community list.

show ip bgp dampened-paths   
This command displays BGP dampened routes.

show ip bgp flap-statistics   
This command displays information about flapping BGP routes.

show ip bgp filter-list <name>   
This command displays BGP routes allowed by the specified AS Path access list.

show <ip|ipv6> bgp neighbors <address> advertised-routes This command displays BGP routes advertised to a neighbor.

show <ip|ipv6> bgp neighbors <address> received-routes   
This command displays BGP routes originating from the specified BGP neighbor before inbound policy is applied. To use this command inbound soft reconfiguration must be enabled.

show <ip|ipv6> bgp neighbors <address> routes   
This command displays BGP received-routes that are accepted after filtering.

show <ip|ipv6> bgp neighbors <address> dampened-routes   
This command displays dampened routes received from BGP neighbor.

show <ip|ipv6> bgp regexp <text>   
This command displays information about BGP routes whose AS path matches the specified regular expression.

show <ip|ipv6> bgp summary   
This command displays the status of all BGP connections.

IPv4 Unicast Summary:   
BGP router identifier 10.0.35.3, local AS number 65000 vrf-id 0 BGP table version 11   
RIB entries 5, using 920 bytes of memory   
Peers 4, using 82 KiB of memory

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd 10.0.13.1 4 65000 148 159 0 0 0 02:16:01 0   
10.0.23.2 4 65000 136 143 0 0 0 02:13:21 0   
10.0.34.4 4 65004 161 163 0 0 0 02:16:01 1   
10.0.35.5 4 65005 162 166 0 0 0 02:16:01 1

Total number of neighbors 4   
Reset  
 reset <ip|ipv6> bgp <address> [soft [in|out]]   
This command resets BGP connections to the specified neighbor IP address. With argument soft this command initiates a soft reset. If you do not specify the in or out options, both inbound and outbound soft reconfiguration are triggered.

reset ip bgp all   
This command resets all BGP connections of given router.

reset ip bgp dampening   
This command uses to clear BGP route dampening information and to unsuppress suppressed routes.

reset ip bgp external   
This command resets all external BGP peers of given router.

reset ip bgp peer-group <name> [soft [in|out]]   
This command resets BGP connections to the specified peer group. With argument soft this command initiates a soft reset. If you do not specify the in or out options, both inbound and outbound soft reconfiguration are triggered.

Examples  
IPv4 peering  
A simple eBGP configuration:   
Node 1:   
set protocols bgp system-as 65534   
set protocols bgp neighbor 192.168.0.2 ebgp-multihop '2'   
set protocols bgp neighbor 192.168.0.2 remote-as '65535'   
set protocols bgp neighbor 192.168.0.2 update-source '192.168.0.1' set protocols bgp address-family ipv4-unicast network '172.16.0.0/16' set protocols bgp parameters router-id '192.168.0.1'

Node 2:   
set protocols bgp system-as 65535   
set protocols bgp neighbor 192.168.0.1 ebgp-multihop '2'   
set protocols bgp neighbor 192.168.0.1 remote-as '65534'   
set protocols bgp neighbor 192.168.0.1 update-source '192.168.0.2'   
set protocols bgp address-family ipv4-unicast network '172.17.0.0/16'   
set protocols bgp parameters router-id '192.168.0.2'   
Don’t forget, the CIDR declared in the network statement MUST exist in your routing table (dynamic or static), the best way to make sure that is true is creating a static route: Node 1:   
set protocols static route 172.16.0.0/16 blackhole distance '254'   
Node 2:   
set protocols static route 172.17.0.0/16 blackhole distance '254'   
IPv6 peering  
A simple BGP configuration via IPv6.

Node 1:   
set protocols bgp system-as 65534   
set protocols bgp neighbor 2001:db8::2 ebgp-multihop '2'   
set protocols bgp neighbor 2001:db8::2 remote-as '65535'   
set protocols bgp neighbor 2001:db8::2 update-source '2001:db8::1'   
set protocols bgp neighbor 2001:db8::2 address-family ipv6-unicast   
set protocols bgp address-family ipv6-unicast network '2001:db8:1::/48'   
set protocols bgp parameters router-id '10.1.1.1'   
Node 2:   
set protocols bgp system-as 65535   
set protocols bgp neighbor 2001:db8::1 ebgp-multihop '2'   
set protocols bgp neighbor 2001:db8::1 remote-as '65534'   
set protocols bgp neighbor 2001:db8::1 update-source '2001:db8::2'   
set protocols bgp neighbor 2001:db8::1 address-family ipv6-unicast   
set protocols bgp address-family ipv6-unicast network '2001:db8:2::/48'   
set protocols bgp parameters router-id '10.1.1.2'   
Don’t forget, the CIDR declared in the network statement MUST exist in your routing table (dynamic or static), the best way to make sure that is true is creating a static route: Node 1:   
set protocols static route6 2001:db8:1::/48 blackhole distance '254'   
Node 2:   
set protocols static route6 2001:db8:2::/48 blackhole distance '254'   
Route Filtering  
Route filter can be applied using a route-map:   
Node1:   
set policy prefix-list AS65535-IN rule 10 action 'permit'   
set policy prefix-list AS65535-IN rule 10 prefix '172.16.0.0/16'   
set policy prefix-list AS65535-OUT rule 10 action 'deny'   
set policy prefix-list AS65535-OUT rule 10 prefix '172.16.0.0/16'

set policy prefix-list6 AS65535-IN rule 10 action 'permit'   
set policy prefix-list6 AS65535-IN rule 10 prefix '2001:db8:2::/48' set policy prefix-list6 AS65535-OUT rule 10 action 'deny'   
set policy prefix-list6 AS65535-OUT rule 10 prefix '2001:db8:2::/48'

set policy route-map AS65535-IN rule 10 action 'permit'   
set policy route-map AS65535-IN rule 10 match ip address prefix-list 'AS65535-IN' set policy route-map AS65535-IN rule 10 match ipv6 address prefix-list 'AS65535-IN' set policy route-map AS65535-IN rule 20 action 'deny'   
set policy route-map AS65535-OUT rule 10 action 'deny'   
set policy route-map AS65535-OUT rule 10 match ip address prefix-list 'AS65535-OUT' set policy route-map AS65535-OUT rule 10 match ipv6 address prefix-list 'AS65535-OUT' set policy route-map AS65535-OUT rule 20 action 'permit'

set protocols bgp system-as 65534   
set protocols bgp neighbor 2001:db8::2 address-family ipv4-unicast route-map export 'AS65535-OUT'   
set protocols bgp neighbor 2001:db8::2 address-family ipv4-unicast route-map import 'AS65535-IN'   
set protocols bgp neighbor 2001:db8::2 address-family ipv6-unicast route-map export 'AS65535-OUT'   
set protocols bgp neighbor 2001:db8::2 address-family ipv6-unicast route-map import 'AS65535-IN'   
Node2:   
set policy prefix-list AS65534-IN rule 10 action 'permit'   
set policy prefix-list AS65534-IN rule 10 prefix '172.17.0.0/16'   
set policy prefix-list AS65534-OUT rule 10 action 'deny'   
set policy prefix-list AS65534-OUT rule 10 prefix '172.17.0.0/16'   
set policy prefix-list6 AS65534-IN rule 10 action 'permit'   
set policy prefix-list6 AS65534-IN rule 10 prefix '2001:db8:1::/48'   
set policy prefix-list6 AS65534-OUT rule 10 action 'deny'   
set policy prefix-list6 AS65534-OUT rule 10 prefix '2001:db8:1::/48'

set policy route-map AS65534-IN rule 10 action 'permit'   
set policy route-map AS65534-IN rule 10 match ip address prefix-list 'AS65534-IN' set policy route-map AS65534-IN rule 10 match ipv6 address prefix-list 'AS65534-IN' set policy route-map AS65534-IN rule 20 action 'deny'   
set policy route-map AS65534-OUT rule 10 action 'deny'   
set policy route-map AS65534-OUT rule 10 match ip address prefix-list 'AS65534-OUT' set policy route-map AS65534-OUT rule 10 match ipv6 address prefix-list 'AS65534-OUT' set policy route-map AS65534-OUT rule 20 action 'permit'

set protocols bgp system-as 65535

set protocols bgp neighbor 2001:db8::1 address-family ipv4-unicast route-map export 'AS65534-OUT'   
set protocols bgp neighbor 2001:db8::1 address-family ipv4-unicast route-map import 'AS65534-IN'   
set protocols bgp neighbor 2001:db8::1 address-family ipv6-unicast route-map export 'AS65534-OUT'   
set protocols bgp neighbor 2001:db8::1 address-family ipv6-unicast route-map import 'AS65534-IN'   
We could expand on this and also deny link local and multicast in the rule 20 action deny.

Boot Options  
Warning   
This function may be highly disruptive. It may cause major service interruption, so make sure you really need it and verify your input carefully.

VyOS has several kernel command line options to modify the normal boot process. To add an option, select the desired image in GRUB menu at load time, press e, edit the first line, and press Ctrl-x to boot when ready.

Specify custom config file  
Tells the system to use specified file instead of /config/config.boot. If specified file does not exist or is not readable, fall back to default config. No additional verification is performed, so make sure you specify a valid config file.

vyos-config=/path/to/file   
To load the factory default config, use:   
vyos-config=/opt/vyatta/etc/config.boot.default   
Disable specific boot process steps  
These options disable some boot steps. Make sure you understand the boot process well before using them!

no-vyos-migrate  
Do not perform config migration.

no-vyos-firewall  
Do not initialize default firewall chains, renders any firewall configuration unusable.

PreviousNext

Build VyOS  
Prerequisites  
There are different ways you can build VyOS.

Building using a Docker container, although not the only way, is the easiest way as all   
dependencies are managed for you. However, you can also set up your own build machine and run a Native Build.

Note   
Starting with VyOS 1.2 the release model of VyOS has changed. VyOS is now free as in speech, but not as in beer. This means that while VyOS is still an open source project, the release ISOs are no longer free and can only be obtained via subscription, or by contributing to the community.

The source code remains public and an ISO can be built using the process outlined in this chapter.

This will guide you through the process of building a VyOS ISO using Docker. This process has been tested on clean installs of Debian Jessie, Stretch, and Buster.

Docker  
Installing Docker and prerequisites:   
$ sudo apt-get update   
$ sudo apt-get install -y apt-transport-https ca-certificates curl gnupg2 software-properties-common   
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -  
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $ (lsb\_release -cs) stable"   
$ sudo apt-get update   
$ sudo apt-get install -y docker-ce   
To be able to use Docker without sudo, the current non-root user must be added to the docker group by calling: sudo usermod -aG docker yourusername.

Hint   
Doing so grants privileges equivalent to the root user! It is recommended to remove the non-root user from the docker group after building the VyOS ISO. See also Docker as non-root.

Note   
The build process needs to be built on a local file system, building on SMB or NFS shares will result in the container failing to build properly! VirtualBox Drive Share is also not an option as block device operations are not implemented and the drive is always mounted as “nodev”Build Container  
The container can be built by hand or by fetching the pre-built one from DockerHub. Using the pre-built containers from the VyOS DockerHub organisation will ensure that the container is always up-to-date. A rebuild is triggered once the container changes (please note this will take 2-3 hours after pushing to the vyos-build repository).

Dockerhub  
To manually download the container from DockerHub, run:   
$ docker pull vyos/vyos-build:crux # For VyOS 1.2   
$ docker pull vyos/vyos-build:equuleus # For VyOS 1.3   
$ docker pull vyos/vyos-build:sagitta # For VyOS 1.4   
$ docker pull vyos/vyos-build:current # For rolling release   
Build from source  
The container can also be built directly from source:   
# For VyOS 1.2 (crux)   
$ git clone -b crux --single-branch https://github.com/vyos/vyos-build # For VyOS 1.3 (equuleus)   
$ git clone -b equuleus --single-branch https://github.com/vyos/vyos-build # For VyOS 1.4 (sagitta)   
$ git clone -b sagitta --single-branch https://github.com/vyos/vyos-build

$ cd vyos-build

$ docker build -t vyos/vyos-build:crux docker # For VyOS 1.2   
$ docker build -t vyos/vyos-build:current docker # For rolling release   
Note   
Since VyOS has switched to Debian (11) Bullseye in its current branch, you will require individual container for current, equuleus and crux builds.

Tips and Tricks  
You can create yourself some handy Bash aliases to always launch the latest - per release train (current or crux) - container. Add the following to your .bash\_aliases file:   
alias vybld='docker pull vyos/vyos-build:current && docker run --rm -it \  
 -v "$(pwd)":/vyos \  
 -v "$HOME/.gitconfig":/etc/gitconfig \  
 -v "$HOME/.bash\_aliases":/home/vyos\_bld/.bash\_aliases \  
 -v "$HOME/.bashrc":/home/vyos\_bld/.bashrc \  
 -w /vyos --privileged --sysctl net.ipv6.conf.lo.disable\_ipv6=0 \  
 -e GOSU\_UID=$(id -u) -e GOSU\_GID=$(id -g) \  
 vyos/vyos-build:current bash'

alias vybld\_crux='docker pull vyos/vyos-build:crux && docker run --rm -it \  
 -v "$(pwd)":/vyos \  
 -v "$HOME/.gitconfig":/etc/gitconfig \  
 -v "$HOME/.bash\_aliases":/home/vyos\_bld/.bash\_aliases \  
 -v "$HOME/.bashrc":/home/vyos\_bld/.bashrc \  
 -w /vyos --privileged --sysctl net.ipv6.conf.lo.disable\_ipv6=0 \  
 -e GOSU\_UID=$(id -u) -e GOSU\_GID=$(id -g) \  
 vyos/vyos-build:crux bash'   
Now you are prepared with two new aliases vybld and vybld\_crux to spawn your development containers in your current working directory.

Note   
Some VyOS packages (namely vyos-1x) come with build-time tests which verify some of the internal library calls that they work as expected. Those tests are carried out through the Python Unittest module. If you want to build the vyos-1x package (which is our main development package) you need to start your Docker container using the following argument: --sysctl net.ipv6.conf.lo.disable\_ipv6=0, otherwise those tests will fail.

Native Build  
To build VyOS natively you require a properly configured build host with the following Debian versions installed:  
• Debian Jessie for VyOS 1.2 (crux)  
• Debian Buster for VyOS 1.3 (equuleus)   
 Debian Bullseye for VyOS 1.4 (sagitta)  
•To start, clone the repository to your local machine:   
# For VyOS 1.2 (crux)   
$ git clone -b crux --single-branch https://github.com/vyos/vyos-build

# For VyOS 1.3 (equuleus)

$ git clone -b equuleus --single-branch https://github.com/vyos/vyos-build

# For VyOS 1.4 (sagitta)   
$ git clone -b sagitta --single-branch https://github.com/vyos/vyos-build   
For the packages required, you can refer to the docker/Dockerfile file in the repository. The ./build-vyos-image script will also warn you if any dependencies are missing.

Once you have the required dependencies installed, you may proceed with the steps described in Build ISO.

Build ISO  
Now as you are aware of the prerequisites we can continue and build our own ISO from source. For this we have to fetch the latest source code from GitHub. Please note as this will differ for both current and crux.

# For VyOS 1.2 (crux)   
$ git clone -b crux --single-branch https://github.com/vyos/vyos-build

# For VyOS 1.3 (equuleus)   
$ git clone -b equuleus --single-branch https://github.com/vyos/vyos-build

# For VyOS 1.4 (sagitta)   
$ git clone -b sagitta --single-branch https://github.com/vyos/vyos-build   
Now a fresh build of the VyOS ISO can begin. Change directory to the vyos-build directory and run:   
$ cd vyos-build   
# For VyOS 1.2 (crux)   
$ docker run --rm -it --privileged -v $(pwd):/vyos -w /vyos vyos/vyos-build:crux bash

# For VyOS 1.3 (equuleus)   
$ docker run --rm -it --privileged -v $(pwd):/vyos -w /vyos vyos/vyos-build:equuleus bash

# For VyOS 1.4 (sagitta)   
$ docker run --rm -it --privileged -v $(pwd):/vyos -w /vyos vyos/vyos-build:sagitta bash # For MacOS (crux, equuleus, sagitta)   
$ git clone https://github.com/vyos/vyos-utils-misc   
$ cd build-tools/macos-build

# For VyOS 1.2 (crux)   
$ os=jessie64 branch=crux make build

# For VyOS 1.3 (equuleus)   
$ os=buster64 branch=equuleus make build

# For VyOS 1.4 (sagitta)   
$ os=buster64 branch=sagitta make build   
Start the build:

# For VyOS 1.2 (crux) and VyOS 1.3 (equuleus)   
vyos\_bld@8153428c7e1f:/vyos$ ./configure --architecture amd64 --build-by "j.randomhacker@vyos.io"   
vyos\_bld@8153428c7e1f:/vyos$ sudo make iso

# For VyOS 1.4 (sagitta)   
vyos\_bld@8153428c7e1f:/vyos$ sudo make clean   
vyos\_bld@8153428c7e1f:/vyos$ sudo ./build-vyos-image iso --architecture amd64 --build-by "j.randomhacker@vyos.io"   
When the build is successful, the resulting iso can be found inside the build directory as live-image-[architecture].hybrid.iso.

Good luck!

Hint   
Building VyOS on Windows WSL2 with Docker integrated into WSL2 will work like a charm. No problems are known so far!

Customize  
This ISO can be customized with the following list of configure options. The full and current list can be generated with ./build-vyos-image --help:   
$ vyos\_bld@8153428c7e1f:/vyos$ sudo ./build-vyos-image --help  
 I: Checking if packages required for VyOS image build are installed  
 usage: build-vyos-image [-h] [--architecture ARCHITECTURE]  
 [--build-by BUILD\_BY] [--debian-mirror DEBIAN\_MIRROR]  
 [--debian-security-mirror DEBIAN\_SECURITY\_MIRROR]  
 [--pbuilder-debian-mirror PBUILDER\_DEBIAN\_MIRROR]  
 [--vyos-mirror VYOS\_MIRROR] [--build-type BUILD\_TYPE]  
 [--version VERSION] [--build-comment BUILD\_COMMENT] [--debug] [--dry-run]  
 [--custom-apt-entry CUSTOM\_APT\_ENTRY] [--custom-apt-key CUSTOM\_APT\_KEY]  
 [--custom-package CUSTOM\_PACKAGE]  
 [build\_flavor]

positional arguments:  
 build\_flavor Build flavor

optional arguments:  
 -h, --help show this help message and exit  
 --architecture ARCHITECTURE  
 Image target architecture (amd64 or arm64)  
 --build-by BUILD\_BY Builder identifier (e.g. jrandomhacker@example.net) --debian-mirror DEBIAN\_MIRROR  
 Debian repository mirror  
 --debian-security-mirror DEBIAN\_SECURITY\_MIRROR  
 Debian security updates mirror  
 --pbuilder-debian-mirror PBUILDER\_DEBIAN\_MIRROR  
 Debian repository mirror for pbuilder env bootstrap

--vyos-mirror VYOS\_MIRROR  
 VyOS package mirror  
 --build-type BUILD\_TYPE  
 Build type, release or development  
 --version VERSION Version number (release builds only)  
 --build-comment BUILD\_COMMENT  
 Optional build comment  
 --debug Enable debug output  
 --dry-run Check build configuration and exit  
 --custom-apt-entry CUSTOM\_APT\_ENTRY  
 Custom APT entry  
 --custom-apt-key CUSTOM\_APT\_KEY  
 Custom APT key file  
 --custom-package CUSTOM\_PACKAGE  
 Custom package to install from repositories   
ISO Build Issues  
There are (rare) situations where building an ISO image is not possible at all due to a broken package feed in the background. APT is not very good at reporting the root cause of the issue.

Your ISO build will likely fail with a more or less similar looking error message: The following packages have unmet dependencies:  
 vyos-1x : Depends: accel-ppp but it is not installable   
E: Unable to correct problems, you have held broken packages.

P: Begin unmounting filesystems...

P: Saving caches...

Reading package lists...

Building dependency tree...

Reading state information...

Del frr-pythontools 7.5-20210215-00-g8a5d3b7cd-0 [38.9 kB]   
Del accel-ppp 1.12.0-95-g59f8e1b [475 kB]   
Del frr 7.5-20210215-00-g8a5d3b7cd-0 [2671 kB]   
Del frr-snmp 7.5-20210215-00-g8a5d3b7cd-0 [55.1 kB]   
Del frr-rpki-rtrlib 7.5-20210215-00-g8a5d3b7cd-0 [37.3 kB]   
make: \*\*\* [Makefile:30: iso] Error 1   
(10:13) vyos\_bld ece068908a5b:/vyos [current] #   
To debug the build process and gain additional information of what could be the root cause, you need to use chroot to change into the build directry. This is explained in the following step by step procedure:   
vyos\_bld ece068908a5b:/vyos [current] # sudo chroot build/chroot /bin/bash   
We now need to mount some required, volatile filesystems   
(live)root@ece068908a5b:/# mount -t proc none /proc   
(live)root@ece068908a5b:/# mount -t sysfs none /sys   
(live)root@ece068908a5b:/# mount -t devtmpfs none /dev   
We now are free to run any command we would like to use for debugging, e.g. re-installing the failed package after updating the repository.

(live)root@ece068908a5b:/# apt-get update; apt-get install vyos-1x   
Get:1 file:/root/packages ./ InRelease   
Ign:1 file:/root/packages ./ InRelease   
Get:2 file:/root/packages ./ Release [1235 B]   
Get:2 file:/root/packages ./ Release [1235 B]   
Get:3 file:/root/packages ./ Release.gpg   
Ign:3 file:/root/packages ./ Release.gpg   
Hit:4 http://repo.powerdns.com/debian buster-rec-43 InRelease   
Hit:5 http://repo.saltstack.com/py3/debian/10/amd64/archive/3002.2 buster InRelease Hit:6 http://deb.debian.org/debian bullseye InRelease   
Hit:7 http://deb.debian.org/debian buster InRelease   
Hit:8 http://deb.debian.org/debian-security buster/updates InRelease   
Hit:9 http://deb.debian.org/debian buster-updates InRelease   
Hit:10 http://deb.debian.org/debian buster-backports InRelease   
Hit:11 http://dev.packages.vyos.net/repositories/current current InRelease   
Reading package lists... Done   
N: Download is performed unsandboxed as root as file '/root/packages/./InRelease' couldn't be accessed by user '\_apt'. - pkgAcquire::Run (13: Permission denied)   
Reading package lists... Done   
Building dependency tree   
Reading state information... Done   
Some packages could not be installed. This may mean that you have   
requested an impossible situation or if you are using the unstable   
distribution that some required packages have not yet been created   
or been moved out of Incoming.

The following information may help to resolve the situation:

The following packages have unmet dependencies:  
 vyos-1x : Depends: accel-ppp but it is not installable   
E: Unable to correct problems, you have held broken packages.

Now it’s time to fix the package mirror and rerun the last step until the package installation succeeds again!

Linux Kernel  
The Linux kernel used by VyOS is heavily tied to the ISO build process. The file data/defaults.json hosts a JSON definition of the kernel version used kernel\_version and the kernel\_flavor of the kernel which represents the kernel’s LOCAL\_VERSION. Both together form the kernel version variable in the system:   
vyos@vyos:~$ uname -r   
6.1.52-amd64-vyos  
•  
 Accel-PPP• Intel NIC drivers  
• Inter QAT   
Each of those modules holds a dependency on the kernel version and if you are lucky enough to receive an ISO build error which sounds like:

I: Create initramfs if it does not exist.

Extra argument '6.1.52-amd64-vyos'   
Usage: update-initramfs {-c|-d|-u} [-k version] [-v] [-b directory] Options:  
 -k version Specify kernel version or 'all'  
 -c Create a new initramfs  
 -u Update an existing initramfs  
 -d Remove an existing initramfs  
 -b directory Set alternate boot directory  
 -v Be verbose   
See update-initramfs(8) for further details.

E: config/hooks/live/17-gen\_initramfs.chroot failed (exit non-zero). You should check for errors.

The most obvious reasons could be:  
• vyos-build repo is outdated, please git pull to update to the latest release kernel version from us.

• You have your own custom kernel \*.deb packages in the packages folder but neglected to create all required out-of tree modules like Accel-PPP, Intel QAT or Intel NIC drivers Building The Kernel  
The kernel build is quite easy, most of the required steps can be found in the   
vyos-build/packages/linux-kernel/Jenkinsfile but we will walk you through it.

Clone the kernel source to vyos-build/packages/linux-kernel/:   
$ cd vyos-build/packages/linux-kernel/   
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git   
Check out the required kernel version - see vyos-build/data/defaults.json file (example uses kernel 4.19.146):   
$ cd vyos-build/packages/linux-kernel/linux   
$ git checkout v4.19.146   
Checking out files: 100% (61536/61536), done.

Note: checking out 'v4.19.146'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

git checkout -b <new-branch-name>

HEAD is now at 015e94d0e37b Linux 4.19.146   
Now we can use the helper script build-kernel.sh which does all the necessary voodoo by applying required patches from the vyos-build/packages/linux-kernel/patches folder, copying our kernel configuration x86\_64\_vyos\_defconfig to the right location, and finally building the Debian packages.

Note   
Building the kernel will take some time depending on the speed and quantity of your CPU/cores and disk speed. Expect 20 minutes (or even longer) on lower end hardware.

(18:59) vyos\_bld 412374ca36b8:/vyos/vyos-build/packages/linux-kernel [current] # ./build-kernel.sh   
I: Copy Kernel config (x86\_64\_vyos\_defconfig) to Kernel Source   
I: Apply Kernel patch: /vyos/vyos-build/packages/linux-kernel/patches/kernel/0001-VyOS-Add-linkstate-IP-device-attribute.patch   
patching file Documentation/networking/ip-sysctl.txt   
patching file include/linux/inetdevice.h   
patching file include/linux/ipv6.h   
patching file include/uapi/linux/ip.h   
patching file include/uapi/linux/ipv6.h   
patching file net/ipv4/devinet.c   
Hunk #1 succeeded at 2319 (offset 1 line).

patching file net/ipv6/addrconf.c   
patching file net/ipv6/route.c   
I: Apply Kernel patch: /vyos/vyos-build/packages/linux-kernel/patches/kernel/0002-VyOS-add-inotify-support-for-stackable-filesystems-o.patch   
patching file fs/notify/inotify/Kconfig   
patching file fs/notify/inotify/inotify\_user.c   
patching file fs/overlayfs/super.c   
Hunk #2 succeeded at 1713 (offset 9 lines).

Hunk #3 succeeded at 1739 (offset 9 lines).

Hunk #4 succeeded at 1762 (offset 9 lines).

patching file include/linux/inotify.h   
I: Apply Kernel patch: /vyos/vyos-build/packages/linux-kernel/patches/kernel/0003-RFC-builddeb-add-linux-tools-package-with-perf.patch   
patching file scripts/package/builddeb   
I: make x86\_64\_vyos\_defconfig  
 HOSTCC scripts/basic/fixdep  
 HOSTCC scripts/kconfig/conf.o  
 YACC scripts/kconfig/zconf.tab.c  
 LEX scripts/kconfig/zconf.lex.c  
 HOSTCC scripts/kconfig/zconf.tab.o  
 HOSTLD scripts/kconfig/conf   
#   
# configuration written to .config   
#   
I: Generate environment file containing Kernel variable   
I: Build Debian Kernel package  
 UPD include/config/kernel.release   
/bin/sh ./scripts/package/mkdebian   
dpkg-buildpackage -r"fakeroot -u" -a$(cat debian/arch) -b -nc -uc

dpkg-buildpackage: info: source package linux-4.19.146-amd64-vyos   
dpkg-buildpackage: info: source version 4.19.146-1   
dpkg-buildpackage: info: source distribution buster   
dpkg-buildpackage: info: source changed by vyos\_bld <christian@poessinger.com> dpkg-buildpackage: info: host architecture amd64   
dpkg-buildpackage: warning: debian/rules is not executable; fixing that  
 dpkg-source --before-build .

debian/rules build   
make KERNELRELEASE=4.19.146-amd64-vyos ARCH=x86 KBUILD\_BUILD\_VERSION=1 KBUILD\_SRC=  
 SYSTBL arch/x86/include/generated/asm/syscalls\_32.h

...

dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: binaries to analyze should already be installed in their package's directory   
dpkg-shlibdeps: warning: package could avoid a useless dependency if   
/vyos/vyos-build/packages/linux-kernel/linux/debian/toolstmp/usr/bin/trace /vyos/vyos-build/packages/linux-kernel/linux/debian/toolstmp/usr/bin/perf were not linked against libcrypto.so.1.1 (they use none of the library's symbols)   
dpkg-shlibdeps: warning: package could avoid a useless dependency if   
/vyos/vyos-build/packages/linux-kernel/linux/debian/toolstmp/usr/bin/trace /vyos/vyos-

build/packages/linux-kernel/linux/debian/toolstmp/usr/bin/perf were not linked against libcrypt.so.1 (they use none of the library's symbols)   
dpkg-deb: building package 'linux-tools-4.19.146-amd64-vyos' in '../linux-tools-4.19.146-amd64-vyos\_4.19.146-1\_amd64.deb'.

dpkg-genbuildinfo --build=binary  
 dpkg-genchanges --build=binary >../linux-4.19.146-amd64-vyos\_4.19.146-1\_amd64.changes dpkg-genchanges: warning: package linux-image-4.19.146-amd64-vyos-dbg in control file but not in files list   
dpkg-genchanges: info: binary-only upload (no source code included)  
 dpkg-source --after-build .

dpkg-buildpackage: info: binary-only upload (no source included)   
In the end you will be presented with the kernel binary packages which you can then use in your custom ISO build process, by placing all the \*.deb files in the vyos-build/packages folder where they will be used automatically when building VyOS as documented above.

Firmware  
If you upgrade your kernel or include new drivers you may need new firmware. Build a new vyos-linux-firmware package with the included helper scripts.

$ cd vyos-build/packages/linux-kernel   
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git $ ./build-linux-firmware.sh   
$ cp vyos-linux-firmware\_\*.deb ../   
This tries to automatically detect which blobs are needed based on which drivers were built. If it fails to find the correct files you can add them manually to   
vyos-build/packages/linux-kernel/build-linux-firmware.sh:   
ADD\_FW\_FILES="iwlwifi\* ath11k/QCA6390/\*/\*.bin"   
Building Out-Of-Tree Modules  
Building the kernel is one part, but now you also need to build the required out-of-tree modules so everything is lined up and the ABIs match. To do so, you can again take a look at   
vyos-build/packages/linux-kernel/Jenkinsfile to see all of the required modules and their selected versions. We will show you how to build all the current required modules.

Accel-PPP  
First, clone the source code and check out the appropriate version by running:   
$ cd vyos-build/packages/linux-kernel   
$ git clone https://github.com/accel-ppp/accel-ppp.git   
We again make use of a helper script and some patches to make the build work. Just run the following command:   
$ ./build-accel-ppp.sh   
I: Build Accel-PPP Debian package   
CMake Deprecation Warning at CMakeLists.txt:3 (cmake\_policy):  
 The OLD behavior for policy CMP0003 will be removed from a future version  
 of CMake.

The cmake-policies(7) manual explains that the OLD behaviors of all policies are deprecated and that a policy should be set to OLD only under

specific short-term circumstances. Projects should be ported to the NEW behavior and not rely on setting a policy to OLD.

-- The C compiler identification is GNU 8.3.0

...

CPack: Create package using DEB   
CPack: Install projects   
CPack: - Run preinstall target for: accel-ppp   
CPack: - Install project: accel-ppp   
CPack: Create package   
CPack: - package: /vyos/vyos-build/packages/linux-kernel/accel-ppp/build/accel-ppp.deb generated.

After compiling the packages you will find yourself the newly generated \*.deb binaries in vyos-build/packages/linux-kernel from which you can copy them to the vyos-build/packages folder for inclusion during the ISO build.

Intel NIC  
The Intel NIC drivers do not come from a Git repository, instead we just fetch the tarballs from our mirror and compile them.

Simply use our wrapper script to build all of the driver modules.

./build-intel-drivers.sh  
 % Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed   
100 490k 100 490k 0 0 648k 0 --:--:-- --:--:-- --:--:-- 648k   
I: Compile Kernel module for Intel ixgbe driver

...

I: Building Debian package vyos-intel-iavf   
Doing `require 'backports'` is deprecated and will not load any backport in the next major release.

Require just the needed backports instead, or 'backports/latest'.

Debian packaging tools generally labels all files in /etc as config files, as mandated by policy, so fpm defaults to this behavior for deb packages. You can disable this default behavior with --deb-no-default-config-files flag {:level=>:warn}   
Created package {:path=>"vyos-intel-iavf\_4.0.1-0\_amd64.deb"}   
I: Cleanup iavf source   
After compiling the packages you will find yourself the newly generated \*.deb binaries in vyos-build/packages/linux-kernel from which you can copy them to the vyos-build/packages folder for inclusion during the ISO build.

Intel QAT  
The Intel QAT (Quick Assist Technology) drivers do not come from a Git repository, instead we just fetch the tarballs from 01.org, Intel’s open-source website.

Simply use our wrapper script to build all of the driver modules.

$ ./build-intel-qat.sh  
 % Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed   
100 5065k 100 5065k 0 0 1157k 0 0:00:04 0:00:04 --:--:-- 1157k I: Compile Kernel module for Intel qat driver   
checking for a BSD-compatible install... /usr/bin/install -c   
checking whether build environment is sane... yes   
checking for a thread-safe mkdir -p... /bin/mkdir -p   
checking for gawk... gawk   
checking whether make sets $(MAKE)... yes

...

I: Building Debian package vyos-intel-qat   
Doing `require 'backports'` is deprecated and will not load any backport in the next major release.

Require just the needed backports instead, or 'backports/latest'.

Debian packaging tools generally labels all files in /etc as config files, as mandated by policy, so fpm defaults to this behavior for deb packages. You can disable this default behavior with --deb-no-default-config-files flag {:level=>:warn}   
Created package {:path=>"vyos-intel-qat\_1.7.l.4.9.0-00008-0\_amd64.deb"}   
I: Cleanup qat source   
After compiling the packages you will find yourself the newly generated \*.deb binaries in vyos-build/packages/linux-kernel from which you can copy them to the vyos-build/packages folder for inclusion during the ISO build.

Packages  
If you are brave enough to build yourself an ISO image containing any modified package from our GitHub organisation - this is the place to be.

Any “modified” package may refer to an altered version of e.g. vyos-1x package that you would like to test before filing a pull request on GitHub.

Building an ISO with any customized package is in no way different than building a regular (customized or not) ISO image. Simply place your modified \*.deb package inside the packages folder within vyos-build. The build process will then pickup your custom package and integrate it into your ISO.

Troubleshooting  
Debian APT is not very verbose when it comes to errors. If your ISO build breaks for whatever reason and you suspect it’s a problem with APT dependencies or installation you can add this small patch which increases the APT verbosity during ISO build.

diff --git i/scripts/live-build-config w/scripts/live-build-config index 1b3b454..3696e4e 100755  
--- i/scripts/live-build-config   
+++ w/scripts/live-build-config   
@@ -57,7 +57,8 @@ lb config noauto \

--firmware-binary false \  
 --updates true \  
 --security true \  
- --apt-options "--yes -oAcquire::Check-Valid-Until=false" \   
+ --apt-options "--yes -oAcquire::Check-Valid-Until=false -oDebug::BuildDeps=true -oDebug::pkgDepCache::AutoInstall=true \   
+ -oDebug::pkgDepCache::Marker=true -oDebug::pkgProblemResolver=true -oDebug::Acquire::gpgv=true" \  
 --apt-indices false  
 "${@}"  
 """   
Virtualization Platforms  
QEMU  
Run the following command after building the ISO image.

$ make qemu   
VMware  
Run the following command after building the QEMU image.

$ make vmware   
Packages  
VyOS itself comes with a bunch of packages that are specific to our system and thus cannot be found in any Debian mirror. Those packages can be found at the VyOS GitHub project in their source format can easily be compiled into a custom Debian (\*.deb) package.

The easiest way to compile your package is with the above mentioned Docker container, it includes all required dependencies for all VyOS related packages.

Assume we want to build the vyos-1x package on our own and modify it to our needs. We first need to clone the repository from GitHub.

$ git clone https://github.com/vyos/vyos-1x   
Build  
Launch Docker container and build package   
# For VyOS 1.3 (equuleus, current)   
$ docker run --rm -it --privileged -v $(pwd):/vyos -w /vyos vyos/vyos-build:current bash

# Change to source directory   
$ cd vyos-1x

# Build DEB   
$ dpkg-buildpackage -uc -us -tc -b   
After a minute or two you will find the generated DEB packages next to the vyos-1x source directory:   
# ls -al ../vyos-1x\*.deb  
-rw-r--r-- 1 vyos\_bld vyos\_bld 567420 Aug 3 12:01 ../vyos-1x\_1.3dev0-1847-gb6dcb0a8\_all.deb-rw-r--r-- 1 vyos\_bld vyos\_bld 3808 Aug 3 12:01 ../vyos-1x-vmware\_1.3dev0-1847-  
gb6dcb0a8\_amd64.deb   
Install

To take your newly created package on a test drive you can simply SCP it to a running VyOS instance and install the new \*.deb package over the current running one.

Just install using the following commands:   
vyos@vyos:~$ dpkg --install /tmp/vyos-1x\_1.3dev0-1847-gb6dcb0a8\_all.deb (Reading database ... 58209 files and directories currently installed.)   
Preparing to unpack .../vyos-1x\_1.3dev0-1847-gb6dcb0a8\_all.deb ...

Unpacking vyos-1x (1.3dev0-1847-gb6dcb0a8) over (1.3dev0-1847-gb6dcb0a8) ...

Setting up vyos-1x (1.3dev0-1847-gb6dcb0a8) ...

Processing triggers for rsyslog (8.1901.0-1) ...

You can also place the generated \*.deb into your ISO build environment to include it in a custom iso, see Linux Kernel for more information.

Warning   
Any packages in the packages directory will be added to the iso during build, replacing the upstream ones. Make sure you delete them (both the source directories and built deb packages) if you want to build an iso from purely upstream packages.

VyOS cloud-init  
Cloud and virtualized instances of VyOS are initialized using the industry-standard cloud-init. Via cloud-init, the system performs tasks such as injecting SSH keys and configuring the network. In addition, the user can supply a custom configuration at the time of instance launch.

Config Sources  
VyOS support three types of config sources.

• Metadata - Metadata is sourced by the cloud platform or hypervisor. In some clouds, there is implemented as an HTTP endpoint at http://169.254.169.254.

• Network configuration - This config source informs the system about the network settings like IP addresses, routes, DNS. Available only in several cloud and virtualization platforms.

• User-data - User-data is specified by the user. This config source offers the ability to insert any CLI configuration commands into the configuration before the first boot.

User-data  
Major cloud providers offer a means of providing user-data at the time of instance launch. It can be provided as plain text or as base64-encoded text, depending on cloud provider. Also, it can be compressed using gzip, which makes sense with a long configuration commands list, because of the hard limit to ~16384 bytes for the whole user-data.

The easiest way to configure the system via user-data is the Cloud-config syntax described below.

Cloud-config modules  
In VyOS, by default, enables only two modules:  
• write\_files - this module allows to insert any files into the filesystem before the first boot, for example, pre-generated encryption keys, certificates, or even a whole config.boot file. The format is described in the cloudinit documentation Cloud-init-write\_files.

• vyos\_userdata - the module accepts a list of CLI configuration commands in a vyos\_config\_commands section, which gives an easy way to configure the system during deployment.

cloud-config file format

A cloud-config document is written in YAML. The file must begin with #cloud-config line. The only supported top-level keys are vyos\_config\_commands and write\_files. The use of these keys is described in the following two sections.

Initial Configuration  
The key used to designate a VyOS configuration is vyos\_config\_commands. What follows is VyOS configuration using the “set-style” syntax. Both “set” and “delete” commands are supported.

Commands requirements:  
• One command per line.

• If command ends in a value, it must be inside single quotes.

• A single-quote symbol is not allowed inside command or value.

The commands list produced by the show configuration commands command on a VyOS router should comply with all the requirements, so it is easy to get a proper commands list by copying it from another router.

The configuration specified in the cloud-config document overwrites default configuration values and values configured via Metadata.

Here is an example cloud-config that appends configuration at the time of first boot.

#cloud-config   
vyos\_config\_commands:  
 - set system host-name 'vyos-prod-ashburn'  
 - set service ntp server 1.pool.ntp.org  
 - set service ntp server 2.pool.ntp.org  
 - delete interfaces ethernet eth1 address 'dhcp'  
 - set interfaces ethernet eth1 address '192.0.2.247/24'  
 - set protocols static route 198.51.100.0/24 next-hop '192.0.2.1' System Defaults/Fallbacks  
These are the VyOS defaults and fallbacks.

• SSH is configured on port 22.

• vyos/vyos credentials if no others specified by data source.

• DHCP on first Ethernet interface if no network configuration is provided.

All of these can be overridden using the configuration in user-data.

Command Execution at Initial Boot  
VyOS supports the execution of operational commands and linux commands at initial boot. This is accomplished using write\_files to certain files in the /opt/vyatta/etc/config/scripts directory. Commands specified in opt/vyatta/etc/config/scripts/vyos-preconfig-bootup.script are executed prior to configuration. The /opt/vyatta/etc/config/scripts/vyos-postconfig-bootup.script file contains commands to be executed after configuration. In both cases, commands are executed as the root user.

Note that the /opt/vyatta/etc/config is used instead of the /config/scripts directory referenced in the Command Scripting section of the documentation because the /config/script directory isn’t mounted when the write\_files module executes.

The following example shows how to execute commands after the initial configuration.

#cloud-config   
write\_files:

- path: /opt/vyatta/etc/config/scripts/vyos-postconfig-bootup.script  
 owner: root:vyattacfg  
 permissions: '0775'  
 content: |  
 #!/bin/vbash  
 source /opt/vyatta/etc/functions/script-template  
 filename=/tmp/bgp\_status\_`date +"%Y\_%m\_%d\_%I\_%M\_%p"`.log  
 run show ip bgp summary >> $filename   
If you need to gather information from linux commands to configure VyOS, you can execute commands and then configure VyOS in the same script.

The following example sets the hostname based on the instance identifier obtained from the EC2 metadata service.

#cloud-config   
write\_files:  
 - path: /opt/vyatta/etc/config/scripts/vyos-postconfig-bootup.script  
 owner: root:vyattacfg  
 permissions: '0775'  
 content: |  
 #!/bin/vbash  
 source /opt/vyatta/etc/functions/script-template  
 hostname=`curl -s http://169.254.169.254/latest/meta-data/instance-id`  
 configure  
 set system host-name $hostname  
 commit  
 exit   
NoCloud  
Injecting configuration data is not limited to cloud platforms. Users can employ the NoCloud data source to inject user-data and meta-data on virtualization platforms such as VMware, Hyper-V and KVM.

While other methods exist, the most straightforward method for using the NoCloud data source is creating a seed ISO and attaching it to the virtual machine as a CD drive. The volume must be formatted as a vfat or ISO 9660 file system with the label “cidata” or “CIDATA”.

Create text files named user-data and meta-data. On linux-based systems, the mkisofs utility can be used to create the seed ISO. The following syntax will add these files to the ISO 9660 file system.

mkisofs -joliet -rock -volid "cidata" -output seed.iso meta-data user-data   
The seed.iso file can be attached to the virtual machine. As an example, the method with KVM to attach the ISO as a CD drive follows.

$ virt-install -n vyos\_r1 \  
 --ram 4096 \  
 --vcpus 2 \  
 --cdrom seed.iso \  
 --os-type linux \  
 --os-variant debian10 \

--network network=default \  
 --graphics vnc \  
 --hvm \  
 --virt-type kvm \  
 --disk path=/var/lib/libvirt/images/vyos\_kvm.qcow2,bus=virtio \  
 --import \  
 --noautoconsole   
For more information on the NoCloud data source, visit its page in the cloud-init documentation.

Troubleshooting  
If you encounter problems, verify that the cloud-config document contains valid YAML. Online resources such as https://www.yamllint.com/ provide a simple tool for validating YAML.

cloud-init logs to /var/log/cloud-init.log. This file can be helpful in determining why the configuration varies from what you expect. You can fetch the most important data filtering output for vyos keyword:   
sudo grep vyos /var/log/cloud-init.log   
Cloud-init on Proxmox  
Before starting, please refer to cloud-init network-config-docs in order to know how to import user and network configurations.

Most important keys that needs to be considered:  
• VyOS configuration commands are defined in user-data file.

• Networking configurations shouldn’t be passed in user-data file.

• If no networking configuration is provided, then dhcp client is going to be enabled on first interface. Bare in mind that this configuration will be inyected at an OS level, so don’t expect to find dhcp client configuration on vyos cli. Because of this behavior, in next example lab we will disable dhcp-client configuration on eth0.

Also, this lab considers:  
• Proxmox IP address: 192.168.0.253/24  
• Storaged used: volume local, which is mounted on directory /var/lib/vz, and contains all type of content, including snippets.

• Remove default dhcp client on first interface, and load other configuration during first boot, using cloud-init.

Generate qcow image  
A VyOS qcow image with cloud-init options is needed. This can be obtained using vyos-vm-images repo. After clonning the repo, edit the file qemu.yml and comment the download-iso role.

In this lab, we are using 1.3.0 VyOS version and setting a disk of 10G. Download VyOS .iso file and save it as /tmp/vyos.iso. Command used for generating qcow image:   
sudo ansible-playbook qemu.yml -e disk\_size=10 \  
 -e iso\_local=/tmp/vyos.iso -e grub\_console=serial -e vyos\_version=1.3.0 \  
 -e cloud\_init=true -e cloud\_init\_ds=NoCloud   
File generated with previous command: /tmp/vyos-1.3.0-cloud-init-10G-qemu.qcow2 Now, that file needs to be copied to proxmox server:   
sudo scp /tmp/vyos-1.3.0-cloud-init-10G-qemu.qcow2 root@192.168.0.253:/tmp/

Prepare cloud-init files  
In Proxmox server three files are going to be used for this setup:  
• network-config: file that will indicate to avoid dhcp client on first interface.

• user-data: includes vyos-commands.

• meta-data: empty file (required).

In this lab, all files are located in /tmp/. So, before going on, lets move to that directory: cd /tmp/   
user-data file must start with #cloud-config and contains vyos-commands. For example: #cloud-config   
vyos\_config\_commands:  
 - set system host-name 'vyos-BRAS'  
 - set service ntp server 1.pool.ntp.org  
 - set service ntp server 2.pool.ntp.org  
 - delete interfaces ethernet eth0 address 'dhcp'  
 - set interfaces ethernet eth0 address '198.51.100.2/30'  
 - set interfaces ethernet eth0 description 'WAN - ISP01'  
 - set interfaces ethernet eth1 address '192.168.25.1/24'  
 - set interfaces ethernet eth1 description 'Comming through VLAN 25'  
 - set interfaces ethernet eth2 address '192.168.26.1/24'  
 - set interfaces ethernet eth2 description 'Comming through VLAN 26'  
 - set protocols static route 0.0.0.0/0 next-hop '198.51.100.1'   
network-config file only has configuration that disables the automatic dhcp client on first interface.

Content of network-config file:   
version: 2   
ethernets:  
 eth0:  
 dhcp4: false  
 dhcp6: false   
Finaly, file meta-data has no content, but it’s required.

Create seed.iso  
Once the three files were created, it’s time to generate the seed.iso image, which needs to be mounted to the new VM as a cd.

Command for generating seed.iso   
mkisofs -joliet -rock -volid "cidata" -output seed.iso meta-data \   
user-data network-config   
NOTE: be carefull while copying and pasting previous commands. Doble quotes may need to be corrected.

Creating the VM  
Notes for this particular example, that may need to be modified in other setups:• VM ID: in this example, VM ID used is 555.

• VM Storage: local volume is used.

ISO files storage: local volume is used for .iso file storage. In this scenario local volume •  
type is set to directory, abd attached to /var/lib/vz.

• VM Resources: these parameters can be modified as needed.

seed.iso was previously created in directory /tmp/. It’s necessary to move it to /var/lib/vz/template/iso   
mv /tmp/seed.iso /var/lib/vz/template/iso/   
On proxmox server:   
## Create VM, import disk and define boot order   
qm create 555 --name vyos-1.3.0-cloudinit --memory 1024 --net0 virtio,bridge=vmbr0 qm importdisk 555 vyos-1.3.0-cloud-init-10G-qemu.qcow2 local   
qm set 555 --virtio0 local:555/vm-555-disk-0.raw   
qm set 555 --boot order=virtio0

## Import seed.iso for cloud init   
qm set 555 --ide2 media=cdrom,file=local:iso/seed.iso

## Since this server has 1 nic, lets add network intefaces (vlan 25 and 26)   
qm set 555 --net1 virtio,bridge=vmbr0,firewall=1,tag=25   
qm set 555 --net2 virtio,bridge=vmbr0,firewall=1,tag=26   
Power on VM and verifications  
From cli or GUI, power on VM, and after it boots, verify configuration   
Command Scripting  
VyOS supports executing configuration and operational commands non-interactively from shell scripts.

To include VyOS specific functions and aliases you need to source /opt/vyatta/etc/functions/script-template files at the top of your script.

#!/bin/vbash   
source /opt/vyatta/etc/functions/script-template   
exit   
Run configuration commands  
Configuration commands are executed just like from a normal config session. For example, if you want to disable a BGP peer on VRRP transition to backup:   
#!/bin/vbash   
source /opt/vyatta/etc/functions/script-template   
configure   
set protocols bgp system-as 65536   
set protocols bgp neighbor 192.168.2.1 shutdown   
commit   
exit   
Run operational commands  
Unlike a normal configuration session, all operational commands must be prepended with run, even if you haven’t created a session with configure.

#!/bin/vbash   
source /opt/vyatta/etc/functions/script-template run show interfaces   
exit

Run commands remotely  
Sometimes you simply wan’t to execute a bunch of op-mode commands via SSH on a remote VyOS system.

ssh 192.0.2.1 'vbash -s' <<EOF   
source /opt/vyatta/etc/functions/script-template   
run show interfaces   
exit   
EOF   
Will return:   
Welcome to VyOS   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 192.0.2.1/24 u/u   
lo 127.0.0.1/8 u/u  
 ::1/128   
Other script languages  
If you want to script the configs in a language other than bash you can have your script output commands and then source them in a bash script.

Here is a simple example:   
#!/usr/bin/env python3   
print("delete firewall group address-group somehosts")   
print("set firewall group address-group somehosts address '192.0.2.3'")   
print("set firewall group address-group somehosts address '203.0.113.55'")   
#!/bin/vbash   
source /opt/vyatta/etc/functions/script-template   
configure   
source < /config/scripts/setfirewallgroup.py   
commit   
Executing Configuration Scripts  
There is a pitfall when working with configuration scripts. It is tempting to call configuration scripts with “sudo” (i.e., temporary root permissions), because that’s the common way on most Linux platforms to call system commands.

On VyOS this will cause the following problem: After modifying the configuration via script like this once, it is not possible to manually modify the config anymore:   
sudo ./myscript.sh # Modifies config   
configure   
set ... # Any configuration parameter   
This will result in the following error message: Set failed If this happens, a reboot is required to be able to edit the config manually again.

To avoid these problems, the proper way is to call a script with the vyattacfg group, e.g., by using the sg (switch group) command:   
sg vyattacfg -c ./myscript.sh

To make sure that a script is not accidentally called without the vyattacfg group, the script can be safeguarded like this:   
if [ "$(id -g -n)" != 'vyattacfg' ] ; then  
 exec sg vyattacfg -c "/bin/vbash $(readlink -f $0) $@"   
fi   
Executing pre-hooks/post-hooks Scripts  
VyOS has the ability to run custom scripts before and after each commit   
The default directories where your custom Scripts should be located are:   
/config/scripts/commit/pre-hooks.d - Directory with scripts that run before  
 each commit.

/config/scripts/commit/post-hooks.d - Directory with scripts that run after each commit.

Scripts are run in alphabetical order. Their names must consist entirely of ASCII upper- and lower-case letters,ASCII digits, ASCII underscores, and ASCII minus-hyphens.No other characters are allowed.

Note   
Custom scripts are not executed with root privileges (Use sudo inside if this is necessary). A simple example is shown below, where the ops command executed in the post-hook script is “show interfaces”.

vyos@vyos# set interfaces ethernet eth1 address 192.0.2.3/24   
vyos@vyos# commit   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 198.51.100.10/24 u/u   
eth1 192.0.2.3/24 u/u   
eth2 - u/u   
eth3 - u/u   
lo 203.0.113.5/24 u/u   
Preconfig on boot  
The /config/scripts/vyos-preconfig-bootup.script script is called on boot before the VyOS configuration during boot process.

Any modifications were done to work around unfixed bugs and implement enhancements that are not complete in the VyOS system can be placed here.

The default file looks like this:   
#!/bin/sh   
# This script is executed at boot time before VyOS configuration is applied.

# Any modifications required to work around unfixed bugs or use   
# services not available through the VyOS CLI system can be placed here.

Postconfig on boot  
The /config/scripts/vyos-postconfig-bootup.script script is called on boot after the VyOS configuration is fully applied.

Any modifications were done to work around unfixed bugs and implement enhancements that are not complete in the VyOS system can be placed here.

The default file looks like this:   
#!/bin/sh   
# This script is executed at boot time after VyOS configuration is fully # applied. Any modifications required to work around unfixed bugs or use # services not available through the VyOS CLI system can be placed here.

Failover  
Failover routes are manually configured routes, but they install to the routing table if the health-check target is alive. If the target is not alive the route is removed from the routing table until the target will be available.

Failover Routes  
 set protocols failover route <subnet> next-hop <address> check target <target-address> Configure next-hop <address> and <target-address> for an IPv4 static route. Specify the target IPv4 address for health checking.

set protocols failover route <subnet> next-hop <address> check timeout <timeout> Timeout in seconds between health target checks.

Range is 1 to 300, default is 10.

set protocols failover route <subnet> next-hop <address> check type <protocol> Defines protocols for checking ARP, ICMP, TCP   
Default is icmp.

set protocols failover route <subnet> next-hop <address> check policy <policy>   
Policy for checking targets  
• all-available all checking target addresses must be available to pass this check• any-available any of the checking target addresses must be available to pass this check Default is any-available.

set protocols failover route <subnet> next-hop <address> interface <interface> Next-hop interface for the route  
 set protocols failover route <subnet> next-hop <address> metric <metric> Route metric   
Default 1.

Example  
One gateway:   
set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check target '192.0.2.1' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check timeout '5' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check type 'icmp' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 interface 'eth0' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 metric '10'   
Show the route   
vyos@vyos:~$ show ip route 203.0.113.1  
 Routing entry for 203.0.113.1/32  
 Known via "kernel", distance 0, metric 10, best  
 Last update 00:00:39 ago

\* 192.0.2.1, via eth0   
Two gateways and different metrics:   
set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check target '192.0.2.1' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check timeout '5' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 check type 'icmp' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 interface 'eth0' set protocols failover route 203.0.113.1/32 next-hop 192.0.2.1 metric '10'

set protocols failover route 203.0.113.1/32 next-hop 198.51.100.1 check target '198.51.100.99' set protocols failover route 203.0.113.1/32 next-hop 198.51.100.1 check timeout '5'   
set protocols failover route 203.0.113.1/32 next-hop 198.51.100.1 check type 'icmp'   
set protocols failover route 203.0.113.1/32 next-hop 198.51.100.1 interface 'eth2'   
set protocols failover route 203.0.113.1/32 next-hop 198.51.100.1 metric '20'   
Show the route   
vyos@vyos:~$ show ip route 203.0.113.1   
Routing entry for 203.0.113.1/32  
 Known via "kernel", distance 0, metric 10, best  
 Last update 00:08:06 ago  
 \* 192.0.2.1, via eth0

Routing entry for 203.0.113.1/32  
 Known via "kernel", distance 0, metric 20  
 Last update 00:08:14 ago  
 \* 198.51.100.1, via eth2   
IGMP Proxy  
IGMP proxy sends IGMP host messages on behalf of a connected client. The configuration must define one, and only one upstream interface, and one or more downstream interfaces.

Configuration  
 set protocols igmp-proxy interface <interface> role <upstream | downstream>  
• upstream: The upstream network interface is the outgoing interface which is responsible for communicating to available multicast data sources. There can only be one upstream interface.

• downstream: Downstream network interfaces are the distribution interfaces to the destination networks, where multicast clients can join groups and receive multicast data. One or more downstream interfaces must be configured.

set protocols igmp-proxy interface <interface> alt-subnet <network>   
Defines alternate sources for multicasting and IGMP data. The network address must be on the following format ‘a.b.c.d/n’. By default, the router will accept data from sources on the same network as configured on an interface. If the multicast source lies on a remote network, one must define from where traffic should be accepted.

This is especially useful for the upstream interface, since the source for multicast traffic is often from a remote location.

This option can be supplied multiple times.

set protocols igmp-proxy disable-quickleave

Disables quickleave mode. In this mode the daemon will not send a Leave IGMP message upstream as soon as it receives a Leave message for any downstream interface. The daemon will not ask for Membership reports on the downstream interfaces, and if a report is received the group is not joined again the upstream.

If it’s vital that the daemon should act exactly like a real multicast client on the upstream interface, this function should be enabled.

Enabling this function increases the risk of bandwidth saturation.

set protocols igmp-proxy disable   
Disable this service.

Example  
Interface eth1 LAN is behind NAT. In order to subscribe 10.0.0.0/23 subnet multicast which is in eth0 WAN we need to configure igmp-proxy.

set protocols igmp-proxy interface eth0 role upstream   
set protocols igmp-proxy interface eth0 alt-subnet 10.0.0.0/23 set protocols igmp-proxy interface eth1 role downstream Operation  
 restart igmp-proxy   
Restart the IGMP proxy process.

Information  
VyOS features a rich set of operational level commands to retrieve arbitrary information about your running system.

Hardware  
USB  
In the past serial interface have been defined as ttySx and ttyUSBx where x was an instance number of the serial interface. It was discovered that from system boot to system boot the mapping of USB based serial interfaces will differ, depending which driver was loaded first by the operating system. This will become rather painful if you not only have serial interfaces for a console server connected but in addition also a serial backed WWAN - Wireless Wide-Area-Network.

To overcome this issue and the fact that in almost 50% of all cheap USB to serial converters there is no serial number programmed, the USB to serial interface is now directly identified by the USB root bridge and bus it connects to. This somehow mimics the new network interface definitions we see in recent Linux distributions.

For additional details you can refer to https://vyos.dev/T2490.

show hardware usb   
Retrieve a tree like representation of all connected USB devices.

Note   
If a device is unplugged and re-plugged it will receive a new Port, Dev, If identification.

vyos@vyos:~$ show hardware usb   
/: Bus 03.Port 1: Dev 1, Class=root\_hub, Driver=ehci-pci/2p, 480M  
 |\_\_ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M  
 |\_\_ Port 3: Dev 4, If 0, Class=Vendor Specific Class, Driver=qcserial, 480M |\_\_ Port 3: Dev 4, If 2, Class=Vendor Specific Class, Driver=qcserial, 480M |\_\_ Port 3: Dev 4, If 3, Class=Vendor Specific Class, Driver=qcserial, 480M

|\_\_ Port 3: Dev 4, If 8, Class=Vendor Specific Class, Driver=qmi\_wwan, 480M   
/: Bus 02.Port 1: Dev 1, Class=root\_hub, Driver=xhci\_hcd/2p, 5000M   
/: Bus 01.Port 1: Dev 1, Class=root\_hub, Driver=xhci\_hcd/2p, 480M  
 |\_\_ Port 1: Dev 2, If 0, Class=Vendor Specific Class, Driver=pl2303, 12M  
 |\_\_ Port 2: Dev 3, If 0, Class=Hub, Driver=hub/4p, 480M  
 |\_\_ Port 4: Dev 5, If 2, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 5, If 0, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 5, If 3, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 5, If 1, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 3: Dev 4, If 0, Class=Hub, Driver=hub/4p, 480M  
 |\_\_ Port 3: Dev 6, If 0, Class=Hub, Driver=hub/4p, 480M  
 |\_\_ Port 4: Dev 8, If 2, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 8, If 0, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 8, If 3, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 8, If 1, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 7, If 3, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 7, If 1, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 7, If 2, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 |\_\_ Port 4: Dev 7, If 0, Class=Vendor Specific Class, Driver=ftdi\_sio, 480M  
 show hardware usb serial   
Retrieve a list and description of all connected USB serial devices. The device name displayed, e.g. usb0b2.4p1.0 can be directly used when accessing the serial console as console-server device.

vyos@vyos$ show hardware usb serial   
Device Model Vendor  
------ ------ ------  
usb0b1.3p1.0 MC7710 Sierra Wireless, Inc.

usb0b1.3p1.2 MC7710 Sierra Wireless, Inc.

usb0b1.3p1.3 MC7710 Sierra Wireless, Inc.

usb0b1p1.0 USB-Serial\_Controller\_D Prolific Technology, Inc.

usb0b2.3.3.4p1.0 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.3.4p1.1 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.3.4p1.2 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.3.4p1.3 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.4p1.0 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.4p1.1 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.4p1.2 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.3.4p1.3 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.4p1.0 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.4p1.1 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.4p1.2 Quad\_RS232-HS Future Technology Devices International, Ltd usb0b2.4p1.3 Quad\_RS232-HS Future Technology Devices International, Ltd Version  
 show version

Return the current running VyOS version and build information. This includes also the name of the release train which is crux on VyOS 1.2, equuleus on VyOS 1.3 and sagitta on VyOS 1.4.

vyos@vyos:~$ show version

Version: VyOS 1.4-rolling-202106270801   
Release Train: sagitta

Built by: autobuild@vyos.net   
Built on: Sun 27 Jun 2021 09:50 UTC   
Build UUID: ab43e735-edcb-405a-9f51-f16a1b104e52 Build Commit ID: f544d75eab758f

Architecture: x86\_64   
Boot via: installed image   
System type: KVM guest

Hardware vendor: QEMU   
Hardware model: Standard PC (i440FX + PIIX, 1996) Hardware S/N:   
Hardware UUID: Unknown

Copyright: VyOS maintainers and contributors  
 show version kernel   
Return version number of the Linux Kernel used in this release.

vyos@vyos:~$ show version kernel   
5.10.46-amd64-vyos  
 show version frr   
Return version number of FRR (Free Range Routing - https://frrouting.org/) used in this release. This is the routing control plane and a successor to GNU Zebra and Quagga.

vyos@vyos:~$ show version frr   
FRRouting 7.5.1-20210625-00-gf07d935a2 (vyos).

Copyright 1996-2005 Kunihiro Ishiguro, et al.

IPsec  
GRE, GRE/IPsec (or IPIP/IPsec, SIT/IPsec, or any other stateless tunnel protocol over IPsec) is the usual way to protect the traffic inside a tunnel.

An advantage of this scheme is that you get a real interface with its own address, which makes it easier to setup static routes or use dynamic routing protocols without having to modify IPsec policies. The other advantage is that it greatly simplifies router to router communication, which can be tricky with plain IPsec because the external outgoing address of the router usually doesn’t match the IPsec policy of typical site-to-site setup and you need to add special configuration for it, or adjust the source address for outgoing traffic of your applications. GRE/IPsec has no such problem and is completely transparent for the applications.

GRE/IPIP/SIT and IPsec are widely accepted standards, which make this scheme easy to implement between VyOS and virtually any other router.

For simplicity we’ll assume that the protocol is GRE, it’s not hard to guess what needs to be changed to make it work with a different protocol. We assume that IPsec will use pre-shared secret authentication and will use AES128/SHA1 for the cipher and hash. Adjust this as necessary.

Note   
VMware users should ensure that a VMXNET3 adapter is used. E1000 adapters have known issues with GRE processing.

IKE (Internet Key Exchange) Attributes  
IKE performs mutual authentication between two parties and establishes an IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) or Authentication Header (AH) and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.

https://datatracker.ietf.org/doc/html/rfc5996   
In VyOS, IKE attributes are specified through IKE groups. Multiple proposals can be specified in a single group.

VyOS IKE group has the next options:  
• close-action defines the action to take if the remote peer unexpectedly closes a CHILD\_SA:  
• none set action to none (default);  
• hold set action to hold;   
 restart set action to restart;•  
• dead-peer-detection controls the use of the Dead Peer Detection protocol (DPD, RFC 3706) where R\_U\_THERE notification messages (IKEv1) or empty INFORMATIONAL messages (IKEv2) are periodically sent in order to check the liveliness of the IPsec peer:  
• action keep-alive failure action:   
 hold set action to hold (default)•  
• clear set action to clear;  
• restart set action to restart;  
• interval keep-alive interval in seconds <2-86400> (default 30);   
 timeout keep-alive timeout in seconds <2-86400> (default 120) IKEv1 only•  
• ikev2-reauth whether rekeying of an IKE\_SA should also reauthenticate the peer. In IKEv1, reauthentication is always done. Setting this parameter enables remote host re-authentication during an IKE rekey.

• key-exchange which protocol should be used to initialize the connection If not set both protocols are handled and connections will use IKEv2 when initiating, but accept any protocol version when responding:  
• ikev1 use IKEv1 for Key Exchange;  
• ikev2 use IKEv2 for Key Exchange;  
• lifetime IKE lifetime in seconds <0-86400> (default 28800);   
 disable-mobike disables MOBIKE Support. MOBIKE is only available for IKEv2 and •  
enabled by default.

• mode IKEv1 Phase 1 Mode Selection:  
• main use Main mode for Key Exchanges in the IKEv1 Protocol (Recommended Default);

• aggressive use Aggressive mode for Key Exchanges in the IKEv1 protocol aggressive

mode is much more insecure compared to Main mode;

• proposal the list of proposals and their parameters:

• dh-group dh-group;

• encryption encryption algorithm;

• hash hash algorithm.

• prf pseudo-random function.

ESP (Encapsulating Security Payload) Attributes

ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an

anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.

https://datatracker.ietf.org/doc/html/rfc4303

In VyOS, ESP attributes are specified through ESP groups. Multiple proposals can be specified in

a single group.

VyOS ESP group has the next options:

• compression Enables the IPComp(IP Payload Compression) protocol which allows

compressing the content of IP packets.

• life-bytes ESP life in bytes <1024-26843545600000>. Number of bytes transmitted over

an IPsec SA before it expires;

• life-packets ESP life in packets <1000-26843545600000>. Number of packets transmitted

over an IPsec SA before it expires;

• lifetime ESP lifetime in seconds <30-86400> (default 3600). How long a particular

instance of a connection (a set of encryption/authentication keys for user packets) should last,

from successful negotiation to expiry;

• mode the type of the connection:

• tunnel tunnel mode (default);

• transport transport mode;

• pfs whether Perfect Forward Secrecy of keys is desired on the connection’s keying

channel and defines a Diffie-Hellman group for PFS:

• enable Inherit Diffie-Hellman group from IKE group (default);

• disable Disable PFS;

• < dh-group > defines a Diffie-Hellman group for PFS;

• proposal ESP-group proposal with number <1-65535>:

• encryption encryption algorithm (default 128 bit AES-CBC);

• hash hash algorithm (default sha1).

Options (Global IPsec settings) Attributes

• options

• disable-route-autoinstall Do not automatically install routes to remote networks;

• flexvpn Allows FlexVPN vendor ID payload (IKEv2 only). Send the Cisco FlexVPN vendor

ID payload (IKEv2 only), which is required in order to make Cisco brand devices allow

negotiating a local traffic selector (from strongSwan’s point of view) that is not the assigned

virtual IP address if such an address is requested by strongSwan. Sending the Cisco FlexVPN

vendor ID prevents the peer from narrowing the initiator’s local traffic selector and allows it to

e.g. negotiate a TS of 0.0.0.0/0 == 0.0.0.0/0 instead. This has been tested with a “tunnel mode

ipsec ipv4” Cisco template but should also work for GRE encapsulation;

• interface Interface Name to use. The name of the interface on which virtual IP addresses should be installed. If not specified the addresses will be installed on the outbound interface;• virtual-ip Allows to install virtual-ip addresses. Comma separated list of virtual IPs to request in IKEv2 configuration payloads or IKEv1 Mode Config. The wildcard addresses 0.0.0.0 and :: request an arbitrary address, specific addresses may be defined. The responder may return a different address, though, or none at all. Define the virtual-address option to configure the IP address in site-to-site hierarchy.

IPsec policy matching GRE  
The first and arguably cleaner option is to make your IPsec policy match GRE packets between external addresses of your routers. This is the best option if both routers have static external addresses.

Suppose the LEFT router has external address 192.0.2.10 on its eth0 interface, and the RIGHT router is 203.0.113.45   
On the LEFT:   
# GRE tunnel   
set interfaces tunnel tun0 encapsulation gre   
set interfaces tunnel tun0 source-address 192.0.2.10   
set interfaces tunnel tun0 remote 203.0.113.45   
set interfaces tunnel tun0 address 10.10.10.1/30

## IPsec   
set vpn ipsec interface eth0

# Pre-shared-secret   
set vpn ipsec authentication psk vyos id 192.0.2.10   
set vpn ipsec authentication psk vyos id 203.0.113.45 set vpn ipsec authentication psk vyos secret MYSECRETKEY

# IKE group   
set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group '2'   
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption 'aes128' set vpn ipsec ike-group MyIKEGroup proposal 1 hash 'sha1'

# ESP group   
set vpn ipsec esp-group MyESPGroup proposal 1 encryption 'aes128' set vpn ipsec esp-group MyESPGroup proposal 1 hash 'sha1'

# IPsec tunnel   
set vpn ipsec site-to-site peer right authentication mode pre-shared-secret set vpn ipsec site-to-site peer right authentication remote-id 203.0.113.45

set vpn ipsec site-to-site peer right ike-group MyIKEGroup   
set vpn ipsec site-to-site peer right default-esp-group MyESPGroup

set vpn ipsec site-to-site peer right local-address 192.0.2.10 set vpn ipsec site-to-site peer right remote-address 203.0.113.45

# This will match all GRE traffic to the peer   
set vpn ipsec site-to-site peer right tunnel 1 protocol gre   
On the RIGHT, setup by analogy and swap local and remote addresses.

Source tunnel from loopbacks  
The scheme above doesn’t work when one of the routers has a dynamic external address though. The classic workaround for this is to setup an address on a loopback interface and use it as a source address for the GRE tunnel, then setup an IPsec policy to match those loopback addresses.

We assume that the LEFT router has static 192.0.2.10 address on eth0, and the RIGHT router has a dynamic address on eth0.

The peer names RIGHT and LEFT are used as informational text.

Setting up the GRE tunnel   
On the LEFT:   
set interfaces loopback lo address 192.168.99.1/32

set interfaces tunnel tun0 encapsulation gre   
set interfaces tunnel tun0 address 10.10.10.1/30   
set interfaces tunnel tun0 source-address 192.168.99.1 set interfaces tunnel tun0 remote 192.168.99.2   
On the RIGHT:   
set interfaces loopback lo address 192.168.99.2/32

set interfaces tunnel tun0 encapsulation gre   
set interfaces tunnel tun0 address 10.10.10.2/30   
set interfaces tunnel tun0 source-address 192.168.99.2   
set interfaces tunnel tun0 remote 192.168.99.1   
Setting up IPSec   
However, now you need to make IPsec work with dynamic address on one side. The tricky part is that pre-shared secret authentication doesn’t work with dynamic address, so we’ll have to use RSA keys.

First, on both routers run the operational command “generate pki key-pair install <key-pair name>”. You may choose different length than 2048 of course.

vyos@left# run generate pki key-pair install ipsec-LEFT   
Enter private key type: [rsa, dsa, ec] (Default: rsa)   
Enter private key bits: (Default: 2048)   
Note: If you plan to use the generated key on this router, do not encrypt the private key.

Do you want to encrypt the private key with a passphrase? [y/N] N Configure mode commands to install key pair:   
Do you want to install the public key? [Y/n] Y   
set pki key-pair ipsec-LEFT public key 'MIIBIjANBgkqh...'   
Do you want to install the private key? [Y/n] Y

set pki key-pair ipsec-LEFT private key 'MIIEvgIBADAN...'   
[edit]   
Configuration commands for the private and public key will be displayed on the screen which needs to be set on the router first. Note the command with the public key (set pki key-pair ipsec-LEFT public key ‘MIIBIjANBgkqh…’). Then do the same on the opposite router:   
vyos@left# run generate pki key-pair install ipsec-RIGHT   
Note the command with the public key (set pki key-pair ipsec-RIGHT public key   
‘FAAOCAQ8AMII…’).

Now the noted public keys should be entered on the opposite routers.

On the LEFT:   
set pki key-pair ipsec-RIGHT public key 'FAAOCAQ8AMII...'   
On the RIGHT:   
set pki key-pair ipsec-LEFT public key 'MIIBIjANBgkqh...'   
Now you are ready to setup IPsec. You’ll need to use an ID instead of address for the peer.

On the LEFT (static address):   
set vpn ipsec interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128 set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1

set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2   
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128 set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer RIGHT authentication local-id LEFT   
set vpn ipsec site-to-site peer RIGHT authentication mode rsa   
set vpn ipsec site-to-site peer RIGHT authentication rsa local-key ipsec-LEFT   
set vpn ipsec site-to-site peer RIGHT authentication rsa remote-key ipsec-RIGHT   
set vpn ipsec site-to-site peer RIGHT authentication remote-id RIGHT   
set vpn ipsec site-to-site peer RIGHT default-esp-group MyESPGroup   
set vpn ipsec site-to-site peer RIGHT ike-group MyIKEGroup   
set vpn ipsec site-to-site peer RIGHT local-address 192.0.2.10   
set vpn ipsec site-to-site peer RIGHT connection-type respond   
set vpn ipsec site-to-site peer RIGHT tunnel 1 local prefix 192.168.99.1/32 # Additional loopback address on the local   
set vpn ipsec site-to-site peer RIGHT tunnel 1 remote prefix 192.168.99.2/32 # Additional loopback address on the remote   
On the RIGHT (dynamic address):   
set vpn ipsec interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128 set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1

set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2

set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128 set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer LEFT authentication local-id RIGHT   
set vpn ipsec site-to-site peer LEFT authentication mode rsa   
set vpn ipsec site-to-site peer LEFT authentication rsa local-key ipsec-RIGHT   
set vpn ipsec site-to-site peer LEFT authentication rsa remote-key ipsec-LEFT   
set vpn ipsec site-to-site peer LEFT authentication remote-id LEFT   
set vpn ipsec site-to-site peer LEFT connection-type initiate   
set vpn ipsec site-to-site peer LEFT default-esp-group MyESPGroup   
set vpn ipsec site-to-site peer LEFT ike-group MyIKEGroup   
set vpn ipsec site-to-site peer LEFT local-address any   
set vpn ipsec site-to-site peer LEFT remote-address 192.0.2.10   
set vpn ipsec site-to-site peer LEFT tunnel 1 local prefix 192.168.99.2/32 # Additional loopback address on the local   
set vpn ipsec site-to-site peer LEFT tunnel 1 remote prefix 192.168.99.1/32 # Additional loopback address on the remote   
IS-IS  
IS-IS is a link-state interior gateway protocol (IGP) which is described in ISO10589, RFC 1195, RFC 5308. IS-IS runs the Dijkstra shortest-path first (SPF) algorithm to create a database of the network’s topology, and from that database to determine the best (that is, lowest cost) path to a destination. The intermediate systems (the name for routers) exchange topology information with their directly conencted neighbors. IS-IS runs directly on the data link layer (Layer 2). IS-IS addresses are called NETs and can be 8 to 20 bytes long, but are generally 10 bytes long. The tree database that is created with IS-IS is similar to the one that is created with OSPF in that the paths chosen should be similar. Comparisons to OSPF are inevitable and often are reasonable ones to make in regards to the way a network will respond with either IGP.

General  
Configuration  
Mandatory Settings  
For IS-IS top operate correctly, one must do the equivalent of a Router ID in CLNS. This Router ID is called the NET. This must be unique for each and every router that is operating in IS-IS. It also must not be duplicated otherwise the same issues that occur within OSPF will occur within IS-IS when it comes to said duplication.

set protocols isis net <network-entity-title>   
This commad sets network entity title (NET) provided in ISO format.

Here is an example NET value:   
49.0001.1921.6800.1002.00   
The CLNS address consists of the following parts:  
• AFI - 49 The AFI value 49 is what IS-IS uses for private addressing.

• Area identifier: 0001 IS-IS area number (numberical area 1)  
• System identifier: 1921.6800.1002 - for system idetifiers we recommend to use IP address or MAC address of the router itself. The way to construct this is to keep all of the zeroes of the router IP address, and then change the periods from being every three numbers to every

four numbers. The address that is listed here is 192.168.1.2, which if expanded will turn into 192.168.001.002. Then all one has to do is move the dots to have four numbers instead of three. This gives us 1921.6800.1002.

• NET selector: 00 Must always be 00. This setting indicates “this system” or “local system.”  
 set protocols isis interface <interface>   
This command enables IS-IS on this interface, and allows for adjacency to occur. Note that the name of IS-IS instance must be the same as the one used to configure the IS-IS process.

IS-IS Global Configuration  
 set protocols isis dynamic-hostname   
This command enables support for dynamic hostname TLV. Dynamic hostname mapping determined as described in RFC 2763, Dynamic Hostname Exchange Mechanism for IS-IS.

set protocols isis level <level-1|level-1-2|level-2>   
This command defines the IS-IS router behavior:   
 level-1 - Act as a station (Level 1) router only.•  
• level-1-2 - Act as a station (Level 1) router and area (Level 2) router.

• level-2-only - Act as an area (Level 2) router only.

set protocols isis lsp-mtu <size>   
This command configures the maximum size of generated LSPs, in bytes. The size range is 128 to 4352.

set protocols isis metric-style <narrow|transition|wide>   
This command sets old-style (ISO 10589) or new style packet formats:• narrow - Use old style of TLVs with narrow metric.

• transition - Send and accept both styles of TLVs during transition.

• wide - Use new style of TLVs to carry wider metric.

set protocols isis purge-originator   
This command enables RFC 6232 purge originator identification. Enable purge originator identification (POI) by adding the type, length and value (TLV) with the Intermediate System (IS) identification to the LSPs that do not contain POI information. If an IS generates a purge, VyOS adds this TLV with the system ID of the IS to the purge.

set protocols isis set-attached-bit   
This command sets ATT bit to 1 in Level1 LSPs. It is described in RFC 3787.

set protocols isis set-overload-bit   
This command sets overload bit to avoid any transit traffic through this router. It is described in RFC 3787.

set protocols isis name default-information originate <ipv4|ipv6> level-1 This command will generate a default-route in L1 database.

set protocols isis name default-information originate <ipv4|ipv6> level-2 This command will generate a default-route in L2 database.

set protocols isis ldp-sync   
This command will enable IGP-LDP synchronization globally for ISIS. This requires for LDP to be functional. This is described in RFC 5443. By default all interfaces operational in IS-IS are enabled for synchronization. Loopbacks are exempt.

set protocols isis ldp-sync holddown <seconds>

This command will change the hold down value globally for IGP-LDP synchronization during convergence/interface flap events.

Interface Configuration  
 set protocols isis interface <interface> circuit-type <level-1|level-1-2|level-2-only> This command specifies circuit type for interface:  
• level-1 - Level-1 only adjacencies are formed.

level-1-2 - Level-1-2 adjacencies are formed•  
• level-2-only - Level-2 only adjacencies are formed  
 set protocols isis interface <interface> hello-interval <seconds>   
This command sets hello interval in seconds on a given interface. The range is 1 to 600.

set protocols isis interface <interface> hello-multiplier <seconds>   
This command sets multiplier for hello holding time on a given interface. The range is 2 to 100.

set protocols isis interface <interface> hello-padding   
This command configures padding on hello packets to accommodate asymmetrical maximum transfer units (MTUs) from different hosts as described in RFC 3719. This helps to prevent a premature adjacency Up state when one routing devices MTU does not meet the requirements to establish the adjacency.

set protocols isis interface <interface> metric <metric> This command set default metric for circuit.

The metric range is 1 to 16777215 (Max value depend if metric support narrow or wide value).

set protocols isis interface <interface> network point-to-point   
This command specifies network type to Point-to-Point. The default network type is broadcast.

set protocols isis interface <interface> passive   
This command configures the passive mode for this interface.

set protocols isis interface <interface> password plaintext-password <text> This command configures the authentication password for the interface.

set protocols isis interface <interface> priority <number>   
This command sets priority for the interface for DIS election. The priority range is 0 to 127.

set protocols isis interface <interface> psnp-interval <number>   
This command sets PSNP interval in seconds. The interval range is 0 to 127.

set protocols isis interface <interface> no-three-way-handshake   
This command disables Three-Way Handshake for P2P adjacencies which described in RFC 5303. Three-Way Handshake is enabled by default.

set protocols isis interface <interface> ldp-sync disable   
This command disables IGP-LDP sync for this specific interface.

set protocols isis interface <interface> ldp-sync holddown <seconds>   
This command will change the hold down value for IGP-LDP synchronization during convergence/interface flap events, but for this interface only.

Route Redistribution  
 set protocols isis redistribute ipv4 <route source> level-1   
This command redistributes routing information from the given route source into the ISIS database as Level-1. There are six modes available for route source: bgp, connected, kernel, ospf, rip, static.

set protocols isis redistribute ipv4 <route source> level-2

This command redistributes routing information from the given route source into the ISIS database as Level-2. There are six modes available for route source: bgp, connected, kernel, ospf, rip, static.

set protocols isis redistribute ipv4 <route source> <level-1|level-2> metric <number>   
This command specifies metric for redistributed routes from the given route source. There are six modes available for route source: bgp, connected, kernel, ospf, rip, static. The metric range is 1 to 16777215.

set protocols isis redistribute ipv4 <route source> <level-1|level-2> route-map <name> This command allows to use route map to filter redistributed routes from the given route source. There are six modes available for route source: bgp, connected, kernel, ospf, rip, static.

Timers  
 set protocols isis lsp-gen-interval <seconds>   
This command sets minimum interval in seconds between regenerating same LSP. The interval range is 1 to 120.

set protocols isis lsp-refresh-interval <seconds>   
This command sets LSP refresh interval in seconds. IS-IS generates LSPs when the state of a link changes. However, to ensure that routing databases on all routers remain converged, LSPs in stable networks are generated on a regular basis even though there has been no change to the state of the links. The interval range is 1 to 65235. The default value is 900 seconds.

set protocols isis max-lsp-lifetime <seconds>   
This command sets LSP maximum LSP lifetime in seconds. The interval range is 350 to 65535.

LSPs remain in a database for 1200 seconds by default. If they are not refreshed by that time, they are deleted. You can change the LSP refresh interval or the LSP lifetime. The LSP refresh interval should be less than the LSP lifetime or else LSPs will time out before they are refreshed.

set protocols isis spf-interval <seconds>   
This command sets minimum interval between consecutive SPF calculations in seconds.The interval range is 1 to 120.

set protocols isis spf-delay-ietf holddown <milliseconds>  
 set protocols isis spf-delay-ietf init-delay <milliseconds>  
 set protocols isis spf-delay-ietf long-delay <milliseconds>  
 set protocols isis spf-delay-ietf short-delay <milliseconds>  
 set protocols isis spf-delay-ietf time-to-learn <milliseconds>   
This commands specifies the Finite State Machine (FSM) intended to control the timing of the execution of SPF calculations in response to IGP events. The process described in RFC 8405.

Examples  
Enable IS-IS  
Node 1:   
set interfaces loopback lo address '192.168.255.255/32' set interfaces ethernet eth1 address '192.0.2.1/24'

set protocols isis interface eth1   
set protocols isis interface lo   
set protocols isis net '49.0001.1921.6825.5255.00' Node 2:

set interfaces ethernet eth1 address '192.0.2.2/24'

set interfaces loopback lo address '192.168.255.254/32' set interfaces ethernet eth1 address '192.0.2.2/24'

set protocols isis interface eth1   
set protocols isis interface lo   
set protocols isis net '49.0001.1921.6825.5254.00'   
This gives us the following neighborships, Level 1 and Level 2: Node-1@vyos:~$ show isis neighbor   
Area VyOS:  
 System Id Interface L State Holdtime SNPA  
 vyos eth1 1 Up 28 0c87.6c09.0001 vyos eth1 2 Up 28 0c87.6c09.0001

Node-2@vyos:~$ show isis neighbor   
Area VyOS:  
 System Id Interface L State Holdtime SNPA  
 vyos eth1 1 Up 29 0c33.0280.0001  
 vyos eth1 2 Up 28 0c33.0280.0001   
Here’s the IP routes that are populated. Just the loopback:   
Node-1@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.2, eth1 inactive, weight 1, 00:02:22 I>\* 192.168.255.254/32 [115/20] via 192.0.2.2, eth1, weight 1, 00:02:22

Node-2@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.1, eth1 inactive, weight 1, 00:02:21 I>\* 192.168.255.255/32 [115/20] via 192.0.2.1, eth1, weight 1, 00:02:21 Enable IS-IS and redistribute routes not natively in IS-IS  
Node 1:

set interfaces dummy dum0 address '203.0.113.1/24' set interfaces ethernet eth1 address '192.0.2.1/24'

set policy prefix-list EXPORT-ISIS rule 10 action 'permit'   
set policy prefix-list EXPORT-ISIS rule 10 prefix '203.0.113.0/24'   
set policy route-map EXPORT-ISIS rule 10 action 'permit'   
set policy route-map EXPORT-ISIS rule 10 match ip address prefix-list 'EXPORT-ISIS'

set protocols isis interface eth1   
set protocols isis net '49.0001.1921.6800.1002.00'   
set protocols isis redistribute ipv4 connected level-2 route-map 'EXPORT-ISIS' Node 2:   
set interfaces ethernet eth1 address '192.0.2.2/24'

set protocols isis interface eth1   
set protocols isis net '49.0001.1921.6800.2002.00'   
Routes on Node 2:   
Node-2@r2:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,  
 F - PBR, f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued route, r - rejected route

I 203.0.113.0/24 [115/10] via 192.0.2.1, eth1, 00:03:42 Enable IS-IS and IGP-LDP synchronization  
Node 1:   
set interfaces loopback lo address 192.168.255.255/32 set interfaces ethernet eth0 address 192.0.2.1/24

set protocols isis interface eth0   
set protocols isis interface lo passive   
set protocols isis ldp-sync   
set protocols isis net 49.0001.1921.6825.5255.00

set protocols mpls interface eth0   
set protocols mpls ldp discovery transport-ipv4-address 192.168.255.255   
set protocols mpls ldp interface lo   
set protocols mpls ldp interface eth0   
set protocols mpls ldp parameters transport-prefer-ipv4   
set protocols mpls ldp router-id 192.168.255.255   
This gives us IGP-LDP synchronization for all non-loopback interfaces with a holddown timer of zero seconds:   
Node-1@vyos:~$ show isis mpls ldp-sync

eth0  
 LDP-IGP Synchronization enabled: yes  
 holddown timer in seconds: 0  
 State: Sync achieved   
Enable IS-IS with Segment Routing (Experimental)Node 1:   
set interfaces loopback lo address '192.168.255.255/32' set interfaces ethernet eth1 address '192.0.2.1/24'

set protocols isis interface eth1   
set protocols isis interface lo   
set protocols isis net '49.0001.1921.6825.5255.00'   
set protocols isis segment-routing global-block high-label-value '599'   
set protocols isis segment-routing global-block low-label-value '550'   
set protocols isis segment-routing prefix 192.168.255.255/32 index value '1' set protocols isis segment-routing prefix 192.168.255.255/32 index explicit-null set protocols mpls interface 'eth1'   
Node 2:   
set interfaces loopback lo address '192.168.255.254/32'   
set interfaces ethernet eth1 address '192.0.2.2/24'

set protocols isis interface eth1   
set protocols isis interface lo   
set protocols isis net '49.0001.1921.6825.5254.00'   
set protocols isis segment-routing global-block high-label-value '599'   
set protocols isis segment-routing global-block low-label-value '550'   
set protocols isis segment-routing prefix 192.168.255.254/32 index value '2'   
set protocols isis segment-routing prefix 192.168.255.254/32 index explicit-null   
set protocols mpls interface 'eth1'   
This gives us MPLS segment routing enabled and labels for far end loopbacks:   
Node-1@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 ----------------------------------------------------------------------  
 552 SR (IS-IS) 192.0.2.2 IPv4 Explicit Null <-- Node-2 loopback learned on Node-1 15000 SR (IS-IS) 192.0.2.2 implicit-null  
 15001 SR (IS-IS) fe80::e87:6cff:fe09:1 implicit-null  
 15002 SR (IS-IS) 192.0.2.2 implicit-null  
 15003 SR (IS-IS) fe80::e87:6cff:fe09:1 implicit-null

Node-2@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 ---------------------------------------------------------------------  
 551 SR (IS-IS) 192.0.2.1 IPv4 Explicit Null <-- Node-1 loopback learned on Node-2 15000 SR (IS-IS) 192.0.2.1 implicit-null

15001 SR (IS-IS) fe80::e33:2ff:fe80:1 implicit-null  
 15002 SR (IS-IS) 192.0.2.1 implicit-null  
 15003 SR (IS-IS) fe80::e33:2ff:fe80:1 implicit-null   
Here is the routing tables showing the MPLS segment routing label operations: Node-1@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup  
 t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.2, eth1 inactive, weight 1, 00:07:48   
I>\* 192.168.255.254/32 [115/20] via 192.0.2.2, eth1, label IPv4 Explicit Null, weight 1, 00:03:39

Node-2@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.1, eth1 inactive, weight 1, 00:07:46   
I>\* 192.168.255.255/32 [115/20] via 192.0.2.1, eth1, label IPv4 Explicit Null, weight 1, 00:03:43 L2TP  
VyOS utilizes accel-ppp to provide L2TP server functionality. It can be used with local   
authentication or a connected RADIUS server.

L2TP over IPsec  
Example for configuring a simple L2TP over IPsec VPN for remote access (works with native Windows and Mac VPN clients):   
set vpn ipsec interface eth0

set vpn l2tp remote-access outside-address 192.0.2.2   
set vpn l2tp remote-access client-ip-pool start 192.168.255.2   
set vpn l2tp remote-access client-ip-pool stop 192.168.255.254   
set vpn l2tp remote-access ipsec-settings authentication mode pre-shared-secret set vpn l2tp remote-access ipsec-settings authentication pre-shared-secret <secret> set vpn l2tp remote-access authentication mode local   
set vpn l2tp remote-access authentication local-users username test password 'test' In the above example, an external IP of 192.0.2.2 is assumed.

If a local firewall policy is in place on your external interface you will need to allow the ports below:  
• UDP port 500 (IKE)

• IP protocol number 50 (ESP)  
• UDP port 1701 for IPsec   
As well as the below to allow NAT-traversal (when NAT is detected by the VPN client, ESP is encapsulated in UDP for NAT-traversal):  
• UDP port 4500 (NAT-T)   
Example:   
set firewall name OUTSIDE-LOCAL rule 40 action 'accept'   
set firewall name OUTSIDE-LOCAL rule 40 protocol 'esp'   
set firewall name OUTSIDE-LOCAL rule 41 action 'accept'   
set firewall name OUTSIDE-LOCAL rule 41 destination port '500'   
set firewall name OUTSIDE-LOCAL rule 41 protocol 'udp'   
set firewall name OUTSIDE-LOCAL rule 42 action 'accept'   
set firewall name OUTSIDE-LOCAL rule 42 destination port '4500'   
set firewall name OUTSIDE-LOCAL rule 42 protocol 'udp'   
set firewall name OUTSIDE-LOCAL rule 43 action 'accept'   
set firewall name OUTSIDE-LOCAL rule 43 destination port '1701'   
set firewall name OUTSIDE-LOCAL rule 43 ipsec 'match-ipsec'   
set firewall name OUTSIDE-LOCAL rule 43 protocol 'udp'   
To allow VPN-clients access via your external address, a NAT rule is required:   
set nat source rule 110 outbound-interface 'eth0'   
set nat source rule 110 source address '192.168.255.0/24'   
set nat source rule 110 translation address masquerade   
VPN-clients will request configuration parameters, optionally you can DNS parameter to the client.

set vpn l2tp remote-access name-server '198.51.100.8'   
set vpn l2tp remote-access name-server '198.51.100.4'   
Established sessions can be viewed using the show l2tp-server sessions operational command vyos@vyos:~$ show l2tp-server sessions  
 ifname | username | ip | ip6 | ip6-dp | calling-sid | rate-limit | state | uptime | rx-bytes | tx-bytes  
--------+----------+---------------+-----+--------+-------------+------------+--------+----------+----------+---------- l2tp0 | test | 192.168.255.3 | | | 192.168.0.36 | | active | 02:01:47 | 7.7 KiB | 1.2 KiB   
LNS (L2TP Network Server)  
LNS are often used to connect to a LAC (L2TP Access Concentrator).

Below is an example to configure a LNS:   
set vpn l2tp remote-access outside-address 192.0.2.2   
set vpn l2tp remote-access client-ip-pool start 192.168.255.2   
set vpn l2tp remote-access client-ip-pool stop 192.168.255.254   
set vpn l2tp remote-access lns shared-secret 'secret'   
set vpn l2tp remote-access ccp-disable   
set vpn l2tp remote-access authentication mode local   
set vpn l2tp remote-access authentication local-users username test password 'test'

The example above uses 192.0.2.2 as external IP address. A LAC normally requires an   
authentication password, which is set in the example configuration to lns shared-secret 'secret'.

This setup requires the Compression Control Protocol (CCP) being disabled, the command set vpn l2tp remote-access ccp-disable accomplishes that.

Bandwidth Shaping  
Bandwidth rate limits can be set for local users or via RADIUS based attributes.

Bandwidth Shaping for local users  
The rate-limit is set in kbit/sec.

set vpn l2tp remote-access outside-address 192.0.2.2   
set vpn l2tp remote-access client-ip-pool start 192.168.255.2   
set vpn l2tp remote-access client-ip-pool stop 192.168.255.254   
set vpn l2tp remote-access authentication mode local   
set vpn l2tp remote-access authentication local-users username test password test   
set vpn l2tp remote-access authentication local-users username test rate-limit download 20480 set vpn l2tp remote-access authentication local-users username test rate-limit upload 10240

vyos@vyos:~$ show l2tp-server sessions  
 ifname | username | ip | ip6 | ip6-dp | calling-sid | rate-limit | state | uptime | rx-bytes | tx-bytes  
--------+----------+---------------+-----+--------+-------------+------------+--------+----------+----------+---------- l2tp0 | test | 192.168.255.3 | | | 192.168.0.36 | | active | 02:01:47 | 7.7 KiB | 1.2 KiB   
RADIUS authentication  
To enable RADIUS based authentication, the authentication mode needs to be changed within the configuration. Previous settings like the local users, still exists within the configuration, however they are not used if the mode has been changed from local to radius. Once changed back to local, it will use all local accounts again.

set vpn l2tp remote-access authentication mode <local|radius>   
Since the RADIUS server would be a single point of failure, multiple RADIUS servers can be setup and will be used subsequentially.

set vpn l2tp remote-access authentication radius server 10.0.0.1 key 'foo'   
set vpn l2tp remote-access authentication radius server 10.0.0.2 key 'foo'   
Note   
Some RADIUS severs use an access control list which allows or denies queries, make sure to add your VyOS router to the allowed client list.

RADIUS source address  
If you are using OSPF as IGP, always the closest interface connected to the RADIUS server is used. With VyOS 1.2 you can bind all outgoing RADIUS requests to a single source IP e.g. the loopback interface.

set vpn l2tp remote-access authentication radius source-address 10.0.0.3   
Above command will use 10.0.0.3 as source IPv4 address for all RADIUS queries on this NAS.

Note   
The source-address must be configured on one of VyOS interface. Best practice would be a loopback or dummy interface.

RADIUS bandwidth shaping attribute  
To enable bandwidth shaping via RADIUS, the option rate-limit needs to be enabled.

set vpn l2tp remote-access authentication radius rate-limit enable   
The default RADIUS attribute for rate limiting is Filter-Id, but you may also redefine it.

set vpn l2tp remote-access authentication radius rate-limit attribute Download-Speed Note   
If you set a custom RADIUS attribute you must define it on both dictionaries at RADIUS server and client, which is the vyos router in our example.

The RADIUS dictionaries in VyOS are located at /usr/share/accel-ppp/radius/   
RADIUS advanced features  
Received RADIUS attributes have a higher priority than parameters defined within the CLI configuration, refer to the explanation below.

Allocation clients ip addresses by RADIUS  
If the RADIUS server sends the attribute Framed-IP-Address then this IP address will be allocated to the client and the option ip-pool within the CLI config is being ignored.

Renaming clients interfaces by RADIUS  
If the RADIUS server uses the attribute NAS-Port-Id, ppp tunnels will be renamed.

Note   
The value of the attribute NAS-Port-Id must be less than 16 characters, otherwise the interface won’t be renamed.

MPLS  
MPLS is a packet forwarding paradigm which differs from regular IP forwarding. Instead of IP addresses being used to make the decision on finding the exit interface, a router will instead use an exact match on a 32 bit/4 byte header called the MPLS label. This label is inserted between the ethernet (layer 2) header and the IP (layer 3) header. One can statically or dynamically assign label allocations, but we will focus on dynamic allocation of labels using some sort of label distribution protocol (such as the aptly named Label Distribution Protocol / LDP, Resource Reservation Protocol / RSVP, or Segment Routing through OSPF/ISIS). These protocols allow for the creation of a unidirectional/unicast path called a labeled switched path (initialized as LSP) throughout the network that operates very much like a tunnel through the network. An easy way of thinking about how an MPLS LSP actually forwards traffic throughout a network is to think of a GRE tunnel. They are not the same in how they operate, but they are the same in how they handle the tunneled packet. It would be good to think of MPLS as a tunneling technology that can be used to transport many different types of packets, to aid in traffic engineering by allowing one to specify paths throughout the network (using RSVP or SR), and to generally allow for easier intra/inter network transport of data packets.

For more information on how MPLS label switching works, please go visit Wikipedia (MPLS).

Note   
MPLS support in VyOS is not finished yet, and therefore its functionality is limited. Currently there is no support for MPLS enabled VPN services such as L2VPNs and mVPNs. RSVP support is also not present as the underlying routing stack (FRR) does not implement it. Currently VyOS implements LDP as described in RFC 5036; other LDP standard are the following ones: RFC 6720, RFC 6667, RFC 5919, RFC 5561, RFC 7552, RFC 4447. Because MPLS is already available (FRR also supports RFC 3031).

Label Distribution Protocol  
The MPLS architecture does not assume a single protocol to create MPLS paths. VyOS supports the Label Distribution Protocol (LDP) as implemented by FRR, based on RFC 5036.

LDP is a TCP based MPLS signaling protocol that distributes labels creating MPLS label switched paths in a dynamic manner. LDP is not a routing protocol, as it relies on other routing protocols for forwarding decisions. LDP cannot bootstrap itself, and therefore relies on said routing protocols for communication with other routers that use LDP.

In order to allow for LDP on the local router to exchange label advertisements with other routers, a TCP session will be established between automatically discovered and statically assigned routers. LDP will try to establish a TCP session to the transport address of other routers. Therefore for LDP to function properly please make sure the transport address is shown in the routing table and reachable to traffic at all times.

It is highly recommended to use the same address for both the LDP router-id and the discovery transport address, but for VyOS MPLS LDP to work both parameters must be explicitly set in the configuration.

Another thing to keep in mind with LDP is that much like BGP, it is a protocol that runs on top of TCP. It however does not have an ability to do something like a refresh capability like BGPs route refresh capability. Therefore one might have to reset the neighbor for a capability change or a configuration change to work.

Configuration Options  
 set protocols mpls interface <interface>   
Use this command to enable MPLS processing on the interface you define.

set protocols mpls ldp interface <interface>   
Use this command to enable LDP on the interface you define.

set protocols mpls ldp router-id <address>   
Use this command to configure the IP address used as the LDP router-id of the local device.

set protocols mpls ldp discovery transport-ipv4-address <address> set protocols mpls ldp discovery transport-ipv6-address <address> Use this command to set the IPv4 or IPv6 transport-address used by LDP.

set protocols mpls ldp neighbor <address> password <password>   
Use this command to configure authentication for LDP peers. Set the IP address of the LDP peer and a password that should be shared in order to become neighbors.

set protocols mpls ldp neighbor <address> session-holdtime <seconds>   
Use this command to configure a specific session hold time for LDP peers. Set the IP address of the LDP peer and a session hold time that should be configured for it. You may have to reset the neighbor for this to work.

set protocols mpls ldp neighbor <address> ttl-security <disable | hop count>   
Use this command to enable, disable, or specify hop count for TTL security for LDP peers. By default the value is set to 255 (or max TTL).

set protocols mpls ldp discovery hello-ipv4-interval <seconds>  
 set protocols mpls ldp discovery hello-ipv4-holdtime <seconds>  
 set protocols mpls ldp discovery hello-ipv6-interval <seconds>  
 set protocols mpls ldp discovery hello-ipv6-holdtime <seconds>   
Use these commands if you would like to set the discovery hello and hold time parameters.

set protocols mpls ldp discovery session-ipv4-holdtime <seconds>  
 set protocols mpls ldp discovery session-ipv6-holdtime <seconds>   
Use this command if you would like to set the TCP session hold time intervals.

set protocols mpls ldp import ipv4 import-filter filter-access-list <access list number>  
 set protocols mpls ldp import ipv6 import-filter filter-access-list6 <access list number> Use these commands to control the importing of forwarding equivalence classes (FECs) for LDP from neighbors. This would be useful for example on only accepting the labeled routes that are needed and not ones that are not needed, such as accepting loopback interfaces and rejecting all others.

set protocols mpls ldp export ipv4 export-filter filter-access-list <access list number>  
 set protocols mpls ldp export ipv6 export-filter filter-access-list6 <access list number> Use these commands to control the exporting of forwarding equivalence classes (FECs) for LDP to neighbors. This would be useful for example on only announcing the labeled routes that are needed and not ones that are not needed, such as announcing loopback interfaces and no others.

set protocols mpls ldp export ipv4 explicit-null  
 set protocols mpls ldp export ipv6 explicit-null   
Use this command if you would like for the router to advertise FECs with a label of 0 for explicit null operations.

set protocols mpls ldp allocation ipv4 access-list <access list number>  
 set protocols mpls ldp allocation ipv6 access-list6 <access list number>   
Use this command if you would like to control the local FEC allocations for LDP. A good example would be for your local router to not allocate a label for everything. Just a label for what it’s useful. A good example would be just a loopback label.

set protocols mpls ldp parameters cisco-interop-tlv   
Use this command to use a Cisco non-compliant format to send and interpret the Dual-Stack capability TLV for IPv6 LDP communications. This is related to RFC 7552.

set protocols mpls ldp parameters ordered-control   
Use this command to use ordered label distribution control mode. FRR by default uses independent label distribution control mode for label distribution. This is related to RFC 5036.

set protocols mpls ldp parameters transport-prefer-ipv4   
Use this command to prefer IPv4 for TCP peer transport connection for LDP when both an IPv4 and IPv6 LDP address are configured on the same interface.

set protocols mpls ldp targeted-neighbor ipv4 enable  
 set protocols mpls ldp targeted-neighbor ipv6 enable   
Use this command to enable targeted LDP sessions to the local router. The router will then respond to any sessions that are trying to connect to it that are not a link local type of TCP connection.

set protocols mpls ldp targeted-neighbor ipv4 address <address>  
 set protocols mpls ldp targeted-neighbor ipv6 address <address>   
Use this command to enable the local router to try and connect with a targeted LDP session to another router.

set protocols mpls ldp targeted-neighbor ipv4 hello-holdtime <seconds> set protocols mpls ldp targeted-neighbor ipv4 hello-interval <seconds>

set protocols mpls ldp targeted-neighbor ipv6 hello-holdtime <seconds>  
 set protocols mpls ldp targeted-neighbor ipv6 hello-interval <seconds>   
Use these commands if you would like to set the discovery hello and hold time parameters for the targeted LDP neighbors.

Sample configuration to setup LDP on VyOS  
set protocols ospf area 0 network '192.168.255.252/32' <--- Routing for loopback set protocols ospf area 0 network '192.168.0.5/32' <--- Routing for an interface connecting to the network   
set protocols ospf parameters router-id '192.168.255.252' <--- Router ID setting for OSPF   
set protocols mpls interface 'eth1' <--- Enable MPLS for an interface connecting to network   
set protocols mpls ldp discovery transport-ipv4-address '192.168.255.252' <--- Transport address for LDP for TCP sessions to connect to   
set protocols mpls ldp interface 'eth1' <--- Enable LDP for an interface connecting to network   
set protocols mpls ldp interface 'lo' <--- Enable LDP on loopback for future services connectivity   
set protocols mpls ldp router-id '192.168.255.252' <--- Router ID setting for LDP set interfaces ethernet eth1 address '192.168.0.5/31' <--- Interface IP for   
connecting to network   
set interfaces loopback lo address '192.168.255.252/32' <--- Interface loopback IP for router ID and other uses   
Operational Mode Commands  
When LDP is working, you will be able to see label information in the outcome of show ip route.

Besides that information, there are also specific show commands for LDP: Show  
 show mpls ldp binding   
Use this command to see the Label Information Base.

show mpls ldp discovery   
Use this command to see discovery hello information  
 show mpls ldp interface   
Use this command to see LDP interface information  
 show mpls ldp neighbor   
Use this command to see LDP neighbor information  
 show mpls ldp neighbor detail   
Use this command to see detailed LDP neighbor information   
Reset  
 reset mpls ldp neighbor <IPv4 or IPv6 address>   
Use this command to reset an LDP neighbor/TCP session that is established

Napalm  
VyOS supports some napalm functions for configuration and op-mode. It requires more tests.

Install napalm-vyos module

apt install python3-pip   
pip3 install napalm   
pip3 install napalm-vyos   
Op-mode  
#!/usr/bin/env python3

import json   
from napalm import get\_network\_driver

driver = get\_network\_driver('vyos')

vyos\_router = driver(  
 hostname="192.0.2.1",  
 username="vyos",  
 password="vyospass",  
 optional\_args={"port": 22},   
)

vyos\_router.open()   
output = vyos\_router.get\_facts()   
print(json.dumps(output, indent=4))

output = vyos\_router.get\_arp\_table()   
print(json.dumps(output, indent=4))

vyos\_router.close()   
Output op-mode   
$ ./vyos-napalm.py   
{  
 "uptime": 7185,  
 "vendor": "VyOS",  
 "os\_version": "1.3.0-rc5",  
 "serial\_number": "",  
 "model": "Standard PC (Q35 + ICH9, 2009)",  
 "hostname": "r4-1.3",  
 "fqdn": "vyos.local",  
 "interface\_list": [  
 "eth0",  
 "eth1",  
 "eth2",  
 "lo",  
 "vtun10"  
 ]   
}

[  
 {  
 "interface": "eth1",  
 "mac": "52:54:00:b2:38:2c",  
 "ip": "192.0.2.2",  
 "age": 0.0  
 },  
 {  
 "interface": "eth0",  
 "mac": "52:54:00:a2:b9:5b",  
 "ip": "203.0.113.11",  
 "age": 0.0  
 }   
]   
Configuration  
We need 2 files, commands.conf and script itself.

Content of commands.conf   
set service ssh disable-host-validation   
set service ssh port '2222'   
set system name-server '192.0.2.8'   
set system name-server '203.0.113.8'   
set interfaces ethernet eth1 description 'FOO'   
Script vyos-napalm.py   
#!/usr/bin/env python3

from napalm import get\_network\_driver

driver = get\_network\_driver('vyos')

vyos\_router = driver(  
 hostname="192.0.2.1",  
 username="vyos",  
 password="vyospass",  
 optional\_args={"port": 22},   
)

vyos\_router.open()   
vyos\_router.load\_merge\_candidate(filename='commands.conf') diffs = vyos\_router.compare\_config()

if bool(diffs) == True:  
 print(diffs)  
 vyos\_router.commit\_config()   
else:

print('No configuration changes to commit')  
 vyos\_router.discard\_config()

vyos\_router.close()   
Output   
$./vyos-napalm.py   
[edit interfaces ethernet eth1]   
+description FOO   
[edit service ssh]   
+disable-host-validation   
+port 2222   
[edit system]   
+name-server 192.0.2.8   
+name-server 203.0.113.8   
[edit]

Netmiko  
VyOS supports configuration via netmiko. It requires to install python3-netmiko module.

Example  
#!/usr/bin/env python3

from netmiko import ConnectHandler

vyos\_router = {  
 "device\_type": "vyos",  
 "host": "192.0.2.1",  
 "username": "vyos",  
 "password": "vyospass",  
 "port": 22,  
 }

net\_connect = ConnectHandler(\*\*vyos\_router)

config\_commands = [  
 'set interfaces ethernet eth0 description WAN', 'set interfaces ethernet eth1 description LAN', ]

# set congiguration   
output = net\_connect.send\_config\_set(config\_commands, exit\_config\_mode=False) print(output)

# commit configuration   
output = net\_connect.commit()

print(output)

# op-mode commands   
output = net\_connect.send\_command("run show interfaces")   
print(output)   
Output   
$ ./vyos-netmiko.py   
configure   
set interfaces ethernet eth0 description WAN   
[edit]   
vyos@r4-1.3# set interfaces ethernet eth1 description LAN   
[edit]   
vyos@r4-1.3#   
commit   
[edit]   
vyos@r4-1.3#   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 203.0.113.1/24 u/u WAN   
eth1 192.0.2.1/30 u/u LAN   
eth2 - u/u   
lo 127.0.0.1/8 u/u  
 ::1/128   
vtun10 10.10.0.1/24 u/u   
[edit]   
OpenConnect  
OpenConnect-compatible server feature is available from this release. Openconnect VPN supports SSL connection and offers full network access. SSL VPN network extension connects the end-user system to the corporate network with access controls based only on network layer information, such as destination IP address and port number. So, it provides safe   
communication for all types of device traffic across public networks and private networks, also encrypts the traffic with SSL protocol.

The remote user will use the openconnect client to connect to the router and will receive an IP address from a VPN pool, allowing full access to the network.

Configuration  
SSL Certificates  
We need to generate the certificate which authenticates users who attempt to access the network resource through the SSL VPN tunnels. The following commands will create a self signed certificates and will be stored in configuration:   
run generate pki ca install <CA name>   
run generate pki certificate sign <CA name> install <Server name>

We can also create the certificates using Cerbort which is an easy-to-use client that fetches a certificate from Let’s Encrypt an open certificate authority launched by the EFF, Mozilla, and others and deploys it to a web server.

sudo certbot certonly --standalone --preferred-challenges http -d <domain name>   
Server Configuration  
set vpn openconnect authentication local-users username <user> password <pass>   
set vpn openconnect authentication mode <local password|radius>   
set vpn openconnect network-settings client-ip-settings subnet <subnet>   
set vpn openconnect network-settings name-server <address>   
set vpn openconnect network-settings name-server <address>   
set vpn openconnect ssl ca-certificate <pki-ca-name>   
set vpn openconnect ssl certificate <pki-cert-name>   
set vpn openconnect ssl passphrase <pki-password>   
2FA OTP support  
Instead of password only authentication, 2FA password authentication + OTP key can be used.

Alternatively, OTP authentication only, without a password, can be used. To do this, an OTP configuration must be added to the configuration above:   
set vpn openconnect authentication mode local <password-otp|otp>   
set vpn openconnect authentication local-users username <user> otp <key>   
set vpn openconnect authentication local-users username <user> interval <interval (optional)> set vpn openconnect authentication local-users username <user> otp-length <otp-length (optional)>   
set vpn openconnect authentication local-users username <user> token-type <token-type (optional)>   
For generating an OTP key in VyOS, you can use the CLI command (operational mode): generate openconnect username <user> otp-key hotp-time   
Verification  
vyos@vyos:~$ sh openconnect-server sessions   
interface username ip remote IP RX TX state uptime  
----------- ---------- ------------- ----------- ------- --------- --------- --------  
sslvpn0 tst 172.20.20.198 192.168.6.1 0 bytes 152 bytes connected 3s   
Note   
It is compatible with Cisco (R) AnyConnect (R) clients.

Example  
SSL Certificates generation  
Follow the instructions to generate CA cert (in configuration mode): vyos@vyos# run generate pki ca install ca-ocserv   
Enter private key type: [rsa, dsa, ec] (Default: rsa)   
Enter private key bits: (Default: 2048)   
Enter country code: (Default: GB) US   
Enter state: (Default: Some-State) Delaware   
Enter locality: (Default: Some-City) Mycity   
Enter organization name: (Default: VyOS) MyORG   
Enter common name: (Default: vyos.io) oc-ca

Enter how many days certificate will be valid: (Default: 1825) 3650   
Note: If you plan to use the generated key on this router, do not encrypt the private key.

Do you want to encrypt the private key with a passphrase? [y/N] N   
2 value(s) installed. Use "compare" to see the pending changes, and "commit" to apply.

[edit]   
Follow the instructions to generate server cert (in configuration mode):   
vyos@vyos# run generate pki certificate sign ca-ocserv install srv-ocserv   
Do you already have a certificate request? [y/N] N   
Enter private key type: [rsa, dsa, ec] (Default: rsa)   
Enter private key bits: (Default: 2048)   
Enter country code: (Default: GB) US   
Enter state: (Default: Some-State) Delaware   
Enter locality: (Default: Some-City) Mycity   
Enter organization name: (Default: VyOS) MyORG   
Enter common name: (Default: vyos.io) oc-srv   
Do you want to configure Subject Alternative Names? [y/N] N   
Enter how many days certificate will be valid: (Default: 365) 1830   
Enter certificate type: (client, server) (Default: server)   
Note: If you plan to use the generated key on this router, do not encrypt the private key.

Do you want to encrypt the private key with a passphrase? [y/N] N   
2 value(s) installed. Use "compare" to see the pending changes, and "commit" to apply.

[edit]   
Each of the install command should be applied to the configuration and commited before using under the openconnect configuration:   
vyos@vyos# commit   
[edit]   
vyos@vyos# save   
Saving configuration to '/config/config.boot'...

Done   
[edit]   
Openconnect Configuration  
Simple setup with one user added and password authentication:   
set vpn openconnect authentication local-users username tst password 'OC\_bad\_Secret' set vpn openconnect authentication mode local password   
set vpn openconnect network-settings client-ip-settings subnet '172.20.20.0/24' set vpn openconnect network-settings name-server '10.1.1.1'   
set vpn openconnect network-settings name-server '10.1.1.2'   
set vpn openconnect ssl ca-certificate 'ca-ocserv'   
set vpn openconnect ssl certificate 'srv-ocserv'   
Adding a 2FA with an OTP-key  
First the OTP keys must be generated and sent to the user and to the configuration: vyos@vyos:~$ generate openconnect username tst otp-key hotp-time   
# You can share it with the user, he just needs to scan the QR in his OTP app   
# username: tst

# OTP KEY: 5PA4SGYTQSGOBO3H3EQSSNCUNZAYAPH2

# OTP URL: otpauth://totp/tst@vyos?

secret=5PA4SGYTQSGOBO3H3EQSSNCUNZAYAPH2&digits=6&period=30██████████████████████████████████████████████████████████████████████████████████████▄▄▄▄▄█▀██▄▀▄█▄▀▀▄▄▄▄██▄▄▄▄▄████  
███████▀█▄▄▀▀▀▄█▄▄▀▄███████  
█████▄▄▄██▀█▀▄▄▀▄▀█▀▀▄███▄▄▄█████  
████▄▄▄▄▄▄▄█▄█▄▀▀▄█▀▀▀█▄█▄▄▄▄▄▄▄████████▄▄▄▀▄▄▄███▀▄▀█▄██▀▀▄▀▄█▀▀████  
████▀▀▀▄█▄▀▀▄▄█▀▄█▄▀▀▄███████  
████▄█▄▀▀▄█▀▀█▄█▄▄▄▄▄▀█▀▀█▀▄▀█▀████  
█████▀█▀▄▄█▀▄▄▄█▄▀█▀▀█▀▄█████  
████▀██▀█▄▄▀▀▀▀█▄▀▀█▄▄▀▀▀▀▀█▄██▀▀████████▄▄▄▀▄██▀█▄▀▄██▄▄▀▀▄█▄██▄█████  
████▀▀▄▀▄▀█▀█▀██▀█▄▄▀█▀█▄██▄▄█▀████  
█████▀█▄▄█▄▀▄▄▀▀▀█▄█▀█████▀▀████  
████▄██▄██▄█▀▄▀▄▄▀▄▄▀█▄▄▄▄▀█▄████  
████▄▄▄▄▄█▄▀█▄█▄▀▄▄█▄█▄▀▄█████  
███████▀▄██▄▄▀█▄▀▄██▄▀▄▀██▀████  
█████▄▄▄████▀▄▄▀▄▄▀█▀▀█▄▀█▀██████  
████▄▄▄▄▄▄▄█▄███▄███▄█▄▄▄▄█▄▄█▄██▄█▄███████████████████████████████████████████████████████████████████████████████████████

# To add this OTP key to configuration, run the following commands:

set vpn openconnect authentication local-users username tst otp key

'ebc1c91b13848ce0bb67d9212934546e41803cfa'

Next it is necessary to configure 2FA for OpenConnect:

set vpn openconnect authentication mode local password-otp

set vpn openconnect authentication local-users username tst otp key

'ebc1c91b13848ce0bb67d9212934546e41803cfa'

Now when connecting the user will first be asked for the password and then the OTP key.

Warning

When using Time-based one-time password (TOTP) (OTP HOTP-time), be sure that the time on

the server and the OTP token generator are synchronized by NTP

To display the configured OTP user settings, use the command:

show openconnect-server user <username> otp <full|key-b32|key-hex|qrcode|uri> Identity Based Configuration

OpenConnect supports a subset of it’s configuration options to be applied on a per user/group

basis, for configuration purposes we refer to this functionality as “Identity based config”. The

following OpenConnect Server Manual outlines the set of configuration options that are

allowed. This can be leveraged to apply different sets of configs to different users or groups of

users.

sudo mkdir -p /config/auth/ocserv/config-per-user

sudo touch /config/auth/ocserv/default-user.conf

set vpn set vpn openconnect authentication identity-based-config mode user   
set vpn openconnect authentication identity-based-config directory /config/auth/ocserv/config-per-user   
set vpn openconnect authentication identity-based-config default-config   
/config/auth/ocserv/default-user.conf   
Warning   
The above directory and default-config must be a child directory of /config/auth, since files outside this directory are not persisted after an image upgrade.

Once you commit the above changes you can create a config file in the   
/config/auth/ocserv/config-per-user directory that matches a username of a user you have created e.g. “tst”. Now when logging in with the “tst” user the config options you set in this file will be loaded.

Be sure to set a sane default config in the default config file, this will be loaded in the case that a user is authenticated and no file is found in the configured directory matching the users username/group.

sudo nano /config/auth/ocserv/config-per-user/tst   
The same configuration options apply when Identity based config is configured in group mode except that group mode can only be used with RADIUS authentication.

Warning   
OpenConnect server matches the filename in a case sensitive manner, make sure the username/group name you configure matches the filename exactly.

Configuring RADIUS accounting  
OpenConnect can be configured to send accounting information to a RADIUS server to capture user session data such as time of connect/disconnect, data transferred, and so on.

Configure an accounting server and enable accounting with:   
set vpn openconnect accounting mode radius   
set vpn openconnect accounting radius server 172.20.20.10   
set vpn openconnect accounting radius server 172.20.20.10 port 1813   
set vpn openconnect accounting radius server 172.20.20.10 key your\_radius\_secret Warning   
The RADIUS accounting feature must be used with the OpenConnect authentication mode RADIUS. It cannot be used with local authentication. You must configure the OpenConnect authentication mode to “radius”.

An example of the data captured by a FREERADIUS server with sql accounting: mysql> SELECT username, nasipaddress, acctstarttime, acctstoptime, acctinputoctets, acctoutputoctets, callingstationid, framedipaddress, connectinfo\_start FROM radacct; +----------+---------------+---------------------+---------------------+-----------------+------------------+-------------------+-----------------+-----------------------------------+   
| username | nasipaddress | acctstarttime | acctstoptime | acctinputoctets | acctoutputoctets | callingstationid | framedipaddress | connectinfo\_start | +----------+---------------+---------------------+---------------------+-----------------+------------------+-------------------+-----------------+-----------------------------------+

| test | 198.51.100.15 | 2023-01-13 00:59:15 | 2023-01-13 00:59:21 | 10606 | 152 | 192.168.6.1 | 172.20.20.198 | Open AnyConnect VPN Agent v8.05-1 | +----------+---------------+---------------------+---------------------+-----------------+------------------+-------------------+-----------------+-----------------------------------+

OSPF  
OSPF is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS). It is defined as OSPF Version 2 in RFC 2328 (1998) for IPv4. Updates for IPv6 are specified as OSPF Version 3 in RFC 5340 (2008). OSPF supports the CIDR addressing model.

OSPF is a widely used IGP in large enterprise networks.

OSPFv2 (IPv4)  
Configuration  
General  
VyOS does not have a special command to start the OSPF process. The OSPF process starts when the first ospf enabled interface is configured.

set protocols ospf area <number> network <A.B.C.D/M>   
This command specifies the OSPF enabled interface(s). If the interface has an address from defined range then the command enables OSPF on this interface so router can provide network information to the other ospf routers via this interface.

This command is also used to enable the OSPF process. The area number can be specified in decimal notation in the range from 0 to 4294967295. Or it can be specified in dotted decimal notation similar to ip address.

Prefix length in interface must be equal or bigger (i.e. smaller network) than prefix length in network statement. For example statement above doesn’t enable ospf on interface with address 192.168.1.1/23, but it does on interface with address 192.168.1.129/25.

In some cases it may be more convenient to enable OSPF on a per interface/subnet basis set protocols ospf interface <interface> area <x.x.x.x | x>  
 set protocols ospf auto-cost reference-bandwidth <number>   
This command sets the reference bandwidth for cost calculations, where bandwidth can be in range from 1 to 4294967, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

set protocols ospf parameters router-id <rid>   
This command sets the router-ID of the OSPF process. The router-ID may be an IP address of the router, but need not be – it can be any arbitrary 32bit number. However it MUST be unique within the entire OSPF domain to the OSPF speaker – bad things will happen if multiple OSPF speakers are configured with the same router-ID!

Optional  
 set protocols ospf default-information originate [always] [metric <number>] [metric-type <1| 2>] [route-map <name>]

Originate an AS-External (type-5) LSA describing a default route into all external-routing capable areas, of the specified metric and metric type. If the always keyword is given then the default is always advertised, even when there is no default present in the routing table. The argument route-map specifies to advertise the default route if the route map is satisfied.

set protocols ospf distance global <distance>   
This command change distance value of OSPF globally. The distance range is 1 to 255.

set protocols ospf distance ospf <external|inter-area|intra-area> <distance>   
This command change distance value of OSPF. The arguments are the distance values for external routes, inter-area routes and intra-area routes respectively. The distance range is 1 to 255.

Note   
Routes with a distance of 255 are effectively disabled and not installed into the kernel.

set protocols ospf log-adjacency-changes [detail]   
This command allows to log changes in adjacency. With the optional detail argument, all changes in adjacency status are shown. Without detail, only changes to full or regressions are shown.

set protocols ospf max-metric router-lsa <administrative|on-shutdown <seconds>|on-startup <seconds>>   
This enables RFC 3137 support, where the OSPF process describes its transit links in its router-LSA as having infinite distance so that other routers will avoid calculating transit paths through the router while still being able to reach networks through the router.

This support may be enabled administratively (and indefinitely) with the administrative command. It may also be enabled conditionally. Conditional enabling of max-metric router-lsas can be for a period of seconds after startup with the on-startup <seconds> command and/or for a period of seconds prior to shutdown with the on-shutdown <seconds> command. The time range is 5 to 86400.

set protocols ospf parameters abr-type <cisco|ibm|shortcut|standard>   
This command selects ABR model. OSPF router supports four ABR models:   
cisco – a router will be considered as ABR if it has several configured links to the networks in different areas one of which is a backbone area. Moreover, the link to the backbone area should be active (working). ibm – identical to “cisco” model but in this case a backbone area link may not be active. standard – router has several active links to different areas. shortcut – identical to “standard” but in this model a router is allowed to use a connected areas topology without involving a backbone area for inter-area connections.

Detailed information about “cisco” and “ibm” models differences can be found in RFC 3509. A “shortcut” model allows ABR to create routes between areas based on the topology of the areas connected to this router but not using a backbone area in case if non-backbone route will be cheaper. For more information about “shortcut” model, see ospf-shortcut-abr-02.txt  
 set protocols ospf parameters rfc1583-compatibility   
RFC 2328, the successor to RFC 1583, suggests according to section G.2 (changes) in section 16.4.1 a change to the path preference algorithm that prevents possible routing loops that were possible in the old version of OSPFv2. More specifically it demands that inter-area paths and intra-area backbone path are now of equal preference but still both preferred to external paths. This command should NOT be set normally.

set protocols ospf interface <interface> passive [disable]   
This command specifies interface as passive. Passive interface advertises its address, but does not run the OSPF protocol (adjacencies are not formed and hello packets are not generated).

The optional disable option allows to exclude interface from passive state. This command is used if the command passive-interface default was configured.

set protocols ospf passive-interface default   
This command specifies all interfaces as passive by default. Because this command changes the configuration logic to a default passive; therefore, interfaces where router adjacencies are expected need to be configured with the passive-interface-exclude command.

set protocols ospf maximum-paths <1-64>   
Use this command to control the maximum number of equal cost paths to reach a specific destination. The upper limit may differ if you change the value of MULTIPATH\_NUM during compilation. The default is MULTIPATH\_NUM (64).

set protocols ospf refresh timers <seconds>   
The router automatically updates link-state information with its neighbors. Only an obsolete information is updated which age has exceeded a specific threshold. This parameter changes a threshold value, which by default is 1800 seconds (half an hour). The value is applied to the whole OSPF router. The timer range is 10 to 1800.

set protocols ospf timers throttle spf <delay|initial-holdtime|max-holdtime> <seconds> This command sets the initial delay, the initial-holdtime and the maximum-holdtime between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds. delay sets the initial SPF schedule delay in milliseconds. The default value is 200 ms. initial-holdtime sets the minimum hold time between two consecutive SPF calculations. The default value is 1000 ms. max-holdtime sets the maximum wait time between two consecutive SPF calculations. The default value is 10000 ms.

set protocols ospf ldp-sync   
This command will enable IGP-LDP synchronization globally for OSPF. This requires for LDP to be functional. This is described in RFC 5443. By default all interfaces operational in OSPF are enabled for synchronization. Loopbacks are exempt.

set protocols ospf ldp-sync holddown <seconds>   
This command will change the hold down value globally for IGP-LDP synchronization during convergence/interface flap events.

set protocols ospf capability opaque   
ospfd supports Opaque LSA RFC 2370 as partial support for MPLS Traffic Engineering LSAs. The opaque-lsa capability must be enabled in the configuration.

An alternate command could be “mpls-te on” (Traffic Engineering)   
Note   
FRR offers only partial support for some of the routing protocol extensions that are used with MPLS-TE; it does not support a complete RSVP-TE solution.

Area Configuration  
 set protocols ospf area <number> area-type stub   
This command specifies the area to be a Stub Area. That is, an area where no router originates routes external to OSPF and hence an area where all external routes are via the ABR(s). Hence,

ABRs for such an area do not need to pass AS-External LSAs (type-5) or ASBR-Summary LSAs (type-4) into the area. They need only pass Network-Summary (type-3) LSAs into such an area, along with a default-route summary.

set protocols ospf area <number> area-type stub no-summary   
This command specifies the area to be a Totally Stub Area. In addition to stub area limitations this area type prevents an ABR from injecting Network-Summary (type-3) LSAs into the specified stub area. Only default summary route is allowed.

set protocols ospf area <number> area-type stub default-cost <number>   
This command sets the cost of default-summary LSAs announced to stubby areas. The cost range is 0 to 16777215.

set protocols ospf area <number> area-type nssa   
This command specifies the area to be a Not So Stubby Area. External routing information is imported into an NSSA in Type-7 LSAs. Type-7 LSAs are similar to Type-5 AS-external LSAs, except that they can only be flooded into the NSSA. In order to further propagate the NSSA external information, the Type-7 LSA must be translated to a Type-5 AS-external-LSA by the NSSA ABR.

set protocols ospf area <number> area-type nssa no-summary   
This command specifies the area to be a NSSA Totally Stub Area. ABRs for such an area do not need to pass Network-Summary (type-3) LSAs (except the default summary route), ASBR-Summary LSAs (type-4) and AS-External LSAs (type-5) into the area. But Type-7 LSAs that convert to Type-5 at the NSSA ABR are allowed.

set protocols ospf area <number> area-type nssa default-cost <number>   
This command sets the default cost of LSAs announced to NSSA areas. The cost range is 0 to 16777215.

set protocols ospf area <number> area-type nssa translate <always|candidate|never>   
Specifies whether this NSSA border router will unconditionally translate Type-7 LSAs into Type-5 LSAs. When role is Always, Type-7 LSAs are translated into Type-5 LSAs regardless of the translator state of other NSSA border routers. When role is Candidate, this router participates in the translator election to determine if it will perform the translations duties. When role is Never, this router will never translate Type-7 LSAs into Type-5 LSAs.

set protocols ospf area <number> authentication plaintext-password   
This command specifies that simple password authentication should be used for the given area. The password must also be configured on a per-interface basis.

set protocols ospf area <number> authentication md5   
This command specify that OSPF packets must be authenticated with MD5 HMACs within the given area. Keying material must also be configured on a per-interface basis.

set protocols ospf area <number> range <A.B.C.D/M> [cost <number>]   
This command summarizes intra area paths from specified area into one summary-LSA (Type-3) announced to other areas. This command can be used only in ABR and ONLY router-LSAs (Type-1) and network-LSAs (Type-2) (i.e. LSAs with scope area) can be summarized. AS-external-LSAs (Type-5) can’t be summarized - their scope is AS. The optional argument cost specifies the aggregated link metric. The metric range is 0 to 16777215.

set protocols ospf area <number> range <A.B.C.D/M> not-advertise

This command instead of summarizing intra area paths filter them - i.e. intra area paths from this range are not advertised into other areas. This command makes sense in ABR only.

set protocols ospf area <number> export-list <acl\_number>   
Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area. This command makes sense in ABR only.

set protocols ospf area <number> import-list <acl\_number>   
Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs. This command makes sense in ABR only.

set protocols ospf area <number> range <A.B.C.D/M> substitute <E.F.G.H/M>   
One Type-3 summary-LSA with routing info <E.F.G.H/M> is announced into backbone area if defined area contains at least one intra-area network (i.e. described with router-LSA or network-LSA) from range <A.B.C.D/M>. This command makes sense in ABR only.

set protocols ospf area <number> shortcut <default|disable|enable>   
This parameter allows to “shortcut” routes (non-backbone) for inter-area routes. There are three modes available for routes shortcutting:   
default – this area will be used for shortcutting only if ABR does not have a link to the backbone area or this link was lost. enable – the area will be used for shortcutting every time the route that goes through it is cheaper. disable – this area is never used by ABR for routes shortcutting.

set protocols ospf area <number> virtual-link <A.B.C.D>   
Provides a backbone area coherence by virtual link establishment.

In general, OSPF protocol requires a backbone area (area 0) to be coherent and fully connected. I.e. any backbone area router must have a route to any other backbone area router. Moreover, every ABR must have a link to backbone area. However, it is not always possible to have a physical link to a backbone area. In this case between two ABR (one of them has a link to the backbone area) in the area (not stub area) a virtual link is organized.

<number> – area identifier through which a virtual link goes. <A.B.C.D> – ABR router-id with which a virtual link is established. Virtual link must be configured on both routers.

Formally, a virtual link looks like a point-to-point network connecting two ABR from one area one of which physically connected to a backbone area. This pseudo-network is considered to belong to a backbone area.

Interface Configuration  
 set protocols ospf interface <interface> area <x.x.x.x | x> Enable ospf on an interface and set associated area.

If you have a lot of interfaces, and/or a lot of subnets, then enabling OSPF via this command may result in a slight performance improvement.

set protocols ospf interface <interface> authentication plaintext-password <text> This command sets OSPF authentication key to a simple password. After setting, all OSPF packets are authenticated. Key has length up to 8 chars.

Simple text password authentication is insecure and deprecated in favour of MD5 HMAC authentication.

set protocols ospf interface <interface> authentication md5 key-id <id> md5-key <text> This command specifys that MD5 HMAC authentication must be used on this interface. It sets OSPF authentication key to a cryptographic password. Key-id identifies secret key used to create the message digest. This ID is part of the protocol and must be consistent across routers on a

link. The key can be long up to 16 chars (larger strings will be truncated), and is associated with the given key-id.

set protocols ospf interface <interface> bandwidth <number>   
This command sets the interface bandwidth for cost calculations, where bandwidth can be in range from 1 to 100000, specified in Mbits/s.

set protocols ospf interface <interface> cost <number>   
This command sets link cost for the specified interface. The cost value is set to router-LSA’s metric field and used for SPF calculation. The cost range is 1 to 65535.

set protocols ospf interface <interface> dead-interval <number>   
Set number of seconds for router Dead Interval timer value used for Wait Timer and Inactivity Timer. This value must be the same for all routers attached to a common network. The default value is 40 seconds. The interval range is 1 to 65535.

set protocols ospf interface <interface> hello-multiplier <number>   
The hello-multiplier specifies how many Hellos to send per second, from 1 (every second) to 10 (every 100ms). Thus one can have 1s convergence time for OSPF. If this form is specified, then the hello-interval advertised in Hello packets is set to 0 and the hello-interval on received Hello packets is not checked, thus the hello-multiplier need NOT be the same across multiple routers on a common link.

set protocols ospf interface <interface> hello-interval <number>   
Set number of seconds for Hello Interval timer value. Setting this value, Hello packet will be sent every timer value seconds on the specified interface. This value must be the same for all routers attached to a common network. The default value is 10 seconds. The interval range is 1 to 65535.

set protocols ospf interface <interface> bfd   
This command enables BFD on this OSPF link interface.

set protocols ospf interface <interface> mtu-ignore   
This command disables check of the MTU value in the OSPF DBD packets. Thus, use of this command allows the OSPF adjacency to reach the FULL state even though there is an interface MTU mismatch between two OSPF routers.

set protocols ospf interface <interface> network <type>   
This command allows to specify the distribution type for the network connected to this interface:   
broadcast – broadcast IP addresses distribution. non-broadcast – address distribution in NBMA networks topology. point-to-multipoint – address distribution in point-to-multipoint networks. point-to-point – address distribution in point-to-point networks.

set protocols ospf interface <interface> priority <number>   
This command sets Router Priority integer value. The router with the highest priority will be more eligible to become Designated Router. Setting the value to 0, makes the router ineligible to become Designated Router. The default value is 1. The interval range is 0 to 255.

set protocols ospf interface <interface> retransmit-interval <number>   
This command sets number of seconds for RxmtInterval timer value. This value is used when retransmitting Database Description and Link State Request packets if acknowledge was not received. The default value is 5 seconds. The interval range is 3 to 65535.

set protocols ospf interface <interface> transmit-delay <number>   
This command sets number of seconds for InfTransDelay value. It allows to set and adjust for each interface the delay interval before starting the synchronizing process of the router’s database with all neighbors. The default value is 1 seconds. The interval range is 3 to 65535.

set protocols ospf interface <interface> ldp-sync disable This command disables IGP-LDP sync for this specific interface.

set protocols ospf interface <interface> ldp-sync holddown <seconds>   
This command will change the hold down value for IGP-LDP synchronization during convergence/interface flap events, but for this interface only.

External Route Summarisation  
This feature summarises originated external LSAs (Type-5 and Type-7). Summary Route will be originated on-behalf of all matched external LSAs.

set protocols ospf aggregation timer <seconds>   
Configure aggregation delay timer interval.

Summarisation starts only after this delay timer expiry.

set protocols ospf summary-address x.x.x.x/y [tag (1-4294967295)]   
This command enable/disables summarisation for the configured address range.

Tag is the optional parameter. If tag configured Summary route will be originated with the configured tag.

set protocols ospf summary-address x.x.x.x/y no-advertise   
This command to ensure not advertise the summary lsa for the matched external LSAs.

Graceful Restart  
 set protocols ospf graceful-restart [grace-period (1-1800)]   
Configure Graceful Restart RFC 3623 restarting support. When enabled, the default grace period is 120 seconds.

To perform a graceful shutdown, the FRR graceful-restart prepare ip ospf EXEC-level command needs to be issued before restarting the ospfd daemon.

set protocols ospf graceful-restart helper enable [router-id A.B.C.D]   
Configure Graceful Restart RFC 3623 helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours.

To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

set protocols ospf graceful-restart helper no-strict-lsa-checking   
By default strict-lsa-checking is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router.

This command disables it.

set protocols ospf graceful-restart helper supported-grace-time Supports as HELPER for configured grace period.

set protocols ospf graceful-restart helper planned-only It helps to support as HELPER only for planned restarts.

By default, it supports both planned and unplanned outages.

Manual Neighbor Configuration  
OSPF routing devices normally discover their neighbors dynamically by listening to the broadcast or multicast hello packets on the network. Because an NBMA network does not

support broadcast (or multicast), the device cannot discover its neighbors dynamically, so you must configure all the neighbors statically.

set protocols ospf neighbor <A.B.C.D>   
This command specifies the IP address of the neighboring device.

set protocols ospf neighbor <A.B.C.D> poll-interval <seconds>   
This command specifies the length of time, in seconds, before the routing device sends hello packets out of the interface before it establishes adjacency with a neighbor. The range is 1 to 65535 seconds. The default value is 60 seconds.

set protocols ospf neighbor <A.B.C.D> priority <number>   
This command specifies the router priority value of the nonbroadcast neighbor associated with the IP address specified. The default is 0. This keyword does not apply to point-to-multipoint interfaces.

Redistribution Configuration  
 set protocols ospf redistribute <route source>   
This command redistributes routing information from the given route source to the OSPF process. There are five modes available for route source: bgp, connected, kernel, rip, static.

set protocols ospf default-metric <number>   
This command specifies the default metric value of redistributed routes. The metric range is 0 to 16777214.

set protocols ospf redistribute <route source> metric <number>   
This command specifies metric for redistributed routes from the given route source. There are five modes available for route source: bgp, connected, kernel, rip, static. The metric range is 1 to 16777214.

set protocols ospf redistribute <route source> metric-type <1|2>   
This command specifies metric type for redistributed routes. Difference between two metric types that metric type 1 is a metric which is “commensurable” with inner OSPF links. When calculating a metric to the external destination, the full path metric is calculated as a metric sum path of a router which had advertised this link plus the link metric. Thus, a route with the least summary metric will be selected. If external link is advertised with metric type 2 the path is selected which lies through the router which advertised this link with the least metric despite of the fact that internal path to this router is longer (with more cost). However, if two routers advertised an external link and with metric type 2 the preference is given to the path which lies through the router with a shorter internal path. If two different routers advertised two links to the same external destimation but with different metric type, metric type 1 is preferred. If type of a metric left undefined the router will consider these external links to have a default metric type 2.

set protocols ospf redistribute <route source> route-map <name>   
This command allows to use route map to filter redistributed routes from the given route source. There are five modes available for route source: bgp, connected, kernel, rip, static.

Operational Mode Commands  
 show ip ospf neighbor   
This command displays the neighbors status.

Neighbor ID Pri State Dead Time Address Interface RXmtL RqstL DBsmL 10.0.13.1 1 Full/DR 38.365s 10.0.13.1 eth0:10.0.13.3 0 0 0

10.0.23.2 1 Full/Backup 39.175s 10.0.23.2 eth1:10.0.23.3 0 0 0 show ip ospf neighbor detail   
This command displays the neighbors information in a detailed form, not just a summary table.

Neighbor 10.0.13.1, interface address 10.0.13.1 In the area 0.0.0.0 via interface eth0  
 Neighbor priority is 1, State is Full, 5 state changes Most recent state change statistics:  
 Progressive change 11m55s ago  
 DR is 10.0.13.1, BDR is 10.0.13.3  
 Options 2 \*|-|-|-|-|-|E|-  
 Dead timer due in 34.854s  
 Database Summary List 0  
 Link State Request List 0  
 Link State Retransmission List 0  
 Thread Inactivity Timer on  
 Thread Database Description Retransmision off Thread Link State Request Retransmission on  
 Thread Link State Update Retransmission on

Neighbor 10.0.23.2, interface address 10.0.23.2  
 In the area 0.0.0.1 via interface eth1  
 Neighbor priority is 1, State is Full, 4 state changes  
 Most recent state change statistics:  
 Progressive change 41.193s ago  
 DR is 10.0.23.3, BDR is 10.0.23.2  
 Options 2 \*|-|-|-|-|-|E|-  
 Dead timer due in 35.661s  
 Database Summary List 0  
 Link State Request List 0  
 Link State Retransmission List 0  
 Thread Inactivity Timer on  
 Thread Database Description Retransmision off  
 Thread Link State Request Retransmission on  
 Thread Link State Update Retransmission on  
 show ip ospf neighbor <A.B.C.D>   
This command displays the neighbors information in a detailed form for a neighbor whose IP address is specified.

show ip ospf neighbor <interface>   
This command displays the neighbors status for a neighbor on the specified interface.

show ip ospf interface [<interface>]   
This command displays state and configuration of OSPF the specified interface, or all interfaces if no interface is given.

eth0 is up  
 ifindex 2, MTU 1500 bytes, BW 4294967295 Mbit <UP,BROADCAST,RUNNING,MULTICAST>

Internet Address 10.0.13.3/24, Broadcast 10.0.13.255, Area 0.0.0.0  
 MTU mismatch detection: enabled  
 Router ID 10.0.23.3, Network Type BROADCAST, Cost: 1  
 Transmit Delay is 1 sec, State Backup, Priority 1  
 Backup Designated Router (ID) 10.0.23.3, Interface Address 10.0.13.3  
 Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters  
 Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5  
 Hello due in 4.470s  
 Neighbor Count is 1, Adjacent neighbor count is 1   
eth1 is up  
 ifindex 3, MTU 1500 bytes, BW 4294967295 Mbit <UP,BROADCAST,RUNNING,MULTICAST> Internet Address 10.0.23.3/24, Broadcast 10.0.23.255, Area 0.0.0.1  
 MTU mismatch detection: enabled  
 Router ID 10.0.23.3, Network Type BROADCAST, Cost: 1  
 Transmit Delay is 1 sec, State DR, Priority 1  
 Backup Designated Router (ID) 10.0.23.2, Interface Address 10.0.23.2  
 Saved Network-LSA sequence number 0x80000002  
 Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters  
 Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5  
 Hello due in 4.563s  
 Neighbor Count is 1, Adjacent neighbor count is 1  
 show ip ospf route [detail]   
This command displays the OSPF routing table, as determined by the most recent SPF   
calculation. With the optional detail argument, each route item’s advertiser router and network attribute will be shown.

============ OSPF network routing table ============ N IA 10.0.12.0/24 [3] area: 0.0.0.0  
 via 10.0.13.3, eth0   
N 10.0.13.0/24 [1] area: 0.0.0.0  
 directly attached to eth0   
N IA 10.0.23.0/24 [2] area: 0.0.0.0  
 via 10.0.13.3, eth0   
N 10.0.34.0/24 [2] area: 0.0.0.0  
 via 10.0.13.3, eth0

============ OSPF router routing table ============= R 10.0.23.3 [1] area: 0.0.0.0, ABR  
 via 10.0.13.3, eth0   
R 10.0.34.4 [2] area: 0.0.0.0, ASBR  
 via 10.0.13.3, eth0

============ OSPF external routing table =========== N E2 172.16.0.0/24 [2/20] tag: 0  
 via 10.0.13.3, eth0

The table consists of following data:   
OSPF network routing table – includes a list of acquired routes for all accessible networks (or aggregated area ranges) of OSPF system. “IA” flag means that route destination is in the area to which the router is not connected, i.e. it’s an inter-area path. In square brackets a summary metric for all links through which a path lies to this network is specified. “via” prefix defines a router-gateway, i.e. the first router on the way to the destination (next hop). OSPF router routing table – includes a list of acquired routes to all accessible ABRs and ASBRs. OSPF external routing table – includes a list of acquired routes that are external to the OSPF process. “E” flag points to the external link metric type (E1 – metric type 1, E2 – metric type 2). External link metric is printed in the “<metric of the router which advertised the link>/<link metric>” format.

show ip ospf border-routers   
This command displays a table of paths to area boundary and autonomous system boundary routers.

show ip ospf database   
This command displays a summary table with a database contents (LSA).

OSPF Router with ID (10.0.13.1)

Router Link States (Area 0.0.0.0)

Link ID ADV Router Age Seq# CkSum Link count 10.0.13.1 10.0.13.1 984 0x80000005 0xd915 1 10.0.23.3 10.0.23.3 1186 0x80000008 0xfe62 2 10.0.34.4 10.0.34.4 1063 0x80000004 0x4e3f 1

Net Link States (Area 0.0.0.0)

Link ID ADV Router Age Seq# CkSum 10.0.13.1 10.0.13.1 994 0x80000003 0x30bb 10.0.34.4 10.0.34.4 1188 0x80000001 0x9411

Summary Link States (Area 0.0.0.0)

Link ID ADV Router Age Seq# CkSum Route   
10.0.12.0 10.0.23.3 1608 0x80000001 0x6ab6 10.0.12.0/24 10.0.23.0 10.0.23.3 981 0x80000003 0xe232 10.0.23.0/24

AS External Link States

Link ID ADV Router Age Seq# CkSum Route   
172.16.0.0 10.0.34.4 1063 0x80000001 0xc40d E2 172.16.0.0/24 [0x0] show ip ospf database <type> [A.B.C.D] [adv-router <A.B.C.D>|self-originate] This command displays a database contents for a specific link advertisement type.

The type can be the following: asbr-summary, external, network, nssa-external, opaque-area, opaque-as, opaque-link, router, summary.

[A.B.C.D] – link-state-id. With this specified the command displays portion of the network environment that is being described by the advertisement. The value entered depends on the advertisement’s LS type. It must be entered in the form of an IP address.

adv-router <A.B.C.D> – router id, which link advertisements need to be reviewed. self-originate displays only self-originated LSAs from the local router.

OSPF Router with ID (10.0.13.1)

Router Link States (Area 0.0.0.0)

LS age: 1213   
Options: 0x2 : \*|-|-|-|-|-|E|-  
LS Flags: 0x3   
Flags: 0x0   
LS Type: router-LSA   
Link State ID: 10.0.13.1   
Advertising Router: 10.0.13.1   
LS Seq Number: 80000009   
Checksum: 0xd119   
Length: 36

Number of Links: 1

Link connected to: a Transit Network  
 (Link ID) Designated Router address: 10.0.13.1  
 (Link Data) Router Interface address: 10.0.13.1  
 Number of TOS metrics: 0  
 TOS 0 Metric: 1  
 show ip ospf database max-age   
This command displays LSAs in MaxAge list.

Examples  
Enable OSPF  
Node 1   
set interfaces loopback lo address 10.1.1.1/32   
set interfaces ethernet eth0 address 192.168.0.1/24 set protocols ospf area 0 network 192.168.0.0/24 set protocols ospf area 0 network 10.1.1.1/32   
set protocols ospf parameters router-id 10.1.1.1 Node 2   
set interfaces loopback lo address 10.1.1.2/32   
set interfaces ethernet eth0 address 192.168.0.2/24 set protocols ospf area 0 network 192.168.0.0/24 set protocols ospf area 0 network 10.1.1.2/32   
set protocols ospf parameters router-id 10.1.1.2

Here’s the neighbors up:   
Node-1@vyos:~$ show ip ospf neighbor

Neighbor ID Pri State Up Time Dead Time Address Interface RXmtL RqstL DBsmL   
10.1.1.2 1 Full/DR 3m43s 36.094s 192.168.0.2 eth0:192.168.0.1 0 0 0

Node-2@vyos:~$ show ip ospf neighbor

Neighbor ID Pri State Up Time Dead Time Address Interface RXmtL RqstL DBsmL   
10.1.1.1 1 Full/Backup 3m47s 31.736s 192.168.0.1 eth0:192.168.0.2   
0 0 0   
Here’s the routes:   
Node-1@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup  
 t - trapped, o - offload failure

O 10.1.1.1/32 [110/0] is directly connected, lo, weight 1, 00:00:14 O>\* 10.1.1.2/32 [110/1] via 192.168.0.2, eth0, weight 1, 00:00:07   
O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:32

Node-2@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

O>\* 10.1.1.1/32 [110/1] via 192.168.0.1, eth0, weight 1, 00:00:11   
O 10.1.1.2/32 [110/0] is directly connected, lo, weight 1, 00:00:04   
O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:18 Enable OSPF with route redistribution of the loopback and default originate:Node 1   
set interfaces loopback lo address 10.1.1.1/32   
set protocols ospf area 0 network 192.168.0.0/24

set protocols ospf default-information originate always   
set protocols ospf default-information originate metric 10 set protocols ospf default-information originate metric-type 2 set protocols ospf log-adjacency-changes   
set protocols ospf parameters router-id 10.1.1.1   
set protocols ospf redistribute connected metric-type 2   
set protocols ospf redistribute connected route-map CONNECT

set policy route-map CONNECT rule 10 action permit   
set policy route-map CONNECT rule 10 match interface lo Node 2   
set interfaces loopback lo address 10.2.2.2/32   
set protocols ospf area 0 network 192.168.0.0/24   
set protocols ospf log-adjacency-changes   
set protocols ospf parameters router-id 10.2.2.2   
set protocols ospf redistribute connected metric-type 2   
set protocols ospf redistribute connected route-map CONNECT

set policy route-map CONNECT rule 10 action permit set policy route-map CONNECT rule 10 match interface lo Enable OSPF and IGP-LDP synchronization:  
Node 1:   
set interfaces loopback lo address 10.1.1.1/32   
set interfaces ethernet eth0 address 192.168.0.1/24

set protocols ospf area 0 network '192.168.0.0/24' set protocols ospf area 0 network '10.1.1.1/32' set protocols ospf parameters router-id '10.1.1.1' set protocols ospf ldp-sync

set protocols mpls interface eth0   
set protocols mpls ldp discovery transport-ipv4-address 10.1.1.1   
set protocols mpls ldp interface lo   
set protocols mpls ldp interface eth0   
set protocols mpls ldp parameters transport-prefer-ipv4   
set protocols mpls ldp router-id 10.1.1.1   
This gives us IGP-LDP synchronization for all non-loopback interfaces with a holddown timer of zero seconds:   
Node-1@vyos:~$ show ip ospf mpls ldp-sync  
 eth0  
 LDP-IGP Synchronization enabled: yes  
 Holddown timer in seconds: 0  
 State: Sync achieved   
Enable OSPF with Segment Routing (Experimental):

Node 1   
set interfaces loopback lo address 10.1.1.1/32   
set interfaces ethernet eth0 address 192.168.0.1/24

set protocols ospf area 0 network '192.168.0.0/24'   
set protocols ospf area 0 network '10.1.1.1/32'   
set protocols ospf parameters opaque-lsa   
set protocols ospf parameters router-id '10.1.1.1'   
set protocols ospf segment-routing global-block high-label-value '1100' set protocols ospf segment-routing global-block low-label-value '1000' set protocols ospf segment-routing prefix 10.1.1.1/32 index explicit-null set protocols ospf segment-routing prefix 10.1.1.1/32 index value '1' Node 2   
set interfaces loopback lo address 10.1.1.2/32   
set interfaces ethernet eth0 address 192.168.0.2/24

set protocols ospf area 0 network '192.168.0.0/24'   
set protocols ospf area 0 network '10.1.1.2/32'   
set protocols ospf parameters opaque-lsa   
set protocols ospf parameters router-id '10.1.1.2'   
set protocols ospf segment-routing global-block high-label-value '1100'   
set protocols ospf segment-routing global-block low-label-value '1000'   
set protocols ospf segment-routing prefix 10.1.1.2/32 index explicit-null   
set protocols ospf segment-routing prefix 10.1.1.2/32 index value '2'   
This gives us MPLS segment routing enabled and labels for far end loopbacks:   
Node-1@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 -----------------------------------------------------------  
 1002 SR (OSPF) 192.168.0.2 IPv4 Explicit Null <-- Node-2 loopback learned on Node-1 15000 SR (OSPF) 192.168.0.2 implicit-null  
 15001 SR (OSPF) 192.168.0.2 implicit-null

Node-2@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 -----------------------------------------------------------  
 1001 SR (OSPF) 192.168.0.1 IPv4 Explicit Null <-- Node-1 loopback learned on Node-2 15000 SR (OSPF) 192.168.0.1 implicit-null  
 15001 SR (OSPF) 192.168.0.1 implicit-null   
Here is the routing tables showing the MPLS segment routing label operations:   
Node-1@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,

> - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

O 10.1.1.1/32 [110/0] is directly connected, lo, weight 1, 00:03:43   
O>\* 10.1.1.2/32 [110/1] via 192.168.0.2, eth0, label IPv4 Explicit Null, weight 1, 00:03:32 O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:43

Node-2@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

O>\* 10.1.1.1/32 [110/1] via 192.168.0.1, eth0, label IPv4 Explicit Null, weight 1, 00:03:36 O 10.1.1.2/32 [110/0] is directly connected, lo, weight 1, 00:03:51   
O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:51   
OSPFv3 (IPv6)  
Configuration  
General  
VyOS does not have a special command to start the OSPFv3 process. The OSPFv3 process starts when the first ospf enabled interface is configured.

set protocols ospfv3 interface <interface> area <number>   
This command specifies the OSPFv3 enabled interface. This command is also used to enable the OSPF process. The area number can be specified in decimal notation in the range from 0 to 4294967295. Or it can be specified in dotted decimal notation similar to ip address.

set protocols ospfv3 parameters router-id <rid>   
This command sets the router-ID of the OSPFv3 process. The router-ID may be an IP address of the router, but need not be – it can be any arbitrary 32bit number. However it MUST be unique within the entire OSPFv3 domain to the OSPFv3 speaker – bad things will happen if multiple OSPFv3 speakers are configured with the same router-ID!

Optional  
 set protocols ospfv3 distance global <distance>   
This command change distance value of OSPFv3 globally. The distance range is 1 to 255.

set protocols ospfv3 distance ospfv3 <external|inter-area|intra-area> <distance>   
This command change distance value of OSPFv3. The arguments are the distance values for external routes, inter-area routes and intra-area routes respectively. The distance range is 1 to 255.

Area Configuration  
 set protocols ospfv3 area <number> range <prefix>   
This command summarizes intra area paths from specified area into one Type-3 Inter-Area Prefix LSA announced to other areas. This command can be used only in ABR.

set protocols ospfv3 area <number> range <prefix> not-advertise

This command instead of summarizing intra area paths filter them - i.e. intra area paths from this range are not advertised into other areas. This command makes sense in ABR only.

Interface Configuration  
 set protocols ospfv3 interface <interface> ipv6 cost <number>   
This command sets link cost for the specified interface. The cost value is set to router-LSA’s metric field and used for SPF calculation. The cost range is 1 to 65535.

set protocols ospfv3 interface <interface> dead-interval <number>   
Set number of seconds for router Dead Interval timer value used for Wait Timer and Inactivity Timer. This value must be the same for all routers attached to a common network. The default value is 40 seconds. The interval range is 1 to 65535.

set protocols ospfv3 interface <interface> hello-interval <number>   
Set number of seconds for Hello Interval timer value. Setting this value, Hello packet will be sent every timer value seconds on the specified interface. This value must be the same for all routers attached to a common network. The default value is 10 seconds. The interval range is 1 to 65535.

set protocols ospfv3 interface <interface> mtu-ignore   
This command disables check of the MTU value in the OSPF DBD packets. Thus, use of this command allows the OSPF adjacency to reach the FULL state even though there is an interface MTU mismatch between two OSPF routers.

set protocols ospfv3 interface <interface> network <type>   
This command allows to specify the distribution type for the network connected to this interface:   
broadcast – broadcast IP addresses distribution. point-to-point – address distribution in point-to-point networks.

set protocols ospfv3 interface <interface> priority <number>   
This command sets Router Priority integer value. The router with the highest priority will be more eligible to become Designated Router. Setting the value to 0, makes the router ineligible to become Designated Router. The default value is 1. The interval range is 0 to 255.

set protocols ospfv3 interface <interface> passive   
This command specifies interface as passive. Passive interface advertises its address, but does not run the OSPF protocol (adjacencies are not formed and hello packets are not generated).

set protocols ospfv3 interface <interface> retransmit-interval <number>   
This command sets number of seconds for RxmtInterval timer value. This value is used when retransmitting Database Description and Link State Request packets if acknowledge was not received. The default value is 5 seconds. The interval range is 3 to 65535.

set protocols ospfv3 interface <interface> transmit-delay <number>   
This command sets number of seconds for InfTransDelay value. It allows to set and adjust for each interface the delay interval before starting the synchronizing process of the router’s database with all neighbors. The default value is 1 seconds. The interval range is 3 to 65535.

Graceful Restart  
 set protocols ospfv3 graceful-restart [grace-period (1-1800)]   
Configure Graceful Restart RFC 3623 restarting support. When enabled, the default grace period is 120 seconds.

To perform a graceful shutdown, the FRR graceful-restart prepare ip ospf EXEC-level command needs to be issued before restarting the ospfd daemon.

set protocols ospfv3 graceful-restart helper enable [router-id A.B.C.D]   
Configure Graceful Restart RFC 3623 helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours.

To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

set protocols ospfv3 graceful-restart helper lsa-check-disable   
By default strict-lsa-checking is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router.

This command disables it.

set protocols ospfv3 graceful-restart helper supported-grace-time Supports as HELPER for configured grace period.

set protocols ospfv3 graceful-restart helper planned-only It helps to support as HELPER only for planned restarts.

By default, it supports both planned and unplanned outages.

Redistribution Configuration  
 set protocols ospfv3 redistribute <route source>   
This command redistributes routing information from the given route source to the OSPFv3 process. There are five modes available for route source: bgp, connected, kernel, ripng, static.

set protocols ospf redistribute <route source> route-map <name>   
This command allows to use route map to filter redistributed routes from given route source. There are five modes available for route source: bgp, connected, kernel, ripng, static.

Operational Mode Commands  
 show ipv6 ospfv3 neighbor   
This command displays the neighbors status.

show ipv6 ospfv3 neighbor detail   
This command displays the neighbors information in a detailed form, not just a summary table.

show ipv6 ospfv3 neighbor drchoice   
This command displays the neighbor DR choice information.

show ipv6 ospfv3 interface [prefix]|[<interface> [prefix]]   
This command displays state and configuration of OSPF the specified interface, or all interfaces if no interface is given. Whith the argument prefix this command shows connected prefixes to advertise.

show ipv6 ospfv3 route   
This command displays the OSPF routing table, as determined by the most recent SPF calculation.

show ipv6 ospfv3 border-routers   
This command displays a table of paths to area boundary and autonomous system boundary routers.

show ipv6 ospfv3 database   
This command displays a summary table with a database contents (LSA).

show ipv6 ospfv3 database <type> [A.B.C.D] [adv-router <A.B.C.D>|self-originate] This command displays a database contents for a specific link advertisement type.

show ipv6 ospfv3 redistribute   
This command displays external information redistributed into OSPFv3 Configuration Example  
A typical configuration using 2 nodes.

Node 1:   
set protocols ospfv3 interface eth1 area 0.0.0.0   
set protocols ospfv3 area 0.0.0.0 range 2001:db8:1::/64   
set protocols ospfv3 parameters router-id 192.168.1.1   
set protocols ospfv3 redistribute connected   
Node 2:   
set protocols ospfv3 interface eth1 area 0.0.0.0   
set protocols ospfv3 area 0.0.0.0 range 2001:db8:2::/64   
set protocols ospfv3 parameters router-id 192.168.2.1   
set protocols ospfv3 redistribute connected   
To see the redistributed routes:   
show ipv6 ospfv3 redistribute   
Note   
You cannot easily redistribute IPv6 routes via OSPFv3 on a WireGuard interface link. This requires you to configure link-local addresses manually on the WireGuard interfaces, see T1483.

Example configuration for WireGuard interfaces:   
Node 1   
set interfaces wireguard wg01 address 'fe80::216:3eff:fe51:fd8c/64' set interfaces wireguard wg01 address '192.168.0.1/24'   
set interfaces wireguard wg01 peer ospf02 allowed-ips '::/0'   
set interfaces wireguard wg01 peer ospf02 allowed-ips '0.0.0.0/0' set interfaces wireguard wg01 peer ospf02 endpoint '10.1.1.101:12345' set interfaces wireguard wg01 peer ospf02 pubkey 'ie3...='   
set interfaces wireguard wg01 port '12345'   
set protocols ospfv3 parameters router-id 192.168.1.1   
set protocols ospfv3 interface 'wg01' area 0.0.0.0   
set protocols ospfv3 interface 'lo' area 0.0.0.0   
Node 2   
set interfaces wireguard wg01 address 'fe80::216:3eff:fe0a:7ada/64' set interfaces wireguard wg01 address '192.168.0.2/24'   
set interfaces wireguard wg01 peer ospf01 allowed-ips '::/0'   
set interfaces wireguard wg01 peer ospf01 allowed-ips '0.0.0.0/0' set interfaces wireguard wg01 peer ospf01 endpoint '10.1.1.100:12345' set interfaces wireguard wg01 peer ospf01 pubkey 'NHI...='   
set interfaces wireguard wg01 port '12345'   
set protocols ospfv3 parameters router-id 192.168.1.2   
set protocols ospfv3 interface 'wg01' area 0.0.0.0   
set protocols ospfv3 interface 'lo' area 0.0.0.0   
Status   
vyos@ospf01:~$ sh ipv6 ospfv3 neighbor

Neighbor ID Pri DeadTime State/IfState Duration I/F[State]   
192.168.0.2 1 00:00:37 Full/PointToPoint 00:18:03 wg01[PointToPoint]

vyos@ospf02# run sh ipv6 ospfv3 neighbor   
Neighbor ID Pri DeadTime State/IfState Duration I/F[State]   
192.168.0.1 1 00:00:39 Full/PointToPoint 00:19:44 wg01[PointToPoint]   
PIM – Protocol Independent Multicast  
VyOS supports PIM-SM as well as IGMP v2 and v3   
PIM must be configured in every interface of every participating router. Every router must also have the location of the Rendevouz Point manually configured. Then, unidirectional shared trees rooted at the Rendevouz Point will automatically be built for multicast distribution.

Traffic from multicast sources will go to the Rendezvous Point, and receivers will pull it from a shared tree using IGMP.

Multicast receivers will talk IGMP to their local router, so, besides having PIM configured in every router, IGMP must also be configured in any router where there could be a multicast receiver locally connected.

VyOS supports both IGMP version 2 and version 3 (which allows source-specific multicast).

PIM-SM - PIM Sparse Mode  
 set protocols pim ecmp   
If PIM has the a choice of ECMP nexthops for a particular RPF, PIM will cause S,G flows to be spread out amongst the nexthops. If this command is not specified then the first nexthop found will be used.

set protocols pim ecmp rebalance   
If PIM is using ECMP and an interface goes down, cause PIM to rebalance all S,G flows across the remaining nexthops. If this command is not configured PIM only modifies those S,G flows that were using the interface that went down.

set protocols pim join-prune-interval <n>   
Modify the join/prune interval that PIM uses to the new value. Time is specified in seconds. The default time is 60 seconds.

If you enter a value smaller than 60 seconds be aware that this can and will affect convergence at scale.

set protocols pim keep-alive-timer <n>   
Modify the time out value for a S,G flow from 1-65535 seconds. If choosing a value below 31 seconds be aware that some hardware platforms cannot see data flowing in better than 30 second chunks.

set protocols pim packets <n>   
When processing packets from a neighbor process the number of packets incoming at one time before moving on to the next task.

The default value is 3 packets.

This command is only useful at scale when you can possibly have a large number of PIM control packets flowing.

set protocols pim register-accept-list <prefix-list>

When PIM receives a register packet the source of the packet will be compared to the prefix-list specified, and if a permit is received normal processing continues. If a deny is returned for the source address of the register packet a register stop message is sent to the source.

set protocols pim register-suppress-time <n>   
Modify the time that pim will register suppress a FHR will send register notifications to the kernel.

set protocols pim rp <address> group <group>   
In order to use PIM, it is necessary to configure a RP for join messages to be sent to. Currently the only methodology to do this is via static rendezvous point commands.

All routers in the PIM network must agree on these values.

The first ip address is the RP’s address and the second value is the matching prefix of group ranges covered.

set protocols pim rp keep-alive-timer <n>   
Modify the time out value for a S,G flow from 1-65535 seconds at RP. The normal keepalive period for the KAT(S,G) defaults to 210 seconds. However, at the RP, the keepalive period must be at least the Register\_Suppression\_Time, or the RP may time out the (S,G) state before the next Null-Register arrives. Thus, the KAT(S,G) is set to max(Keepalive\_Period,   
RP\_Keepalive\_Period) when a Register-Stop is sent.

If choosing a value below 31 seconds be aware that some hardware platforms cannot see data flowing in better than 30 second chunks.

See RFC 7761#section-4.1 for details.

set protocols pim no-v6-secondary   
When sending PIM hello packets tell PIM to not send any v6 secondary addresses on the interface. This information is used to allow PIM to use v6 nexthops in it’s decision for RPF lookup if this option is not set (default).

set protocols pim spt-switchover infinity-and-beyond [prefix-list <list>]   
On the last hop router if it is desired to not switch over to the SPT tree configure this command.

Optional parameter prefix-list can be use to control which groups to switch or not switch. If a group is PERMIT as per the prefix-list, then the SPT switchover does not happen for it and if it is DENY, then the SPT switchover happens.

set protocols pim ssm prefix-list <list>   
Specify a range of group addresses via a prefix-list that forces PIM to never do SSM over.

Interface specific commands  
 set protocols pim interface <interface> bfd [profile <name>]   
Automatically create BFD session for each RIP peer discovered in this interface. When the BFD session monitor signalize that the link is down the RIP peer is removed and all the learned routes associated with that peer are removed.

If optional profile parameter is used, select a BFD profile for the BFD sessions created via this interface.

set protocols pim interface <interface> dr-priority <n>   
Set the DR Priority for the interface. This command is useful to allow the user to influence what node becomes the DR for a LAN segment.

set protocols pim interface <interface> hello <n> Set the PIM hello and hold interval for a interface.

set protocols pim interface <interface> no-bsm   
Tell PIM that we would not like to use this interface to process bootstrap messages.

set protocols pim interface <interface> no-unicast-bsm   
Tell PIM that we would not like to use this interface to process unicast bootstrap messages.

set protocols pim interface <interface> passive   
Disable sending and receiving PIM control packets on the interface.

set protocols pim interface <interface> source-address <ip-address>   
If you have multiple addresses configured on a particular interface and would like PIM to use a specific source address associated with that interface.

IGMP - Internet Group Management Protocol)  
 set protocols pim igmp watermark-warning <n>   
Configure watermark warning generation for an IGMP group limit. Generates warning once the configured group limit is reached while adding new groups.

Interface specific commands  
 set protocols pim interface <interface> igmp join <multicast-address> source-address <IP-address>   
Use this command to allow the selected interface to join a multicast group defining the multicast address you want to join and the source IP address too.

set protocols pim interface <interface> igmp query-interval <seconds>   
Use this command to configure in the selected interface the IGMP host query interval (1-1800) in seconds that PIM will use.

set protocols pim interface <interface> igmp query-max-response-time <n>   
Use this command to configure in the selected interface the IGMP query response timeout value (10-250) in deciseconds. If a report is not returned in the specified time, it will be assumed the (S,G) or (\*,G) state RFC 7761#section-4.1 has timed out.

set protocols pim interface <interface> igmp version <version-number>   
Use this command to define in the selected interface whether you choose IGMP version 2 or 3. The default value is 3.

Example  
In the following example we can see a basic multicast setup:

Router 1   
set interfaces ethernet eth2 address '172.16.0.2/24'   
set interfaces ethernet eth1 address '100.64.0.1/24'   
set protocols ospf area 0 network '172.16.0.0/24'   
set protocols ospf area 0 network '100.64.0.0/24'   
set protocols igmp interface eth1   
set protocols pim interface eth1   
set protocols pim interface eth2   
set protocols pim rp address 172.16.255.1 group '224.0.0.0/4' Router 3   
set interfaces dummy dum0 address '172.16.255.1/24'   
set interfaces ethernet eth0 address '172.16.0.1/24'   
set interfaces ethernet eth1 address '172.16.1.1/24'

set protocols ospf area 0 network '172.16.0.0/24'   
set protocols ospf area 0 network '172.16.255.0/24'   
set protocols ospf area 0 network '172.16.1.0/24'   
set protocols pim interface dum0   
set protocols pim interface eth0   
set protocols pim interface eth1   
set protocols pim rp address 172.16.255.1 group '224.0.0.0/4' Router 2   
set interfaces ethernet eth1 address '10.0.0.1/24'   
set interfaces ethernet eth2 address '172.16.1.2/24'   
set protocols ospf area 0 network '10.0.0.0/24'   
set protocols ospf area 0 network '172.16.1.0/24'   
set protocols pim interface eth1   
set protocols pim interface eth2   
set protocols pim rp address 172.16.255.1 group '224.0.0.0/4'

PIM6 - Protocol Independent Multicast for IPv6  
VyOS facilitates IPv6 Multicast by supporting PIMv6 and MLD.

PIMv6 (Protocol Independent Multicast for IPv6) must be configured in every interface of every participating router. Every router must also have the location of the Rendevouz Point manually configured. Then, unidirectional shared trees rooted at the Rendevouz Point will automatically be built for multicast distribution.

Traffic from multicast sources will go to the Rendezvous Point, and receivers will pull it from a shared tree using MLD (Multicast Listener Discovery).

Multicast receivers will talk MLD to their local router, so, besides having PIMv6 configured in every router, MLD must also be configured in any router where there could be a multicast receiver locally connected.

VyOS supports both MLD version 1 and version 2 (which allows source-specific multicast).

Basic commands  
These are the commands for a basic setup.

set protocols pim6 interface <interface-name>   
Use this command to enable PIMv6 in the selected interface so that it can communicate with PIMv6 neighbors. This command also enables MLD reports and query on the interface unless mld disable is configured.

set protocols pim6 interface <interface-name> mld disable Disable MLD reports and query on the interface.

Tuning commands  
You can also tune multicast with the following commands.

set protocols pim6 interface <interface-name> mld interval <seconds>   
Use this command to configure in the selected interface the MLD host query interval (1-65535) in seconds that PIM will use. The default value is 125 seconds.

set protocols pim6 interface <interface-name> mld join <multicast-address> Use this command to allow the selected interface to join a multicast group.

set protocols pim6 interface <interface-name> mld join <multicast-address> source <source-address>   
Use this command to allow the selected interface to join a source-specific multicast group.

set protocols pim6 interface <interface-name> mld last-member-query-count <count> Set the MLD last member query count. The default value is 2.

set protocols pim6 interface <interface-name> mld last-member-query-interval <milliseconds> Set the MLD last member query interval in milliseconds (100-6553500). The default value is 1000 milliseconds.

set protocols pim6 interface <interface-name> mld max-response-time <milliseconds> Set the MLD query response timeout in milliseconds (100-6553500). The default value is 10000 milliseconds.

set protocols pim6 interface <interface-name> mld version <version-number> Set the MLD version used on this interface. The default value is 2.

Configuration Example  
To enable MLD reports and query on interfaces eth0 and eth1:   
set protocols pim6 interface eth0   
set protocols pim6 interface eth1   
The following configuration explicitly joins multicast group ff15::1234 on interface eth1 and source-specific multicast group ff15::5678 with source address 2001:db8::1 on interface eth1: set protocols pim6 interface eth0 mld join ff15::1234   
set protocols pim6 interface eth1 mld join ff15::5678 source 2001:db8::1

PPTP-Server  
The Point-to-Point Tunneling Protocol (PPTP) has been implemented in VyOS only for backwards compatibility. PPTP has many well known security issues and you should use one of the many other new VPN implementations.

As per default and if not otherwise defined, mschap-v2 is being used for authentication and mppe 128-bit (stateless) for encryption. If no gateway-address is set within the configuration, the lowest IP out of the /24 client-ip-pool is being used. For instance, in the example below it would be 192.168.0.1.

server example  
set vpn pptp remote-access authentication local-users username test password 'test' set vpn pptp remote-access authentication mode 'local'   
set vpn pptp remote-access client-ip-pool start '192.168.0.10'   
set vpn pptp remote-access client-ip-pool stop '192.168.0.15'   
set vpn pptp remote-access gateway-address '10.100.100.1'   
set vpn pptp remote-access outside-address '10.1.1.120'   
client example (debian 9)  
Install the client software via apt and execute pptpsetup to generate the configuration.

apt-get install pptp-linux   
pptpsetup --create TESTTUNNEL --server 10.1.1.120 --username test --password test --encrypt pon TESTTUNNEL   
The command pon TESTUNNEL establishes the PPTP tunnel to the remote system.

All tunnel sessions can be checked via:

run sh pptp-server sessions  
 ifname | username | calling-sid | ip | type | comp | state | uptime--------+----------+-------------+--------------+------+------+--------+----------  
 ppp0 | test | 10.1.1.99 | 192.168.0.10 | pptp | mppe | active | 00:00:58

Quick Start  
This chapter will guide you on how to get up to speed quickly using your new VyOS system. It will show you a very basic configuration example that will provide a NAT gateway for a device with two network interfaces (eth0 and eth1).

Configuration Mode  
By default, VyOS is in operational mode, and the command prompt displays a $. To configure VyOS, you will need to enter configuration mode, resulting in the command prompt displaying a #, as demonstrated below:   
vyos@vyos$ configure   
vyos@vyos#   
Commit and Save  
After every configuration change, you need to apply the changes by using the following command:   
commit   
Once your configuration works as expected, you can save it permanently by using the following command:   
save   
Interface Configuration  
• Your outside/WAN interface will be eth0. It will receive its interface address via DHCP.• Your internal/LAN interface will be eth1. It will use a static IP address of 192.168.0.1/24.

After switching to Configuration Mode issue the following commands:   
set interfaces ethernet eth0 address dhcp   
set interfaces ethernet eth0 description 'OUTSIDE'   
set interfaces ethernet eth1 address '192.168.0.1/24'   
set interfaces ethernet eth1 description 'INSIDE'   
SSH Management  
After switching to Configuration Mode issue the following commands, and your system will listen on every interface for incoming SSH connections. You might want to check the SSH chapter on how to listen on specific addresses only.

set service ssh port '22'   
DHCP/DNS quick-start  
The following settings will configure DHCP and DNS services on your internal/LAN network, where VyOS will act as the default gateway and DNS server.

• The default gateway and DNS recursor address will be 192.168.0.1/24• The address range 192.168.0.2/24 - 192.168.0.8/24 will be reserved for static assignments  
• DHCP clients will be assigned IP addresses within the range of 192.168.0.9 - 192.168.0.254 and have a domain name of internal-network

• DHCP leases will hold for one day (86400 seconds)  
• VyOS will serve as a full DNS recursor, replacing the need to utilize Google, Cloudflare, or other public DNS servers (which is good for privacy)  
• Only hosts from your internal/LAN network can use the DNS recursor   
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 default-router '192.168.0.1'   
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 name-server '192.168.0.1'   
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 domain-name 'vyos.net'   
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 lease '86400' set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 range 0 start '192.168.0.9'   
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 range 0 stop '192.168.0.254'

set service dns forwarding cache-size '0'   
set service dns forwarding listen-address '192.168.0.1'   
set service dns forwarding allow-from '192.168.0.0/24'   
NAT  
The following settings will configure SNAT rules for our internal/LAN network, allowing hosts to communicate through the outside/WAN network via IP masquerade.

set nat source rule 100 outbound-interface 'eth0'   
set nat source rule 100 source address '192.168.0.0/24'   
set nat source rule 100 translation address masquerade   
Firewall  
A new firewall structure—which uses the nftables backend, rather than iptables—is available on all installations starting from VyOS 1.4-rolling-202308040557. The firewall supports creation of distinct, interlinked chains for each Netfilter hook and allows for more granular control over the packet filtering process.

Note   
Documentation for most of the new firewall CLI can be found in the firewall chapter.The legacy firewall is still available for versions before 1.4-rolling-202308040557 and can be found in the firewall-legacy chapter. The examples in this section use the new configuration.

The firewall begins with the base filter tables you define for each of the forward, input, and output Netfiter hooks. Each of these tables is populated with rules that are processed in order and can jump to other chains for more granular filtering.

Configure Firewall Groups  
To make firewall configuration easier, we can create groups of interfaces, networks, addresses, ports, and domains that describe different parts of our network. We can then use them for filtering within our firewall rulesets, allowing for more concise and readable configuration. In this case, we will create two interface groups—a WAN group for our interfaces connected to the public internet and a LAN group for the interfaces connected to our internal network. Additionally, we will create a network group, NET-INSIDE-v4, that contains our internal subnet.

set firewall group interface-group WAN interface eth0   
set firewall group interface-group LAN interface eth1   
set firewall group network-group NET-INSIDE-v4 network '192.168.0.0/24'   
Configure Stateful Packet Filtering  
With the new firewall structure, we have have a lot of flexibility in how we group and order our rules, as shown by the two alternative approaches below.

Option 1: Common Chain  
We can create a common chain for stateful connection filtering of multiple interfaces (or multiple netfilter hooks on one interface). Those individual chains can then jump to the common chain for stateful connection filtering, returning to the original chain for further rule processing if no action is taken on the packet.

The chain we will create is called CONN\_FILTER and has three rules:  
• A default action of return, which returns the packet back to the original chain if no action is taken.

A rule to accept packets from established and related connections.•  
• A rule to drop packets from invalid connections.

set firewall ipv4 name CONN\_FILTER default-action 'return'

set firewall ipv4 name CONN\_FILTER rule 10 action 'accept'   
set firewall ipv4 name CONN\_FILTER rule 10 state established 'enable' set firewall ipv4 name CONN\_FILTER rule 10 state related 'enable'

set firewall ipv4 name CONN\_FILTER rule 20 action 'drop'   
set firewall ipv4 name CONN\_FILTER rule 20 state invalid 'enable'   
Then, we can jump to the common chain from both the forward and input hooks as the first filtering rule in the respective chains:   
set firewall ipv4 forward filter rule 10 action 'jump'   
set firewall ipv4 forward filter rule 10 jump-target CONN\_FILTER

set firewall ipv4 input filter rule 10 action 'jump'   
set firewall ipv4 input filter rule 10 jump-target CONN\_FILTER   
Option 2: Per-Hook Chain  
Alternatively, instead of configuring the CONN\_FILTER chain described above, you can take the more traditional stateful connection filtering approach by creating rules on each hook’s chain: set firewall ipv4 forward filter rule 5 action 'accept'   
set firewall ipv4 forward filter rule 5 state established 'enable'   
set firewall ipv4 forward filter rule 5 state related 'enable'   
set firewall ipv4 forward filter rule 10 action 'drop'   
set firewall ipv4 forward filter rule 10 state invalid 'enable'

set firewall ipv4 input filter rule 5 action 'accept'   
set firewall ipv4 input filter rule 5 state established 'enable' set firewall ipv4 input filter rule 5 state related 'enable' set firewall ipv4 input filter rule 10 action 'drop'

set firewall ipv4 input filter rule 10 state invalid 'enable'   
Block Incoming Traffic  
Now that we have configured stateful connection filtering to allow traffic from established and related connections, we can block all other incoming traffic addressed to our local network. Create a new chain (OUTSIDE-IN) which will drop all traffic that is not explicity allowed at some point in the chain. Then, we can jump to that chain from the forward hook when traffic is coming from the WAN interface group and is addressed to our local network.

set firewall ipv4 name OUTSIDE-IN default-action 'drop'

set firewall ipv4 forward filter rule 100 action jump   
set firewall ipv4 forward filter rule 100 jump-target OUTSIDE-IN   
set firewall ipv4 forward filter rule 100 inbound-interface interface-group WAN   
set firewall ipv4 forward filter rule 100 destination group network-group NET-INSIDE-v4 We should also block all traffic destinated to the router itself that isn’t explicitly allowed at some point in the chain for the input hook. As we’ve already configured stateful packet filtering above, we only need to set the default action to drop:   
set firewall ipv4 input filter default-action 'drop'   
Allow Management Access  
We can now configure access to the router itself, allowing SSH access from the inside/LAN network and rate limiting SSH access from the outside/WAN network.

First, create a new dedicated chain (VyOS\_MANAGEMENT) for management access, which returns to the parent chain if no action is taken. Add a rule to accept traffic from the LAN interface group:   
set firewall ipv4 name VyOS\_MANAGEMENT default-action 'return'   
Configure a rule on the input hook filter to jump to the VyOS\_MANAGEMENT chain when new connections are addressed to port 22 (SSH) on the router itself:   
set firewall ipv4 input filter rule 20 action jump   
set firewall ipv4 input filter rule 20 jump-target VyOS\_MANAGEMENT   
set firewall ipv4 input filter rule 20 destination port 22   
set firewall ipv4 input filter rule 20 protocol tcp   
Finally, configure the VyOS\_MANAGEMENT chain to accept connection from the LAN interface group while limiting requests coming from the WAN interface group to 4 per minute:   
set firewall ipv4 name VyOS\_MANAGEMENT rule 15 action 'accept'   
set firewall ipv4 name VyOS\_MANAGEMENT rule 15 inbound-interface interface-group 'LAN'

set firewall ipv4 name VyOS\_MANAGEMENT rule 20 action 'drop'   
set firewall ipv4 name VyOS\_MANAGEMENT rule 20 recent count 4   
set firewall ipv4 name VyOS\_MANAGEMENT rule 20 recent time minute   
set firewall ipv4 name VyOS\_MANAGEMENT rule 20 state new enable   
set firewall ipv4 name VyOS\_MANAGEMENT rule 20 inbound-interface interface-group 'WAN'

set firewall ipv4 name VyOS\_MANAGEMENT rule 21 action 'accept'   
set firewall ipv4 name VyOS\_MANAGEMENT rule 21 state new enable   
set firewall ipv4 name VyOS\_MANAGEMENT rule 21 inbound-interface interface-group 'WAN'

Allow Access to Services  
Here we’re allowing the router to respond to pings. Then, we can allow access to the DNS recursor we configured earlier, accepting traffic bound for port 53 from all hosts on the NET-INSIDE-v4 network:   
set firewall ipv4 input filter rule 30 action 'accept'   
set firewall ipv4 input filter rule 30 icmp type-name 'echo-request'   
set firewall ipv4 input filter rule 30 protocol 'icmp'   
set firewall ipv4 input filter rule 30 state new 'enable'

set firewall ipv4 input filter rule 40 action 'accept'   
set firewall ipv4 input filter rule 40 destination port '53'   
set firewall ipv4 input filter rule 40 protocol 'tcp\_udp'   
set firewall ipv4 input filter rule 40 source group network-group NET-INSIDE-v4 Finally, we can now configure access to the services running on this router, allowing all connections coming from localhost:   
set firewall ipv4 input filter rule 50 action 'accept'   
set firewall ipv4 input filter rule 50 source address 127.0.0.0/8   
Commit changes, save the configuration, and exit configuration mode:   
vyos@vyos# commit   
vyos@vyos# save   
Saving configuration to '/config/config.boot'...

Done   
vyos@vyos# exit   
vyos@vyos$   
Hardening  
Especially if you are allowing SSH remote access from the outside/WAN interface, there are a few additional configuration steps that should be taken.

Replace the default vyos system user:   
set system login user myvyosuser authentication plaintext-password mysecurepassword Set up Key Based Authentication:   
set system login user myvyosuser authentication public-keys myusername@mydesktop type ssh-rsa   
set system login user myvyosuser authentication public-keys myusername@mydesktop key contents\_of\_id\_rsa.pub   
Finally, try and SSH into the VyOS install as your new user. Once you have confirmed that your new user can access your router without a password, delete the original vyos user and completely disable password authentication for SSH:   
delete system login user vyos   
set service ssh disable-password-authentication   
As above, commit your changes, save the configuration, and exit configuration mode: vyos@vyos# commit   
vyos@vyos# save   
Saving configuration to '/config/config.boot'...

Done

vyos@vyos# exit   
vyos@vyos$   
You now should have a simple yet secure and functioning router to experiment with further. Enjoy!

Command Line Interface  
The VyOS CLI comprises an operational and a configuration mode.

Operational Mode  
Operational mode allows for commands to perform operational system tasks and view system and service status, while configuration mode allows for the modification of system   
configuration.

The CLI provides a built-in help system. In the CLI the ? key may be used to display available commands. The TAB key can be used to auto-complete commands and will present the help system upon a conflict or unknown value.

For example typing sh followed by the TAB key will complete to show. Pressing TAB a second time will display the possible sub-commands of the show command.

vyos@vyos:~$ s[tab]   
set show   
Example showing possible show commands:   
vyos@vyos:~$ show [tab]   
Possible completions:  
 arp Show Address Resolution Protocol (ARP) information  
 bridge Show bridging information  
 cluster Show clustering information  
 configuration Show running configuration  
 conntrack Show conntrack entries in the conntrack table  
 conntrack-sync  
 Show connection syncing information  
 date Show system date and time  
 dhcp Show Dynamic Host Configuration Protocol (DHCP) information  
 dhcpv6 Show status related to DHCPv6  
 disk Show status of disk device  
 dns Show Domain Name Server (DNS) information  
 file Show files for a particular image  
 firewall Show firewall information  
 flow-accounting  
 Show flow accounting statistics  
 hardware Show system hardware details  
 history show command history  
 host Show host information  
 incoming Show ethernet input-policy information   
: q   
You can scroll up with the keys [Shift]+[PageUp] and scroll down with [Shift]+[PageDown].

When the output of a command results in more lines than can be displayed on the terminal screen the output is paginated as indicated by a : prompt.

When viewing in page mode the following commands are available:  
• q key can be used to cancel output  
• space will scroll down one page  
• b will scroll back one page   
 return will scroll down one line  
•• up-arrow and down-arrow will scroll up or down one line at a time respectively• left-arrow and right-arrow can be used to scroll left or right in the event that the output has lines which exceed the terminal size.

Configuration Mode  
To enter configuration mode use the configure command:   
vyos@vyos:~$ configure   
[edit]   
vyos@vyos:~#   
Note   
Prompt changes from $ to #. To exit configuration mode, type exit.

vyos@vyos:~# exit   
exit   
vyos@vyos:~$   
See the configuration section of this document for more information on configuration mode.

Configuration Overview  
VyOS makes use of a unified configuration file for the entire system’s configuration: /config/config.boot. This allows easy template creation, backup, and replication of system configuration. A system can thus also be easily cloned by simply copying the required configuration files.

Terminology  
A VyOS system has three major types of configurations:  
• Active or running configuration is the system configuration that is loaded and currently active (used by VyOS). Any change in the configuration will have to be committed to belong to the active/running configuration.

• Working configuration is the one that is currently being modified in configuration mode.

Changes made to the working configuration do not go into effect until the changes are committed with the commit command. At which time the working configuration will become the active or running configuration.

• Saved configuration is the one saved to a file using the save command. It allows you to keep safe a configuration for future uses. There can be multiple configuration files. The default or “boot” configuration is saved and loaded from the file /config/config.boot.

Seeing and navigating the configuration  
 show configuration   
View the current active configuration, also known as the running configuration, from the operational mode.

vyos@vyos:~$ show configuration   
interfaces {  
 ethernet eth0 {  
 address dhcp

hw-id 00:53:00:00:aa:01  
 }  
 loopback lo {  
 }   
}   
service {  
 ssh {  
 port 22  
 }   
}   
system {  
 config-management {  
 commit-revisions 20  
 }  
 console {  
 device ttyS0 {  
 speed 9600  
 }  
 }  
 login {  
 user vyos {  
 authentication {  
 encrypted-password \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }  
 level admin  
 }  
 }  
 ntp {  
 server 0.pool.ntp.org {  
 }  
 server 1.pool.ntp.org {  
 }  
 server 2.pool.ntp.org {  
 }  
 }  
 syslog {  
 global {  
 facility all {  
 level notice  
 }  
 facility protocols {  
 level debug  
 }  
 }

}   
}   
By default, the configuration is displayed in a hierarchy like the above example, this is only one of the possible ways to display the configuration. When the configuration is generated and the device is configured, changes are added through a collection of set and delete commands.

show configuration commands   
Get a collection of all the set commands required which led to the running configuration.

vyos@vyos:~$ show configuration commands   
set interfaces ethernet eth0 address 'dhcp'   
set interfaces ethernet eth0 hw-id '00:53:dd:44:3b:0f'   
set interfaces loopback 'lo'   
set service ssh port '22'   
set system config-management commit-revisions '20'   
set system console device ttyS0 speed '9600'   
set system login user vyos authentication encrypted-password '$6$Vt68...QzF0'   
set system login user vyos level 'admin'   
set system ntp server '0.pool.ntp.org'   
set system ntp server '1.pool.ntp.org'   
set system ntp server '2.pool.ntp.org'   
set system syslog global facility all level 'notice'   
set system syslog global facility protocols level 'debug'   
Both these show commands should be executed when in operational mode, they do not work directly in configuration mode. There is a special way on how to Access opmode from config mode.

Hint   
Use the show configuration commands | strip-private command when you want to hide private data. You may want to do so if you want to share your configuration on the forum.

show configuration json   
View the current active configuration in JSON format.

{"interfaces": {"ethernet": {"eth0": {"address": ["192.0.2.11/24", "192.0.2.35/24"], "hw-id": "52:54:00:48:a0:c6"}, "eth1": {"address": ["203.0.113.1/24"], "hw-id": "52:54:00:fc:50:0b"}}, "loopback": {"lo": {}}}, "protocols": {"static": {"route": {"0.0.0.0/0": {"next-hop": {"192.0.2.254": {}}}}}}, "service": {"ssh": {"disable-host-validation": {}}}, "system": {"config-management": {"commit-revisions": "100"}, "console": {"device": {"ttyS0": {"speed": "115200"}}}, "host-name": "r11-vyos", "login": {"user": {"vyos": {"authentication": {"encrypted-password": "$6$Vt68...F0", "plaintext-password": "", "public-keys": {"vyos@vyos": {"key": "AAAAxxx=", "type": "ssh-rsa"}}}}}}, "name-server": ["203.0.113.254"], "ntp": {"server": {"time1.vyos.net": {},   
"time2.vyos.net": {}, "time3.vyos.net": {}}}, "syslog": {"global": {"facility": {"all": {"level": "info"}, "protocols": {"level": "debug"}}}}, "time-zone": "America/New\_York"}}  
 show configuration json pretty   
View the current active configuration in readable JSON format.

{  
 "interfaces": {  
 "ethernet": {

"eth0": {  
 "address": [  
 "192.0.2.11/24",  
 "192.0.2.35/24"  
 ],  
 "hw-id": "52:54:00:48:a0:c6"  
 },  
 "eth1": {  
 "address": [  
 "203.0.113.1/24"  
 ],  
 "hw-id": "52:54:00:fc:50:0b"  
 }  
 },  
 "loopback": {  
 "lo": {}  
 }  
 },  
 "protocols": {  
 "static": {  
 "route": {  
 "0.0.0.0/0": {  
 "next-hop": {  
 "192.0.2.254": {}  
 }  
 }  
 }  
 }  
 },  
 "service": {  
 "ssh": {  
 "disable-host-validation": {}  
 }  
 },  
 "system": {  
 "config-management": {  
 "commit-revisions": "100"  
 },  
 "console": {  
 "device": {  
 "ttyS0": {  
 "speed": "115200"  
 }  
 }

},  
 "host-name": "r11-vyos",  
 "login": {  
 "user": {  
 "vyos": {  
 "authentication": {  
 "encrypted-password": "$6$Vt68...F0", "plaintext-password": "",  
 "public-keys": {  
 "vyos@vyos": {  
 "key": "AAAAxxx=",  
 "type": "ssh-rsa"  
 }  
 }  
 }  
 }  
 }  
 },  
 "name-server": [  
 "203.0.113.254"  
 ],  
 "ntp": {  
 "server": {  
 "time1.vyos.net": {},  
 "time2.vyos.net": {},  
 "time3.vyos.net": {}  
 }  
 },  
 "syslog": {  
 "global": {  
 "facility": {  
 "all": {  
 "level": "info"  
 },  
 "protocols": {  
 "level": "debug"  
 }  
 }  
 }  
 },  
 "time-zone": "America/New\_York"  
 }   
}   
The config mode

When entering the configuration mode you are navigating inside a tree structure, to enter configuration mode enter the command configure when in operational mode.

vyos@vyos$ configure   
[edit]   
vyos@vyos#   
Note   
When going into configuration mode, prompt changes from $ to #.

All commands executed here are relative to the configuration level you have entered. You can do everything from the top level, but commands will be quite lengthy when manually typing them.

The current hierarchy level can be changed by the edit command.

[edit]   
vyos@vyos# edit interfaces ethernet eth0

[edit interfaces ethernet eth0]   
vyos@vyos#   
You are now in a sublevel relative to interfaces ethernet eth0, all commands executed from this point on are relative to this sublevel. Use either the top or exit command to go back to the top of the hierarchy. You can also use the up command to move only one level up at a time.

show   
The show command within configuration mode will show the working configuration indicating line changes with + for additions, > for replacements and - for deletions.

Example:   
vyos@vyos:~$ configure   
[edit]   
vyos@vyos# show interfaces  
 ethernet eth0 {  
 description MY\_OLD\_DESCRIPTION  
 disable  
 hw-id 00:53:dd:44:3b:03  
 }  
 loopback lo {  
 }   
[edit]   
vyos@vyos# set interfaces ethernet eth0 address dhcp   
[edit]   
vyos@vyos# set interfaces ethernet eth0 description MY\_NEW\_DESCRIPTION [edit]   
vyos@vyos# delete interfaces ethernet eth0 disable   
[edit]   
vyos@vyos# show interfaces  
 ethernet eth0 {   
+ address dhcp   
> description MY\_NEW\_DESCRIPTION

- disable  
 hw-id 00:53:dd:44:3b:03  
 }  
 loopback lo {  
 }   
It is also possible to display all set commands within configuration mode using show | commands   
vyos@vyos# show interfaces ethernet eth0 | commands   
set address dhcp   
set hw-id 00:53:ad:44:3b:03   
These commands are also relative to the level you are inside and only relevant configuration blocks will be displayed when entering a sub-level.

[edit interfaces ethernet eth0]   
vyos@vyos# show  
 address dhcp  
 hw-id 00:53:ad:44:3b:03   
Exiting from the configuration mode is done via the exit command from the top level, executing exit from within a sub-level takes you back to the top level.

[edit interfaces ethernet eth0]   
vyos@vyos# exit   
[edit]   
vyos@vyos# exit   
Warning: configuration changes have not been saved.

Editing the configuration  
The configuration can be edited by the use of set and delete commands from within configuration mode.

set   
Use this command to set the value of a parameter or to create a new element.

Configuration commands are flattened from the tree into ‘one-liner’ commands shown in show configuration commands from operation mode. Commands are relative to the level where they are executed and all redundant information from the current level is removed from the command entered.

[edit]   
vyos@vyos# set interface ethernet eth0 address 192.0.2.100/24   
[edit interfaces ethernet eth0]   
vyos@vyos# set address 203.0.113.6/24   
These two commands above are essentially the same, just executed from different levels in the hierarchy.

delete   
To delete a configuration entry use the delete command, this also deletes all sub-levels under the current level you’ve specified in the delete command. Deleting an entry will also result in the element reverting back to its default value if one exists.

[edit interfaces ethernet eth0]   
vyos@vyos# delete address 192.0.2.100/24

commit   
Any change you do on the configuration, will not take effect until committed using the commit command in configuration mode.

vyos@vyos# commit   
[edit]   
vyos@vyos# exit   
Warning: configuration changes have not been saved.

vyos@vyos:~$  
 save   
Use this command to preserve configuration changes upon reboot. By default it is stored at /config/config.boot. In the case you want to store the configuration file somewhere else, you can add a local path, a SCP address, a FTP address or a TFTP address.

vyos@vyos# save   
Saving configuration to '/config/config.boot'...

Done   
vyos@vyos# save [tab]   
Possible completions:  
 <Enter> Save to system config file  
 <file> Save to file on local machine  
 scp://<user>:<passwd>@<host>:/<file> Save to file on remote machine ftp://<user>:<passwd>@<host>/<file> Save to file on remote machine tftp://<host>/<file> Save to file on remote machine   
vyos@vyos# save tftp://192.168.0.100/vyos-test.config.boot   
Saving configuration to 'tftp://192.168.0.100/vyos-test.config.boot'...

######################################################################## 100.0%   
Done  
 exit [discard]   
Configuration mode can not be exited while uncommitted changes exist. To exit configuration mode without applying changes, the exit discard command must be used.

All changes in the working config will thus be lost.

vyos@vyos# exit   
Cannot exit: configuration modified.

Use 'exit discard' to discard the changes and exit.

[edit]   
vyos@vyos# exit discard  
 commit-confirm <minutes>   
Use this command to temporarily commit your changes and set the number of minutes available for validation. confirm must be entered within those minutes, otherwise the system will reboot into the previous configuration. The default value is 10 minutes.

What if you are doing something dangerous? Suppose you want to setup a firewall, and you are not sure there are no mistakes that will lock you out of your system. You can use confirmed commit. If you issue the commit-confirm command, your changes will be commited, and if you

don’t issue the confirm command in 10 minutes, your system will reboot into previous config revision.

vyos@router# set firewall interface eth0 local name FromWorld   
vyos@router# commit-confirm   
commit confirm will be automatically reboot in 10 minutes unless confirmed   
Proceed? [confirm]y   
[edit]   
vyos@router# confirm   
[edit]   
Note   
A reboot because you did not enter confirm will not take you necessarily to the saved configuration, but to the point before the unfortunate commit.

copy   
Copy a configuration element.

You can copy and remove configuration subtrees. Suppose you set up a firewall ruleset FromWorld with one rule that allows traffic from specific subnet. Now you want to setup a similar rule, but for different subnet. Change your edit level to firewall name FromWorld and use copy rule 10 to rule 20, then modify rule 20.

vyos@router# show firewall name FromWorld  
 default-action drop  
 rule 10 {  
 action accept  
 source {  
 address 203.0.113.0/24  
 }  
 }   
[edit]   
vyos@router# edit firewall name FromWorld   
[edit firewall name FromWorld]   
vyos@router# copy rule 10 to rule 20   
[edit firewall name FromWorld]   
vyos@router# set rule 20 source address 198.51.100.0/24 [edit firewall name FromWorld]   
vyos@router# commit   
[edit firewall name FromWorld]  
 rename   
Rename a configuration element.

You can also rename config subtrees:   
vyos@router# rename rule 10 to rule 5   
[edit firewall name FromWorld]   
vyos@router# commit   
[edit firewall name FromWorld]   
Note that show command respects your edit level and from this level you can view the modified firewall ruleset with just show with no parameters.

vyos@router# show  
 default-action drop  
 rule 5 {  
 action accept  
 source {  
 address 203.0.113.0/24  
 }  
 }  
 rule 20 {  
 action accept  
 source {  
 address 198.51.100.0/24  
 }  
 }  
 comment <config node> “comment text”  
Add comment as an annotation to a configuration node.

The comment command allows you to insert a comment above the <config node> configuration section. When shown, comments are enclosed with /\* and \*/ as open/close delimiters. Comments need to be commited, just like other config changes.

To remove an existing comment from your current configuration, specify an empty string enclosed in double quote marks ("") as the comment text.

Example:   
vyos@vyos# comment firewall all-ping "Yes I know this VyOS is cool" vyos@vyos# commit   
vyos@vyos# show  
 firewall {  
 /\* Yes I know this VyOS is cool \*/  
 all-ping enable  
 broadcast-ping disable  
 ...

}   
Note   
An important thing to note is that since the comment is added on top of the section, it will not appear if the show <section> command is used. With the above example, the show firewall command would return starting after the firewall { line, hiding the comment.

Access opmode from config mode  
When inside configuration mode you are not directly able to execute operational commands.

run   
Access to these commands are possible through the use of the run [command] command. From this command you will have access to everything accessible from operational mode.

Command completion and syntax help with ? and [tab] will also work.

[edit]   
vyos@vyos# run show interfaces   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down

Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 0.0.0.0/0 u/u   
Managing configurations  
VyOS comes with an integrated versioning system for the system configuration. It automatically maintains a backup of every previous configuration which has been committed to the system.

The configurations are versioned locally for rollback but they can also be stored on a remote host for archiving/backup reasons.

Local Archive  
Revisions are stored on disk. You can view, compare and rollback them to any previous revisions if something goes wrong.

show system commit   
View all existing revisions on the local system.

vyos@vyos:~$ show system commit   
0 2015-03-30 08:53:03 by vyos via cli   
1 2015-03-30 08:52:20 by vyos via cli   
2 2015-03-26 21:26:01 by root via boot-config-loader   
3 2015-03-26 20:43:18 by root via boot-config-loader   
4 2015-03-25 11:06:14 by root via boot-config-loader   
5 2015-03-25 01:04:28 by root via boot-config-loader   
6 2015-03-25 00:16:47 by vyos via cli   
7 2015-03-24 23:43:45 by root via boot-config-loader  
 set system config-management commit-revisions <N>   
You can specify the number of revisions stored on disk. N can be in the range of 0 - 65535. When the number of revisions exceeds the configured value, the oldest revision is removed. The default setting for this value is to store 100 revisions locally.

Compare configurations  
VyOS lets you compare different configurations.

compare <saved | N> <M>   
Use this command to spot what the differences are between different configurations.

vyos@vyos# compare [tab]   
Possible completions:  
 <Enter> Compare working & active configurations saved Compare working & saved configurations <N> Compare working with revision N  
 <N> <M> Compare revision N with M  
 Revisions:  
 0 2013-12-17 20:01:37 root by boot-config-loader 1 2013-12-13 15:59:31 root by boot-config-loader 2 2013-12-12 21:56:22 vyos by cli  
 3 2013-12-12 21:55:11 vyos by cli  
 4 2013-12-12 21:27:54 vyos by cli  
 5 2013-12-12 21:23:29 vyos by cli  
 6 2013-12-12 21:13:59 root by boot-config-loader

7 2013-12-12 16:25:19 vyos by cli  
 8 2013-12-12 15:44:36 vyos by cli  
 9 2013-12-12 15:42:07 root by boot-config-loader  
 10 2013-12-12 15:42:06 root by init   
The command compare allows you to compare different type of configurations. It also lets you compare different revisions through the compare N M command, where N and M are revision numbers. The output will describe how the configuration N is when compared to M indicating with a plus sign (+) the additional parts N has when compared to M, and indicating with a minus sign (-) the lacking parts N misses when compared to M.

vyos@vyos# compare 0 6   
[edit interfaces]   
+dummy dum1 {   
+ address 10.189.0.1/31   
+}   
[edit interfaces ethernet eth0]   
+vif 99 {   
+ address 10.199.0.1/31   
+}  
-vif 900 {  
- address 192.0.2.4/24  
-}  
 show system commit diff <number>   
Show commit revision difference.

The command above also lets you see the difference between two commits. By default the difference with the running config is shown.

vyos@router# run show system commit diff 4   
[edit system]   
+ipv6 {   
+ disable-forwarding   
+}   
This means four commits ago we did set system ipv6 disable-forwarding.

Rollback Changes  
You can rollback configuration changes using the rollback command. This will apply the selected revision and trigger a system reboot.

rollback <N>   
Rollback to revision N (currently requires reboot) vyos@vyos# compare 1   
[edit system]   
>host-name vyos-1   
[edit]

vyos@vyos# rollback 1   
Proceed with reboot? [confirm][y]   
Broadcast message from root@vyos-1 (pts/0) (Tue Dec 17 21:07:45 2013):

The system is going down for reboot NOW!

Remote Archive  
VyOS can upload the configuration to a remote location after each call to commit. You will have to set the commit-archive location. TFTP, FTP, SCP and SFTP servers are supported. Every time a commit is successfull the config.boot file will be copied to the defined destination(s). The filename used on the remote host will be config.boot-hostname.YYYYMMDD\_HHMMSS.

set system config-management commit-archive location <URI> Specify remote location of commit archive as any of the below URI• scp://<user>:<passwd>@<host>:/<dir>  
• sftp://<user>:<passwd>@<host>/<dir>  
• ftp://<user>:<passwd>@<host>/<dir>  
• tftp://<host>/<dir>   
Note   
The number of revisions don’t affect the commit-archive.

Note   
You may find VyOS not allowing the secure connection because it cannot verify the legitimacy of the remote server. You can use the workaround below to quickly add the remote host’s SSH fingerprint to your ~/.ssh/known\_hosts file:   
vyos@vyos# ssh-keyscan <host> >> ~/.ssh/known\_hosts   
Saving and loading manually  
You can use the save and load commands if you want to manually manage specific configuration files.

When using the save command, you can add a specific location where to store your   
configuration file. And, when needed it, you will be able to load it with the load command: load <URI>   
Use this command to load a configuration which will replace the running configuration. Define the location of the configuration file to be loaded. You can use a path to a local file, an SCP address, an SFTP address, an FTP address, an HTTP address, an HTTPS address or a TFTP address.

vyos@vyos# load   
Possible completions:  
 <Enter> Load from system config file  
 <file> Load from file on local machine  
 scp://<user>:<passwd>@<host>:/<file> Load from file on remote machine  
 sftp://<user>:<passwd>@<host>/<file> Load from file on remote machine  
 ftp://<user>:<passwd>@<host>/<file> Load from file on remote machine  
 http://<host>/<file> Load from file on remote machine  
 https://<host>/<file> Load from file on remote machine  
 tftp://<host>/<file> Load from file on remote machine   
Restore Default  
In the case you want to completely delete your configuration and restore the default one, you can enter the following command in configuration mode:   
load /opt/vyatta/etc/config.boot.default

You will be asked if you want to continue. If you accept, you will have to use commit if you want to make the changes active.

Then you may want to save in order to delete the saved configuration too.

Note   
If you are remotely connected, you will lose your connection. You may want to copy first the config, edit it to ensure connectivity, and load the edited config.

PreviousNext

Container  
The VyOS container implementation is based on Podman<https://podman.io/> as a deamonless container engine.

Configuration  
 set container name <name> image   
Sets the image name in the hub registry   
set container name mysql-server image mysql:8.0   
If a registry is not specified, Docker.io will be used as the container registry unless an alternative registry is specified using set container registry <name> or the registry is included in the image name   
set container name mysql-server image quay.io/mysql:8.0  
 set container name <name> allow-host-networks   
Allow host networking in a container. The network stack of the container is not isolated from the host and will use the host IP.

The following commands translate to “–net host” when the container is created   
Note   
allow-host-networks cannot be used with network  
 set container name <name> network <networkname>   
Attaches user-defined network to a container. Only one network must be specified and must already exist.

set container name <name> network <networkname> address <address>   
Optionally set a specific static IPv4 or IPv6 address for the container. This address must be within the named network prefix.

Note   
The first IP in the container network is reserved by the engine and cannot be used  
 set container name <name> description <text>   
Set a container description  
 set container name <name> environment <key> value <value>   
Add custom environment variables. Multiple environment variables are allowed. The following commands translate to “-e key=value” when the container is created.

set container name mysql-server environment MYSQL\_DATABASE value 'zabbix'   
set container name mysql-server environment MYSQL\_USER value 'zabbix'   
set container name mysql-server environment MYSQL\_PASSWORD value 'zabbix\_pwd' set container name mysql-server environment MYSQL\_ROOT\_PASSWORD value 'root\_pwd' set container name <name> port <portname> source <portnumber>  
 set container name <name> port <portname> destination <portnumber>

set container name <name> port <portname> protocol <tcp | udp> Publish a port for the container.

set container name zabbix-web-nginx-mysql port http source 80   
set container name zabbix-web-nginx-mysql port http destination 8080 set container name zabbix-web-nginx-mysql port http protocol tcp  
 set container name <name> volume <volumename> source <path>  
 set container name <name> volume <volumename> destination <path> Mount a volume into the container   
set container name coredns volume 'corefile' source /config/coredns/Corefile set container name coredns volume 'corefile' destination /etc/Corefile set container name <name> volume <volumename> mode <ro | rw>   
Volume is either mounted as rw (read-write - default) or ro (read-only) set container name <name> restart [no | on-failure | always]   
Set the restart behavior of the container.

• no: Do not restart containers on exit  
• on-failure: Restart containers when they exit with a non-zero exit code, retrying indefinitely (default)  
• always: Restart containers when they exit, regardless of status, retrying indefinitely set container name <name> memory <MB>   
Constrain the memory available to the container.

Default is 512 MB. Use 0 MB for unlimited memory.

set container name <name> device <devicename> source <path> set container name <name> device <devicename> destination <path> Add a host device to the container.

container name <name> cap-add <text>   
Set container capabilities or permissions.

• net-admin: Network operations (interface, firewall, routing tables)   
 net-bind-service: Bind a socket to privileged ports (port numbers less than 1024)•  
• net-raw: Permission to create raw network sockets  
• setpcap: Capability sets (from bounded or inherited set)  
• sys-admin: Administation operations (quotactl, mount, sethostname, setdomainame) sys-time: Permission to set system clock•  
 set container name <name> disable   
Disable a container.

set container network <networkname>   
Creates a named container network  
 set container registry <name>   
Adds registry to list of unqualified-search-registries. By default, for any image that does not include the registry in the image name, Vyos will use docker.io as the container registry.

Operation Commands  
 add container image <containername>   
Pull a new image for container  
 show container   
Show the list of all active containers.

show container image   
Show the local container images.

show container log <containername>   
Show logs from a given container  
 show container network   
Show a list available container networks  
 restart container <containername>   
Restart a given container  
 update container image <containername>   
Update container image   
Example Configuration  
For the sake of demonstration, example #1 in the official documentation to the declarative VyOS CLI syntax.

set container network zabbix-net prefix 172.20.0.0/16   
set container network zabbix-net description 'Network for Zabbix component containers'

set container name mysql-server image mysql:8.0 set container name mysql-server network zabbix-net

set container name mysql-server environment 'MYSQL\_DATABASE' value 'zabbix'   
set container name mysql-server environment 'MYSQL\_USER' value 'zabbix'   
set container name mysql-server environment 'MYSQL\_PASSWORD' value 'zabbix\_pwd' set container name mysql-server environment 'MYSQL\_ROOT\_PASSWORD' value 'root\_pwd'

set container name zabbix-java-gateway image zabbix/zabbix-java-gateway:alpine-5.2-latest set container name zabbix-java-gateway network zabbix-net

set container name zabbix-server-mysql image zabbix/zabbix-server-mysql:alpine-5.2-latest set container name zabbix-server-mysql network zabbix-net

set container name zabbix-server-mysql environment 'DB\_SERVER\_HOST' value 'mysql-server' set container name zabbix-server-mysql environment 'MYSQL\_DATABASE' value 'zabbix' set container name zabbix-server-mysql environment 'MYSQL\_USER' value 'zabbix'   
set container name zabbix-server-mysql environment 'MYSQL\_PASSWORD' value 'zabbix\_pwd' set container name zabbix-server-mysql environment 'MYSQL\_ROOT\_PASSWORD' value 'root\_pwd'   
set container name zabbix-server-mysql environment 'ZBX\_JAVAGATEWAY' value 'zabbix-java-gateway'

set container name zabbix-server-mysql port zabbix source 10051 set container name zabbix-server-mysql port zabbix destination 10051

set container name zabbix-web-nginx-mysql image zabbix/zabbix-web-nginx-mysql:alpine-5.2-latest

set container name zabbix-web-nginx-mysql network zabbix-net

set container name zabbix-web-nginx-mysql environment 'MYSQL\_DATABASE' value 'zabbix' set container name zabbix-web-nginx-mysql environment 'ZBX\_SERVER\_HOST' value 'zabbix-server-mysql'   
set container name zabbix-web-nginx-mysql environment 'DB\_SERVER\_HOST' value 'mysql-server'   
set container name zabbix-web-nginx-mysql environment 'MYSQL\_USER' value 'zabbix' set container name zabbix-web-nginx-mysql environment 'MYSQL\_PASSWORD' value 'zabbix\_pwd'   
set container name zabbix-web-nginx-mysql environment 'MYSQL\_ROOT\_PASSWORD' value 'root\_pwd'

set container name zabbix-web-nginx-mysql port http source 80   
set container name zabbix-web-nginx-mysql port http destination 8080

Firewall  
Attention

Starting from VyOS 1.4-rolling-202308040557, a new firewall structure can be found on all vyos installations.

Netfilter based  
With VyOS being based on top of Linux and its kernel, the Netfilter project created the iptables and now the successor nftables for the Linux kernel to work directly on the data flows. This now extends the concept of zone-based security to allow for manipulating the data at multiple stages once accepted by the network interface and the driver before being handed off to the destination (e.g. a web server OR another device).

A simplified traffic flow, based on Netfilter packet flow, is shown next, in order to have a full view and understanding of how packets are processed, and what possible paths can take.

../../\_images/firewall-gral-packet-flow.png   
Main notes regarding this packet flow and terminology used in VyOS firewall:

Bridge Port?: choose appropiate path based on if interface were the packet was received is part of a bridge, or not.

If interface were the packet was received isn’t part of a bridge, then packet is processed at the IP Layer:

Prerouting: several actions can be done in this stage, and currently these actions are defined in different parts in vyos configuration. Order is important, and all these actions are performed

before any actions define under firewall section. Relevant configuration that acts in this stage are:   
Conntrack Ignore: rules defined under set system conntrack ignore [ipv4 | ipv6] ....

Policy Route: rules defined under set policy [route | route6] ....

Destination NAT: rules defined under set [nat | nat66] destination....

Destination is the router?: choose appropiate path based on destination IP address. Transit forward continunes to forward, while traffic that destination IP address is configured on the router continues to input.

Input: stage where traffic destinated to the router itself can be filtered and controlled. This is where all rules for securing the router should take place. This includes ipv4 and ipv6 filtering rules, defined in:   
set firewall ipv4 input filter ....

set firewall ipv6 input filter ....

Forward: stage where transit traffic can be filtered and controlled. This includes ipv4 and ipv6 filtering rules, defined in:   
set firewall ipv4 forward filter ....

set firewall ipv6 forward filter ....

Output: stage where traffic that is originated by the router itself can be filtered and controlled. Bare in mind that this traffic can be a new connection originted by a internal process running on VyOS router, such as NTP, or can be a response to traffic received externaly through inputt (for example response to an ssh login attempt to the router). This includes ipv4 and ipv6 filtering rules, defined in:   
set firewall ipv4 input filter ....

set firewall ipv6 output filter ....

Postrouting: as in Prerouting, several actions defined in different parts of VyOS configuration are performed in this stage. This includes:   
Source NAT: rules defined under set [nat | nat66] destination....

If interface were the packet was received is part of a bridge, then packet is processed at the Bridge Layer, which contains a ver basic setup where for bridge filtering:

Forward (Bridge): stage where traffic that is trasspasing through the bridge is filtered and controlled:

set firewall bridge forward filter ....

Main structure VyOS firewall cli is shown next:

- set firewall  
 \* bridge  
 - forward  
 + filter  
 \* flowtable  
 - custom\_flow\_table  
 + ...

\* global-options  
 + all-ping  
 + broadcast-ping  
 + ...

\* group  
 - address-group  
 - ipv6-address-group  
 - network-group  
 - ipv6-network-group  
 - interface-group  
 - mac-group  
 - port-group  
 - domain-group  
 \* ipv4  
 - forward  
 + filter  
 - input  
 + filter  
 - output  
 + filter  
 - name  
 + custom\_name  
 \* ipv6  
 - forward  
 + filter  
 - input  
 + filter

- output  
 + filter  
 - ipv6-name  
 + custom\_name  
 \* zone  
 - custom\_zone\_name  
 + ...

Please, refer to appropiate section for more information about firewall configuration:

Global Options Firewall Configuration   
Firewall groups   
Bridge Firewall Configuration   
IPv4 Firewall Configuration   
IPv6 Firewall Configuration   
Flowtables Firewall Configuration   
Zone Based Firewall   
Note

For more information of Netfilter hooks and Linux networking packet flows can be found in Netfilter-Hooks

Legacy Firewall  
Firewall Configuration (Deprecated)   
Traditionally firewalls weere configured with the concept of data going in and out of an interface. The router just listened to the data flowing through and responding as required if it was directed at the router itself.

To configure VyOS with the legacy firewall configuration

As the example image below shows, the device was configured with rules blocking inbound or outbound traffic on each interface.

../../\_images/firewall-traditional.png   
Zone-based firewall  
Zone Based Firewall   
With zone-based firewalls a new concept was implemented, in addtion to the standard in and out traffic flows, a local flow was added. This local was for traffic originating and destined to the router itself. Which means additional rules were required to secure the firewall itself from the network, in addition to the existing inbound and outbound rules from the traditional concept above.

To configure VyOS with the zone-based firewall configuration

As the example image below shows, the device now needs rules to allow/block traffic to or from the services running on the device that have open connections on that interface.

../../\_images/firewall-zonebased.png   
Global Options Firewall Configuration  
Overview  
Some firewall settings are global and have an affect on the whole system. In this section there’s useful information about these global-options that can be configured using vyos cli.

Configuration commands covered in this section:  
 set firewall global-options …  
Configuration  
 set firewall global-options all-ping [enable | disable]   
By default, when VyOS receives an ICMP echo request packet destined for itself, it will answer with an ICMP echo reply, unless you avoid it through its firewall.

With the firewall you can set rules to accept, drop or reject ICMP in, out or local traffic. You can also use the general firewall all-ping command. This command affects only to LOCAL (packets destined for your VyOS system), not to IN or OUT traffic.

Note   
firewall global-options all-ping affects only to LOCAL and it always behaves in the most restrictive way   
set firewall global-options all-ping enable   
When the command above is set, VyOS will answer every ICMP echo request addressed to itself, but that will only happen if no other rule is applied dropping or rejecting local echo requests. In case of conflict, VyOS will not answer ICMP echo requests.

set firewall global-options all-ping disable   
When the command above is set, VyOS will answer no ICMP echo request addressed to itself at all, no matter where it comes from or whether more specific rules are being applied to accept them.

set firewall global-options broadcast-ping [enable | disable]   
This setting enable or disable the response of icmp broadcast messages. The following system parameter will be altered:   
net.ipv4.icmp\_echo\_ignore\_broadcasts  
 set firewall global-options ip-src-route [enable | disable]

set firewall global-options ipv6-src-route [enable | disable]   
This setting handle if VyOS accept packets with a source route option. The following system parameter will be altered:   
net.ipv4.conf.all.accept\_source\_route   
net.ipv6.conf.all.accept\_source\_route  
 set firewall global-options receive-redirects [enable | disable]  
 set firewall global-options ipv6-receive-redirects [enable | disable]   
enable or disable of ICMPv4 or ICMPv6 redirect messages accepted by VyOS. The following system parameter will be altered:   
net.ipv4.conf.all.accept\_redirects   
net.ipv6.conf.all.accept\_redirects  
 set firewall global-options send-redirects [enable | disable]   
enable or disable ICMPv4 redirect messages send by VyOS The following system parameter will be altered:   
net.ipv4.conf.all.send\_redirects  
 set firewall global-options log-martians [enable | disable]   
enable or disable the logging of martian IPv4 packets. The following system parameter will be altered:   
net.ipv4.conf.all.log\_martians  
 set firewall global-options source-validation [strict | loose | disable]   
Set the IPv4 source validation mode. The following system parameter will be altered:   
net.ipv4.conf.all.rp\_filter  
 set firewall global-options syn-cookies [enable | disable]   
Enable or Disable if VyOS use IPv4 TCP SYN Cookies. The following system parameter will be altered:   
net.ipv4.tcp\_syncookies  
 set firewall global-options twa-hazards-protection [enable | disable]   
Enable or Disable VyOS to be RFC 1337 conform. The following system parameter will be altered:

net.ipv4.tcp\_rfc1337

Firewall groups  
Configuration  
Firewall groups represent collections of IP addresses, networks, ports, mac addresses, domains or interfaces. Once created, a group can be referenced by firewall, nat and policy route rules as either a source or destination matcher, and/or as inbound/outbound in the case of interface group.

Address Groups  
In an address group a single IP address or IP address ranges are defined.

set firewall group address-group <name> address [address | address range] set firewall group ipv6-address-group <name> address <address>   
Define a IPv4 or a IPv6 address group

set firewall group address-group ADR-INSIDE-v4 address 192.168.0.1 set firewall group address-group ADR-INSIDE-v4 address 10.0.0.1-10.0.0.8 set firewall group ipv6-address-group ADR-INSIDE-v6 address 2001:db8::1 set firewall group address-group <name> description <text>  
 set firewall group ipv6-address-group <name> description <text>   
Provide a IPv4 or IPv6 address group description

Network Groups  
While network groups accept IP networks in CIDR notation, specific IP addresses can be added as a 32-bit prefix. If you foresee the need to add a mix of addresses and networks, the network group is recommended.

set firewall group network-group <name> network <CIDR> set firewall group ipv6-network-group <name> network <CIDR> Define a IPv4 or IPv6 Network group.

set firewall group network-group NET-INSIDE-v4 network 192.168.0.0/24 set firewall group network-group NET-INSIDE-v4 network 192.168.1.0/24 set firewall group ipv6-network-group NET-INSIDE-v6 network 2001:db8::/64 set firewall group network-group <name> description <text>  
 set firewall group ipv6-network-group <name> description <text>   
Provide an IPv4 or IPv6 network group description.

Interface Groups  
An interface group represents a collection of interfaces.

set firewall group interface-group <name> interface <text> Define an interface group. Wildcard are accepted too.

set firewall group interface-group LAN interface bond1001 set firewall group interface-group LAN interface eth3\*  
 set firewall group interface-group <name> description <text> Provide an interface group description

Port Groups  
A port group represents only port numbers, not the protocol. Port groups can be referenced for either TCP or UDP. It is recommended that TCP and UDP groups are created separately to avoid accidentally filtering unnecessary ports. Ranges of ports can be specified by using -.

set firewall group port-group <name> port [portname | portnumber | startport-endport] Define a port group. A port name can be any name defined in /etc/services. e.g.: http

set firewall group port-group PORT-TCP-SERVER1 port http   
set firewall group port-group PORT-TCP-SERVER1 port 443   
set firewall group port-group PORT-TCP-SERVER1 port 5000-5010 set firewall group port-group <name> description <text>   
Provide a port group description.

MAC Groups  
A mac group represents a collection of mac addresses.

set firewall group mac-group <name> mac-address <mac-address> Define a mac group.

set firewall group mac-group MAC-G01 mac-address 88:a4:c2:15:b6:4f set firewall group mac-group MAC-G01 mac-address 4c:d5:77:c0:19:81 set firewall group mac-group <name> description <text>   
Provide a mac group description.

Domain Groups  
A domain group represents a collection of domains.

set firewall group domain-group <name> address <domain> Define a domain group.

set firewall group domain-group DOM address example.com set firewall group domain-group <name> description <text> Provide a domain group description.

Examples  
As said before, once firewall groups are created, they can be referenced either in firewall, nat, nat66 and/or policy-route rules.

Here is an example were multiple groups are created:

set firewall group address-group SERVERS address 198.51.100.101   
set firewall group address-group SERVERS address 198.51.100.102   
set firewall group network-group TRUSTEDv4 network 192.0.2.0/30 set firewall group network-group TRUSTEDv4 network 203.0.113.128/25 set firewall group ipv6-network-group TRUSTEDv6 network 2001:db8::/64 set firewall group interface-group LAN interface eth2.2001   
set firewall group interface-group LAN interface bon0   
set firewall group port-group PORT-SERVERS port http   
set firewall group port-group PORT-SERVERS port 443   
set firewall group port-group PORT-SERVERS port 5000-5010   
And next, some configuration example where groups are used:

set firewall ipv4 input filter rule 10 action accept   
set firewall ipv4 input filter rule 10 inbound-interface group !LAN   
set firewall ipv4 forward filter rule 20 action accept   
set firewall ipv4 forward filter rule 20 source group network-group TRUSTEDv4   
set firewall ipv6 input filter rule 10 action accept   
set firewall ipv6 input filter rule 10 source-group network-group TRUSTEDv6   
set nat destination rule 101 inbound-interface group LAN   
set nat destination rule 101 destination group address-group SERVERS   
set nat destination rule 101 protocol tcp   
set nat destination rule 101 destination group port-group PORT-SERVERS   
set nat destination rule 101 translation address 203.0.113.250   
set policy route PBR rule 201 destination group port-group PORT-SERVERS   
set policy route PBR rule 201 protocol tcp   
set policy route PBR rule 201 set table 15   
Operation-mode  
 show firewall group <name>   
Overview of defined groups. You see the type, the members, and where the group is used.

vyos@ZBF-15-CLean:~$ show firewall group   
Firewall Groups

Name Type References Members  
------------ ------------------ ---------------------- ----------------  
SERVERS address\_group nat-destination-101 198.51.100.101 198.51.100.102   
LAN interface\_group ipv4-input-filter-10 bon0  
 nat-destination-101 eth2.2001   
TRUSTEDv6 ipv6\_network\_group ipv6-input-filter-10 2001:db8::/64 TRUSTEDv4 network\_group ipv4-forward-filter-20 192.0.2.0/30

203.0.113.128/25   
PORT-SERVERS port\_group route-PBR-201 443 nat-destination-101 5000-5010 http   
vyos@ZBF-15-CLean:~$

Overview  
In this section there’s useful information of all firewall configuration that can be done regarding bridge, and appropiate op-mode commands. Configuration commands covered in this section:

set firewall bridge …  
From main structure defined in Firewall Overview in this section you can find detailed information only for the next part of the general structure:

- set firewall  
 \* bridge  
 - forward  
 + filter  
 - name  
 + custom\_name   
Traffic which is received by the router on an interface which is member of a bridge is processed on the Bridge Layer. A simplified packet flow diagram for this layer is shown next:

../../\_images/firewall-bridge-packet-flow.png   
For traffic that needs to be forwared internally by the bridge, base chain is is forward, and it’s base command for filtering is set firewall bridge forward filter ...

IPv4 Firewall Configuration  
Overview  
In this section there’s useful information of all firewall configuration that can be done regarding IPv4, and appropiate op-mode commands. Configuration commands covered in this section: set firewall ipv4 …  
From main structure defined in Firewall Overview in this section you can find detailed   
information only for the next part of the general structure:  
- set firewall  
 \* ipv4  
 - forward  
 + filter  
 - input  
 + filter  
 - output  
 + filter  
 - name

+ custom\_name   
For transit traffic, which is received by the router and forwarded, base chain is forward. A simplified packet flow diagram for transit traffic is shown next:

Where firewall base chain to configure firewall filtering rules for transit traffic is set firewall ipv4 forward filter ..., which happens in stage 5, highlightened with red color.

For traffic towards the router itself, base chain is input, while traffic originated by the router, base chain is output. A new simplified packet flow diagram is shown next, which shows the path for traffic destinated to the router itself, and traffic generated by the router (starting from circle number 6):

Base chain is for traffic toward the router is set firewall ipv4 input filter ...

And base chain for traffic generated by the router is set firewall ipv4 output filter ...

Note   
Important note about default-actions: If default action for any base chain is not defined, then the default action is set to accept for that chain. For custom chains, if default action is not defined, then the default-action is set to drop   
Custom firewall chains can be created, with commands set firewall ipv4 name <name> .... In order to use such custom chain, a rule with action jump, and the appropiate target should be defined in a base chain.

Firewall - IPv4 Rules  
For firewall filtering, firewall rules needs to be created. Each rule is numbered, has an action to apply if the rule is matched, and the ability to specify multiple criteria matchers. Data packets go through the rules from 1 - 999999, so order is crucial. At the first match the action of the rule will be executed.

Actions  
If a rule is defined, then an action must be defined for it. This tells the firewall what to do if all criteria matchers defined for such rule do match.

The action can be :  
• accept: accept the packet.

• continue: continue parsing next rule.

• drop: drop the packet.

• reject: reject the packet.

• jump: jump to another custom chain.

• return: Return from the current chain and continue at the next rule of the last chain.• queue: Enqueue packet to userspace.

• synproxy: synproxy the packet.

set firewall ipv4 forward filter rule <1-999999> action [accept | continue | drop | jump | queue | reject | return | synproxy]  
 set firewall ipv4 input filter rule <1-999999> action [accept | continue | drop | jump | queue | reject | return | synproxy]  
 set firewall ipv4 output filter rule <1-999999> action [accept | continue | drop | jump | queue | reject | return]

set firewall ipv4 name <name> rule <1-999999> action [accept | continue | drop | jump | queue | reject | return]   
This required setting defines the action of the current rule. If action is set to jump, then jump-target is also needed.

set firewall ipv4 forward filter rule <1-999999> jump-target <text>  
 set firewall ipv4 input filter rule <1-999999> jump-target <text>  
 set firewall ipv4 output filter rule <1-999999> jump-target <text>  
 set firewall ipv4 name <name> rule <1-999999> jump-target <text>   
To be used only when action is set to jump. Use this command to specify jump target.

Also, default-action is an action that takes place whenever a packet does not match any rule in it’s chain. For base chains, possible options for default-action are accept or drop.

set firewall ipv4 forward filter default-action [accept | drop]  
 set firewall ipv4 input filter default-action [accept | drop]  
 set firewall ipv4 output filter default-action [accept | drop]  
 set firewall ipv4 name <name> default-action [accept | drop | jump | queue | reject | return] This set the default action of the rule-set if no rule matched a packet criteria. If defacult-action is set to jump, then default-jump-target is also needed. Note that for base chains, default action can only be set to accept or drop, while on custom chain, more actions are available.

set firewall ipv4 name <name> default-jump-target <text>   
To be used only when defult-action is set to jump. Use this command to specify jump target for default rule.

Note   
Important note about default-actions: If default action for any base chain is not defined, then the default action is set to accept for that chain. For custom chains, if default action is not defined, then the default-action is set to drop   
Firewall Logs  
Logging can be enable for every single firewall rule. If enabled, other log options can be defined.

set firewall ipv4 forward filter rule <1-999999> log [disable | enable] set firewall ipv4 input filter rule <1-999999> log [disable | enable] set firewall ipv4 output filter rule <1-999999> log [disable | enable] set firewall ipv4 name <name> rule <1-999999> log [disable | enable] Enable or disable logging for the matched packet.

set firewall ipv4 forward filter enable-default-log  
 set firewall ipv4 input filter enable-default-log  
 set firewall ipv4 output filter enable-default-log  
 set firewall ipv4 name <name> enable-default-log   
Use this command to enable the logging of the default action on the specified chain.

set firewall ipv4 forward filter rule <1-999999> log-options level [emerg | alert | crit | err | warn | notice | info | debug]  
 set firewall ipv4 input filter rule <1-999999> log-options level [emerg | alert | crit | err | warn | notice | info | debug]  
 set firewall ipv4 output filter rule <1-999999> log-options level [emerg | alert | crit | err | warn | notice | info | debug]

set firewall ipv4 name <name> rule <1-999999> log-options level [emerg | alert | crit | err | warn | notice | info | debug]   
Define log-level. Only applicable if rule log is enable.

set firewall ipv4 forward filter rule <1-999999> log-options group <0-65535> set firewall ipv4 input filter rule <1-999999> log-options group <0-65535> set firewall ipv4 output filter rule <1-999999> log-options group <0-65535> set firewall ipv4 name <name> rule <1-999999> log-options group <0-65535> Define log group to send message to. Only applicable if rule log is enable.

set firewall ipv4 forward filter rule <1-999999> log-options snapshot-length <0-9000> set firewall ipv4 input filter rule <1-999999> log-options snapshot-length <0-9000> set firewall ipv4 output filter rule <1-999999> log-options snapshot-length <0-9000> set firewall ipv4 name <name> rule <1-999999> log-options snapshot-length <0-9000> Define length of packet payload to include in netlink message. Only applicable if rule log is enable and log group is defined.

set firewall ipv4 forward filter rule <1-999999> log-options queue-threshold <0-65535> set firewall ipv4 input filter rule <1-999999> log-options queue-threshold <0-65535> set firewall ipv4 output filter rule <1-999999> log-options queue-threshold <0-65535> set firewall ipv4 name <name> rule <1-999999> log-options queue-threshold <0-65535> Define number of packets to queue inside the kernel before sending them to userspace. Only applicable if rule log is enable and log group is defined.

Firewall Description  
For reference, a description can be defined for every single rule, and for every defined custom chain.

set firewall ipv4 name <name> description <text>   
Provide a rule-set description to a custom firewall chain.

set firewall ipv4 forward filter rule <1-999999> description <text> set firewall ipv4 input filter rule <1-999999> description <text> set firewall ipv4 output filter rule <1-999999> description <text> set firewall ipv4 name <name> rule <1-999999> description <text> Provide a description for each rule.

Rule Status  
When defining a rule, it is enable by default. In some cases, it is useful to just disable the rule, rather than removing it.

set firewall ipv4 forward filter rule <1-999999> disable  
 set firewall ipv4 input filter rule <1-999999> disable  
 set firewall ipv4 output filter rule <1-999999> disable  
 set firewall ipv4 name <name> rule <1-999999> disable Command for disabling a rule but keep it in the configuration.

Matching criteria  
There are a lot of matching criteria against which the package can be tested.

set firewall ipv4 forward filter rule <1-999999> connection-status nat [destination | source] set firewall ipv4 input filter rule <1-999999> connection-status nat [destination | source] set firewall ipv4 output filter rule <1-999999> connection-status nat [destination | source] set firewall ipv4 name <name> rule <1-999999> connection-status nat [destination | source]

Match criteria based on nat connection status.

set firewall ipv4 forward filter rule <1-999999> connection-mark <1-2147483647> set firewall ipv4 input filter rule <1-999999> connection-mark <1-2147483647> set firewall ipv4 output filter rule <1-999999> connection-mark <1-2147483647> set firewall ipv4 name <name> rule <1-999999> connection-mark <1-2147483647> Match criteria based on connection mark.

set firewall ipv4 forward filter rule <1-999999> source address [address | addressrange | CIDR] set firewall ipv4 input filter rule <1-999999> source address [address | addressrange | CIDR] set firewall ipv4 output filter rule <1-999999> source address [address | addressrange | CIDR] set firewall ipv4 name <name> rule <1-999999> source address [address | addressrange | CIDR]  
 set firewall ipv4 forward filter rule <1-999999> destination address [address | addressrange | CIDR]  
 set firewall ipv4 input filter rule <1-999999> destination address [address | addressrange | CIDR]  
 set firewall ipv4 output filter rule <1-999999> destination address [address | addressrange | CIDR]  
 set firewall ipv4 name <name> rule <1-999999> destination address [address | addressrange | CIDR]   
Match criteria based on source and/or destination address. This is similar to the network groups part, but here you are able to negate the matching addresses.

set firewall ipv4 name FOO rule 50 source address 192.0.2.10-192.0.2.11   
# with a '!' the rule match everything except the specified subnet   
set firewall ipv4 input filter FOO rule 51 source address !203.0.113.0/24  
 set firewall ipv4 forward filter rule <1-999999> source address-mask [address]  
 set firewall ipv4 input filter rule <1-999999> source address-mask [address]  
 set firewall ipv4 output filter rule <1-999999> source address-mask [address]  
 set firewall ipv4 name <name> rule <1-999999> source address-mask [address]  
 set firewall ipv4 forward filter rule <1-999999> destination address-mask [address]  
 set firewall ipv4 input filter rule <1-999999> destination address-mask [address]  
 set firewall ipv4 output filter rule <1-999999> destination address-mask [address]  
 set firewall ipv4 name <name> rule <1-999999> destination address-mask [address]   
An arbitrary netmask can be applied to mask addresses to only match against a specific portion. This functions for both individual addresses and address groups.

# Match any IPv4 address with `11` as the 2nd octet and `13` as the forth octet set firewall ipv4 name FOO rule 100 destination address 0.11.0.13   
set firewall ipv4 name FOO rule 100 destination address-mask 0.255.0.255 set firewall ipv4 forward filter rule <1-999999> source fqdn <fqdn>  
 set firewall ipv4 input filter rule <1-999999> source fqdn <fqdn>  
 set firewall ipv4 output filter rule <1-999999> source fqdn <fqdn>  
 set firewall ipv4 name <name> rule <1-999999> source fqdn <fqdn>  
 set firewall ipv4 forward filter rule <1-999999> destination fqdn <fqdn> set firewall ipv4 input filter rule <1-999999> destination fqdn <fqdn>  
 set firewall ipv4 output filter rule <1-999999> destination fqdn <fqdn>

set firewall ipv4 name <name> rule <1-999999> destination fqdn <fqdn>   
Specify a Fully Qualified Domain Name as source/destination matcher. Ensure router is able to resolve such dns query.

set firewall ipv4 forward filter rule <1-999999> source geoip country-code <country>  
 set firewall ipv4 input filter rule <1-999999> source geoip country-code <country>  
 set firewall ipv4 output filter rule <1-999999> source geoip country-code <country>  
 set firewall ipv4 name <name> rule <1-999999> source geoip country-code <country> set firewall ipv4 forward filter rule <1-999999> destination geoip country-code <country> set firewall ipv4 input filter rule <1-999999> destination geoip country-code <country> set firewall ipv4 output filter rule <1-999999> destination geoip country-code <country> set firewall ipv4 name <name> rule <1-999999> destination geoip country-code <country> set firewall ipv4 forward filter rule <1-999999> source geoip inverse-match  
 set firewall ipv4 input filter rule <1-999999> source geoip inverse-match  
 set firewall ipv4 output filter rule <1-999999> source geoip inverse-match  
 set firewall ipv4 name <name> rule <1-999999> source geoip inverse-match  
 set firewall ipv4 forward filter rule <1-999999> destination geoip inverse-match  
 set firewall ipv4 input filter rule <1-999999> destination geoip inverse-match  
 set firewall ipv4 output filter rule <1-999999> destination geoip inverse-match  
 set firewall ipv4 name <name> rule <1-999999> destination geoip inverse-match   
Match IP addresses based on its geolocation. More info: geoip matching. Use inverse-match to match anything except the given country-codes.

Data is provided by DB-IP.com under CC-BY-4.0 license. Attribution required, permits   
redistribution so we can include a database in images(~3MB compressed). Includes cron script (manually callable by op-mode update geoip) to keep database and rules updated.

set firewall ipv4 forward filter rule <1-999999> source mac-address <mac-address> set firewall ipv4 input filter rule <1-999999> source mac-address <mac-address> set firewall ipv4 output filter rule <1-999999> source mac-address <mac-address> set firewall ipv4 name <name> rule <1-999999> source mac-address <mac-address> Only in the source criteria, you can specify a mac-address.

set firewall ipv4 input filter rule 100 source mac-address 00:53:00:11:22:33   
set firewall ipv4 input filter rule 101 source mac-address !00:53:00:aa:12:34  
 set firewall ipv4 forward filter rule <1-999999> source port [1-65535 | portname | start-end] set firewall ipv4 input filter rule <1-999999> source port [1-65535 | portname | start-end] set firewall ipv4 output filter rule <1-999999> source port [1-65535 | portname | start-end] set firewall ipv4 name <name> rule <1-999999> source port [1-65535 | portname | start-end] set firewall ipv4 forward filter rule <1-999999> destination port [1-65535 | portname | start-end]  
 set firewall ipv4 input filter rule <1-999999> destination port [1-65535 | portname | start-end] set firewall ipv4 output filter rule <1-999999> destination port [1-65535 | portname | start-end]  
 set firewall ipv4 name <name> rule <1-999999> destination port [1-65535 | portname | start-end]   
A port can be set with a port number or a name which is here defined: /etc/services.

set firewall ipv4 forward filter rule 10 source port '22'

set firewall ipv4 forward filter rule 11 source port '!http'   
set firewall ipv4 forward filter rule 12 source port 'https'   
Multiple source ports can be specified as a comma-separated list. The whole list can also be “negated” using !. For example:  
 set firewall ipv4 forward filter rule <1-999999> source group address-group <name | !name> set firewall ipv4 input filter rule <1-999999> source group address-group <name | !name> set firewall ipv4 output filter rule <1-999999> source group address-group <name | !name> set firewall ipv4 name <name> rule <1-999999> source group address-group <name | !name> set firewall ipv4 forward filter rule <1-999999> destination group address-group <name | !

name>  
 set firewall ipv4 input filter rule <1-999999> destination group address-group <name | !name> set firewall ipv4 output filter rule <1-999999> destination group address-group <name | !

name>  
 set firewall ipv4 name <name> rule <1-999999> destination group address-group <name | !

name>   
Use a specific address-group. Prepend character ! for inverted matching criteria.

set firewall ipv4 forward filter rule <1-999999> source group network-group <name | !name> set firewall ipv4 input filter rule <1-999999> source group network-group <name | !name> set firewall ipv4 output filter rule <1-999999> source group network-group <name | !name> set firewall ipv4 name <name> rule <1-999999> source group network-group <name | !name> set firewall ipv4 forward filter rule <1-999999> destination group network-group <name | !

name>  
 set firewall ipv4 input filter rule <1-999999> destination group network-group <name | !name> set firewall ipv4 output filter rule <1-999999> destination group network-group <name | !

name>  
 set firewall ipv4 name <name> rule <1-999999> destination group network-group <name | !

name>   
Use a specific network-group. Prepend character ! for inverted matching criteria.

set firewall ipv4 forward filter rule <1-999999> source group port-group <name | !name> set firewall ipv4 input filter rule <1-999999> source group port-group <name | !name> set firewall ipv4 output filter rule <1-999999> source group port-group <name | !name> set firewall ipv4 name <name> rule <1-999999> source group port-group <name | !name> set firewall ipv4 forward filter rule <1-999999> destination group port-group <name | !name> set firewall ipv4 input filter rule <1-999999> destination group port-group <name | !name> set firewall ipv4 output filter rule <1-999999> destination group port-group <name | !name> set firewall ipv4 name <name> rule <1-999999> destination group port-group <name | !name> Use a specific port-group. Prepend character ! for inverted matching criteria.

set firewall ipv4 forward filter rule <1-999999> source group domain-group <name | !name> set firewall ipv4 input filter rule <1-999999> source group domain-group <name | !name> set firewall ipv4 output filter rule <1-999999> source group domain-group <name | !name> set firewall ipv4 name <name> rule <1-999999> source group domain-group <name | !name> set firewall ipv4 forward filter rule <1-999999> destination group domain-group <name | !

name>  
 set firewall ipv4 input filter rule <1-999999> destination group domain-group <name | !name>

set firewall ipv4 output filter rule <1-999999> destination group domain-group <name | !

name>  
 set firewall ipv4 name <name> rule <1-999999> destination group domain-group <name | !

name>   
Use a specific domain-group. Prepend character ! for inverted matching criteria.

set firewall ipv4 forward filter rule <1-999999> source group mac-group <name | !name> set firewall ipv4 input filter rule <1-999999> source group mac-group <name | !name> set firewall ipv4 output filter rule <1-999999> source group mac-group <name | !name> set firewall ipv4 name <name> rule <1-999999> source group mac-group <name | !name> set firewall ipv4 forward filter rule <1-999999> destination group mac-group <name | !name> set firewall ipv4 input filter rule <1-999999> destination group mac-group <name | !name> set firewall ipv4 output filter rule <1-999999> destination group mac-group <name | !name> set firewall ipv4 name <name> rule <1-999999> destination group mac-group <name | !name> Use a specific mac-group. Prepend character ! for inverted matching criteria.

set firewall ipv4 forward filter rule <1-999999> dscp [0-63 | start-end]  
 set firewall ipv4 input filter rule <1-999999> dscp [0-63 | start-end]  
 set firewall ipv4 output filter rule <1-999999> dscp [0-63 | start-end]  
 set firewall ipv4 name <name> rule <1-999999> dscp [0-63 | start-end]  
 set firewall ipv4 forward filter rule <1-999999> dscp-exclude [0-63 | start-end] set firewall ipv4 input filter rule <1-999999> dscp-exclude [0-63 | start-end] set firewall ipv4 output filter rule <1-999999> dscp-exclude [0-63 | start-end] set firewall ipv4 name <name> rule <1-999999> dscp-exclude [0-63 | start-end] Match based on dscp value.

set firewall ipv4 forward filter rule <1-999999> fragment [match-frag | match-non-frag] set firewall ipv4 input filter rule <1-999999> fragment [match-frag | match-non-frag] set firewall ipv4 output filter rule <1-999999> fragment [match-frag | match-non-frag] set firewall ipv4 name <name> rule <1-999999> fragment [match-frag | match-non-frag] Match based on fragment criteria.

set firewall ipv4 forward filter rule <1-999999> icmp [code | type] <0-255> set firewall ipv4 input filter rule <1-999999> icmp [code | type] <0-255> set firewall ipv4 output filter rule <1-999999> icmp [code | type] <0-255> set firewall ipv4 name <name> rule <1-999999> icmp [code | type] <0-255> Match based on icmp code and type.

set firewall ipv4 forward filter rule <1-999999> icmp type-name <text>  
 set firewall ipv4 input filter rule <1-999999> icmp type-name <text>  
 set firewall ipv4 output filter rule <1-999999> icmp type-name <text>  
 set firewall ipv4 name <name> rule <1-999999> icmp type-name <text>   
Match based on icmp type-name criteria. Use tab for information about what type-name criteria are supported.

set firewall ipv4 forward filter rule <1-999999> inbound-interface name <iface>  
 set firewall ipv4 input filter rule <1-999999> inbound-interface name <iface>  
 set firewall ipv4 name <name> rule <1-999999> inbound-interface name <iface>   
Match based on inbound interface. Wilcard \* can be used. For example: eth2\*. Prepending character ! for inverted matching criteria is also supportd. For example !eth2

set firewall ipv4 forward filter rule <1-999999> inbound-interface group <iface\_group> set firewall ipv4 input filter rule <1-999999> inbound-interface group <iface\_group>  
 set firewall ipv4 name <name> rule <1-999999> inbound-interface group <iface\_group> Match based on inbound interface group. Prepending character ! for inverted matching criteria is also supportd. For example !IFACE\_GROUP  
 set firewall ipv4 forward filter rule <1-999999> outbound-interface name <iface>  
 set firewall ipv4 output filter rule <1-999999> outbound-interface name <iface>  
 set firewall ipv4 name <name> rule <1-999999> outbound-interface name <iface>   
Match based on outbound interface. Wilcard \* can be used. For example: eth2\*. Prepending character ! for inverted matching criteria is also supportd. For example !eth2  
 set firewall ipv4 forward filter rule <1-999999> outbound-interface group <iface\_group> set firewall ipv4 output filter rule <1-999999> outbound-interface group <iface\_group> set firewall ipv4 name <name> rule <1-999999> outbound-interface group <iface\_group> Match based on outbound interface group. Prepending character ! for inverted matching criteria is also supportd. For example !IFACE\_GROUP  
 set firewall ipv4 forward filter rule <1-999999> ipsec [match-ipsec | match-none]  
 set firewall ipv4 input filter rule <1-999999> ipsec [match-ipsec | match-none]  
 set firewall ipv4 output filter rule <1-999999> ipsec [match-ipsec | match-none]  
 set firewall ipv4 name <name> rule <1-999999> ipsec [match-ipsec | match-none]   
Match based on ipsec criteria.

set firewall ipv4 forward filter rule <1-999999> limit burst <0-4294967295> set firewall ipv4 input filter rule <1-999999> limit burst <0-4294967295> set firewall ipv4 output filter rule <1-999999> limit burst <0-4294967295> set firewall ipv4 name <name> rule <1-999999> limit burst <0-4294967295> Match based on the maximum number of packets to allow in excess of rate.

set firewall ipv4 forward filter rule <1-999999> limit rate <text>  
 set firewall ipv4 input filter rule <1-999999> limit rate <text>  
 set firewall ipv4 output filter rule <1-999999> limit rate <text>  
 set firewall ipv4 name <name> rule <1-999999> limit rate <text>   
Match based on the maximum average rate, specified as integer/unit. For example 5/minutes set firewall ipv4 forward filter rule <1-999999> packet-length <text>  
 set firewall ipv4 input filter rule <1-999999> packet-length <text>  
 set firewall ipv4 output filter rule <1-999999> packet-length <text>  
 set firewall ipv4 name <name> rule <1-999999> packet-length <text>  
 set firewall ipv4 forward filter rule <1-999999> packet-length-exclude <text>  
 set firewall ipv4 input filter rule <1-999999> packet-length-exclude <text>  
 set firewall ipv4 output filter rule <1-999999> packet-length-exclude <text>  
 set firewall ipv4 name <name> rule <1-999999> packet-length-exclude <text>   
Match based on packet length criteria. Multiple values from 1 to 65535 and ranges are supported.

set firewall ipv4 forward filter rule <1-999999> packet-type [broadcast | host | multicast | other]  
 set firewall ipv4 input filter rule <1-999999> packet-type [broadcast | host | multicast | other] set firewall ipv4 output filter rule <1-999999> packet-type [broadcast | host | multicast | other]

set firewall ipv4 name <name> rule <1-999999> packet-type [broadcast | host | multicast | other]   
Match based on packet type criteria.

set firewall ipv4 forward filter rule <1-999999> protocol [<text> | <0-255> | all | tcp\_udp] set firewall ipv4 input filter rule <1-999999> protocol [<text> | <0-255> | all | tcp\_udp] set firewall ipv4 output filter rule <1-999999> protocol [<text> | <0-255> | all | tcp\_udp] set firewall ipv4 name <name> rule <1-999999> protocol [<text> | <0-255> | all | tcp\_udp] Match a protocol criteria. A protocol number or a name which is here defined: /etc/protocols. Special names are all for all protocols and tcp\_udp for tcp and udp based packets. The ! negate the selected protocol.

set firewall ipv4 forward fitler rule 10 protocol tcp\_udp   
set firewall ipv4 forward fitler rule 11 protocol !tcp\_udp  
 set firewall ipv4 forward filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 input filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 output filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 name <name> rule <1-999999> recent count <1-255>  
 set firewall ipv4 forward filter rule <1-999999> recent time [second | minute | hour] set firewall ipv4 input filter rule <1-999999> recent time [second | minute | hour] set firewall ipv4 output filter rule <1-999999> recent time [second | minute | hour] set firewall ipv4 name <name> rule <1-999999> recent time [second | minute | hour] Match bases on recently seen sources.

set firewall ipv4 forward filter rule <1-999999> tcp flags [not] <text>  
 set firewall ipv4 input filter rule <1-999999> tcp flags [not] <text>  
 set firewall ipv4 output filter rule <1-999999> tcp flags [not] <text>  
 set firewall ipv4 name <name> rule <1-999999> tcp flags [not] <text>   
Allowed values fpr TCP flags: ack, cwr, ecn, fin, psh, rst, syn and urg. Multiple values are supported, and for inverted selection use not, as shown in the example.

set firewall ipv4 input filter rule 10 tcp flags 'ack'   
set firewall ipv4 input filter rule 12 tcp flags 'syn'   
set firewall ipv4 input filter rule 13 tcp flags not 'fin'  
 set firewall ipv4 forward filter rule <1-999999> state [established | invalid | new | related] [enable | disable]  
 set firewall ipv4 input filter rule <1-999999> state [established | invalid | new | related] [enable | disable]  
 set firewall ipv4 output filter rule <1-999999> state [established | invalid | new | related] [enable | disable]  
 set firewall ipv4 name <name> rule <1-999999> state [established | invalid | new | related] [enable | disable]   
Match against the state of a packet.

set firewall ipv4 forward filter rule <1-999999> time startdate <text> set firewall ipv4 input filter rule <1-999999> time startdate <text> set firewall ipv4 output filter rule <1-999999> time startdate <text> set firewall ipv4 name <name> rule <1-999999> time startdate <text> set firewall ipv4 forward filter rule <1-999999> time starttime <text>

set firewall ipv4 input filter rule <1-999999> time starttime <text> set firewall ipv4 output filter rule <1-999999> time starttime <text> set firewall ipv4 name <name> rule <1-999999> time starttime <text> set firewall ipv4 forward filter rule <1-999999> time stopdate <text> set firewall ipv4 input filter rule <1-999999> time stopdate <text> set firewall ipv4 output filter rule <1-999999> time stopdate <text> set firewall ipv4 name <name> rule <1-999999> time stopdate <text> set firewall ipv4 forward filter rule <1-999999> time stoptime <text> set firewall ipv4 input filter rule <1-999999> time stoptime <text> set firewall ipv4 output filter rule <1-999999> time stoptime <text> set firewall ipv4 name <name> rule <1-999999> time stoptime <text> set firewall ipv4 forward filter rule <1-999999> time weekdays <text> set firewall ipv4 input filter rule <1-999999> time weekdays <text> set firewall ipv4 output filter rule <1-999999> time weekdays <text> set firewall ipv4 name <name> rule <1-999999> time weekdays <text> Time to match the defined rule.

set firewall ipv4 forward filter rule <1-999999> ttl <eq | gt | lt> <0-255>  
 set firewall ipv4 input filter rule <1-999999> ttl <eq | gt | lt> <0-255>  
 set firewall ipv4 output filter rule <1-999999> ttl <eq | gt | lt> <0-255>  
 set firewall ipv4 name <name> rule <1-999999> ttl <eq | gt | lt> <0-255>   
Match time to live parameter, where ‘eq’ stands for ‘equal’; ‘gt’ stands for ‘greater than’, and ‘lt’ stands for ‘less than’.

set firewall ipv4 forward filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 input filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 output filter rule <1-999999> recent count <1-255>  
 set firewall ipv4 name <name> rule <1-999999> recent count <1-255>  
 set firewall ipv4 forward filter rule <1-999999> recent time <second | minute | hour> set firewall ipv4 input filter rule <1-999999> recent time <second | minute | hour>  
 set firewall ipv4 output filter rule <1-999999> recent time <second | minute | hour>  
 set firewall ipv4 name <name> rule <1-999999> recent time <second | minute | hour> Match when ‘count’ amount of connections are seen within ‘time’. These matching criteria can be used to block brute-force attempts.

Synproxy  
Synproxy connections  
 set firewall ipv4 [input | forward] filter rule <1-999999> action synproxy  
 set firewall ipv4 [input | forward] filter rule <1-999999> protocol tcp  
 set firewall ipv4 [input | forward] filter rule <1-999999> synproxy tcp mss <501-65535> Set TCP-MSS (maximum segment size) for the connection  
 set firewall ipv4 [input | forward] filter rule <1-999999> synproxy tcp window-scale <1-14> Set the window scale factor for TCP window scaling   
Example synproxy  
Requirements to enable synproxy:  
• Traffic must be symmetric  
• Synproxy relies on syncookies and TCP timestamps, ensure these are enabled

• Disable conntrack loose track option   
set system sysctl parameter net.ipv4.tcp\_timestamps value '1'

set system conntrack tcp loose disable   
set system conntrack ignore ipv4 rule 10 destination port '8080' set system conntrack ignore ipv4 rule 10 protocol 'tcp'   
set system conntrack ignore ipv4 rule 10 tcp flags syn

set firewall global-options syn-cookies 'enable'   
set firewall ipv4 input filter rule 10 action 'synproxy'   
set firewall ipv4 input filter rule 10 destination port '8080'   
set firewall ipv4 input filter rule 10 inbound-interface interface-name 'eth1' set firewall ipv4 input filter rule 10 protocol 'tcp'   
set firewall ipv4 input filter rule 10 synproxy tcp mss '1460'   
set firewall ipv4 input filter rule 10 synproxy tcp window-scale '7'   
set firewall ipv4 input filter rule 1000 action 'drop'   
set firewall ipv4 input filter rule 1000 state invalid 'enable'   
Operation-mode Firewall  
Rule-set overview  
 show firewall   
This will show you a basic firewall overview, for all ruleset, and not only for ipv4 vyos@vyos:~$ show firewall   
Rulesets Information

---------------------------------  
ipv4 Firewall "forward filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- -----------------------------  
20 accept all 0 0 ip saddr @N\_TRUSTEDv4 accept 21 jump all 0 0 jump NAME\_AUX   
default accept all 0 0

---------------------------------  
ipv4 Firewall "input filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- -------------------------  
10 accept all 156 14377 iifname != @I\_LAN accept default accept all 0 0

---------------------------------  
ipv4 Firewall "name AUX"

Rule Action Protocol Packets Bytes Conditions  
------ -------- ---------- --------- ------- --------------------------------------------  
 10 accept icmp 0 0 meta l4proto icmp accept  
 20 accept udp 0 0 meta l4proto udp ip saddr @A\_SERVERS accept 30 drop all 0 0 ip saddr != @A\_SERVERS iifname "eth2"

---------------------------------  
ipv4 Firewall "output filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- ----------------------------------------  
10 reject all 0 0 oifname @I\_LAN   
20 accept icmp 2 168 meta l4proto icmp oifname "eth0" accept default accept all 72 9258

---------------------------------  
ipv6 Firewall "input filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- -------------------------------  
10 accept all 0 0 ip6 saddr @N6\_TRUSTEDv6 accept default accept all 2 112

vyos@vyos:~$  
 show firewall summary   
This will show you a summary of rule-sets and groups vyos@vyos:~$ show firewall summary   
Ruleset Summary

IPv6 Ruleset:

Ruleset Hook Ruleset Priority Description  
-------------- -------------------- -------------------------  
forward filter   
input filter   
ipv6\_name IPV6-VyOS\_MANAGEMENT   
ipv6\_name IPV6-WAN\_IN PUBLIC\_INTERNET

IPv4 Ruleset:

Ruleset Hook Ruleset Priority Description  
-------------- ------------------ -------------------------  
forward filter   
input filter

name VyOS\_MANAGEMENT   
name WAN\_IN PUBLIC\_INTERNET

Firewall Groups

Name Type References Members  
----------------------- ------------------ ----------------------- ----------------  
PBX address\_group WAN\_IN-100 198.51.100.77   
SERVERS address\_group WAN\_IN-110 192.0.2.10  
 WAN\_IN-111 192.0.2.11  
 WAN\_IN-112 192.0.2.12  
 WAN\_IN-120  
 WAN\_IN-121  
 WAN\_IN-122   
SUPPORT address\_group VyOS\_MANAGEMENT-20 192.168.1.2 WAN\_IN-20   
PHONE\_VPN\_SERVERS address\_group WAN\_IN-160 10.6.32.2 PINGABLE\_ADRESSES address\_group WAN\_IN-170 192.168.5.2 WAN\_IN-171   
PBX ipv6\_address\_group IPV6-WAN\_IN-100 2001:db8::1   
SERVERS ipv6\_address\_group IPV6-WAN\_IN-110 2001:db8::2 IPV6-WAN\_IN-111 2001:db8::3  
 IPV6-WAN\_IN-112 2001:db8::4  
 IPV6-WAN\_IN-120  
 IPV6-WAN\_IN-121  
 IPV6-WAN\_IN-122   
SUPPORT ipv6\_address\_group IPV6-VyOS\_MANAGEMENT-20 2001:db8::5 IPV6-WAN\_IN-20  
 show firewall ipv4 [forward | input | output] filter  
 show firewall ipv4 name <name>   
This command will give an overview of a single rule-set.

vyos@vyos:~$ show firewall ipv4 input filter   
Ruleset Information

---------------------------------  
IPv4 Firewall "input filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- -----------------------------------------  
5 jump all 0 0 iifname "eth2" jump NAME\_VyOS\_MANAGEMENT default accept all  
 show firewall ipv4 [forward | input | output] filter rule <1-999999>  
 show firewall ipv4 name <name> rule <1-999999>

This command will give an overview of a rule in a single rule-set, plus information for default action.

vyos@vyos:~$show firewall ipv4 output filter rule 20 Rule Information

---------------------------------  
ipv4 Firewall "output filter"

Rule Action Protocol Packets Bytes Conditions  
------- -------- ---------- --------- ------- ----------------------------------------  
20 accept icmp 2 168 meta l4proto icmp oifname "eth0" accept default accept all 286 47614

vyos@vyos:~$  
 show firewall statistics   
This will show you a statistic of all rule-sets since the last boot.

Show Firewall log  
 show log firewall  
 show log firewall ipv4  
 show log firewall ipv4 [forward | input | output | name]  
 show log firewall ipv4 [forward | input | output] filter  
 show log firewall ipv4 name <name>  
 show log firewall ipv4 [forward | input | output] filter rule <rule>  
 show log firewall ipv4 name <name> rule <rule>   
Show the logs of all firewall; show all ipv4 firewall logs; show all logs for particular hook; show all logs for particular hook and priority; show all logs for particular custom chain; show logs for specific Rule-Set.

Example Partial Config  
firewall {  
 group {  
 network-group BAD-NETWORKS {  
 network 198.51.100.0/24  
 network 203.0.113.0/24  
 }  
 network-group GOOD-NETWORKS {  
 network 192.0.2.0/24  
 }  
 port-group BAD-PORTS {  
 port 65535  
 }  
 }  
 ipv4 {  
 forward {  
 filter {

default-action accept  
 rule 5 {  
 action accept  
 source {  
 group {  
 network-group GOOD-NETWORKS  
 }  
 }  
 }  
 rule 10 {  
 action drop  
 description "Bad Networks"  
 protocol all  
 source {  
 group {  
 network-group BAD-NETWORKS  
 }  
 }  
 }  
 }  
 }  
 }   
}   
Update geoip database  
 update geoip   
Command used to update GeoIP database and firewall sets.

Flowtables Firewall Configuration  
Note

Documentation under development

Overview  
In this section there’s useful information of all firewall configuration that can be done regarding flowtables

set firewall flowtables …  
From main structure defined in Firewall Overview in this section you can find detailed information only for the next part of the general structure:

- set firewall  
 \* flowtable  
 - custom\_flow\_table  
 + ...

Flowtables allows you to define a fastpath through the flowtable datapath. The flowtable supports for the layer 3 IPv4 and IPv6 and the layer 4 TCP and UDP protocols.

../../\_images/firewall-flowtable-packet-flow.png   
Once the first packet of the flow successfully goes through the IP forwarding path (black circles path), from the second packet on, you might decide to offload the flow to the flowtable through your ruleset. The flowtable infrastructure provides a rule action that allows you to specify when to add a flow to the flowtable (On forward filtering, red circle number 6)

A packet that finds a matching entry in the flowtable (flowtable hit) is transmitted to the output netdevice, hence, packets bypass the classic IP forwarding path and uses the Fast Path (orange circles path). The visible effect is that you do not see these packets from any of the Netfilter hooks coming after ingress. In case that there is no matching entry in the flowtable (flowtable miss), the packet follows the classic IP forwarding path.

Zone Based Firewall  
Overview  
Note

Starting from VyOS 1.4-rolling-202308040557, a new firewall structure can be found on all vyos instalations. Zone based firewall was removed in that version, but re introduced in VyOS 1.4 and 1.5. All versions built after 2023-10-22 has this feature. Documentation for most of the new firewall CLI can be found in the firewall chapter. The legacy firewall is still available for versions before 1.4-rolling-202308040557 and can be found in the legacy firewall configuration chapter.

In this section there’s useful information of all firewall configuration that is needed for zone-based firewall. Configuration commands covered in this section:

set firewall zone …  
From main structure defined in Firewall Overview in this section you can find detailed information only for the next part of the general structure:

- set firewall  
 \* zone  
 - custom\_zone\_name  
 + ...

In zone-based policy, interfaces are assigned to zones, and inspection policy is applied to traffic moving between the zones and acted on according to firewall rules. A zone is a group of interfaces that have similar functions or features. It establishes the security borders of a network. A zone defines a boundary where traffic is subjected to policy restrictions as it crosses to another region of a network.

Key Points:

A zone must be configured before an interface is assigned to it and an interface can be assigned to only a single zone.

All traffic to and from an interface within a zone is permitted.

All traffic between zones is affected by existing policies   
Traffic cannot flow between zone member interface and any interface that is not a zone member.

You need 2 separate firewalls to define traffic: one for each direction.

Note   
In T2199 the syntax of the zone configuration was changed. The zone configuration moved from zone-policy zone <name> to firewall zone <name>.

Configuration  
As an alternative to applying policy to an interface directly, a zone-based firewall can be created to simplify configuration when multiple interfaces belong to the same security zone. Instead of applying rule-sets to interfaces, they are applied to source zone-destination zone pairs. A basic introduction to zone-based firewalls can be found here, and an example at Zone-Policy example.

Define a Zone  
To define a zone setup either one with interfaces or a local zone.

set firewall zone <name> interface <interface>   
Set interfaces to a zone. A zone can have multiple interfaces. But an interface can only be a member in one zone.

set firewall zone <name> local-zone   
Define the zone as a local zone. A local zone has no interfaces and will be applied to the router itself.

set firewall zone <name> default-action [drop | reject] Change the default-action with this setting.

set firewall zone <name> description   
Set a meaningful description.

Applying a Rule-Set to a Zone  
Before you are able to apply a rule-set to a zone you have to create the zones first.

It helps to think of the syntax as: (see below). The ‘rule-set’ should be written from the perspective of: Source Zone-to->\*Destination Zone\*

set firewall zone <Destination Zone> from <Source Zone> firewall name <rule-set>  
 set firewall zone <name> from <name> firewall name <rule-set>  
 set firewall zone <name> from <name> firewall ipv6-name <rule-set>   
You apply a rule-set always to a zone from an other zone, it is recommended to create one rule-set for each zone pair.

set firewall zone DMZ from LAN firewall name LANv4-to-DMZv4 set firewall zone LAN from DMZ firewall name DMZv4-to-LANv4

High availability  
VRRP (Virtual Router Redundancy Protocol) provides active/backup redundancy for routers. Every VRRP router has a physical IP/IPv6 address, and a virtual address. On startup, routers elect the master, and the router with the highest priority becomes the master and assigns the virtual address to its interface. All routers with lower priorities become backup routers. The master then starts sending keepalive packets to notify other routers that it’s available. If the master fails and stops sending keepalive packets, the router with the next highest priority becomes the new master and takes over the virtual address.

VRRP keepalive packets use multicast, and VRRP setups are limited to a single datalink layer segment. You can setup multiple VRRP groups (also called virtual routers). Virtual routers are identified by a VRID (Virtual Router IDentifier). If you setup multiple groups on the same interface, their VRIDs must be unique if they use the same address family, but it’s possible (even if not recommended for readability reasons) to use duplicate VRIDs on different interfaces.

Basic setup  
VRRP groups are created with the set high-availability vrrp group $GROUP\_NAME commands. The required parameters are interface, vrid, and address.

minimal config

set high-availability vrrp group Foo vrid 10   
set high-availability vrrp group Foo interface eth0   
set high-availability vrrp group Foo address 192.0.2.1/24   
You can verify your VRRP group status with the operational mode run show vrrp command:

vyos@vyos# run show vrrp   
Name Interface VRID State Last Transition---------- ----------- ------ ------- -----------------  
Foo eth1 10 MASTER 2s   
IPv6 support

The address parameter can be either an IPv4 or IPv6 address, but you can not mix IPv4 and IPv6 in the same group, and will need to create groups with different VRIDs specially for IPv4 and IPv6. If you want to use IPv4 + IPv6 address you can use option excluded-address

Address  
The address can be configured either on the VRRP interface or on not VRRP interface.

set high-availability vrrp group Foo address 192.0.2.1/24   
set high-availability vrrp group Foo address 203.0.113.22/24 interface eth2 set high-availability vrrp group Foo address 198.51.100.33/24 interface eth3 Disabling a VRRP group  
You can disable a VRRP group with disable option:

set high-availability vrrp group Foo disable   
A disabled group will be removed from the VRRP process and your router will not participate in VRRP for that VRID. It will disappear from operational mode commands output, rather than enter the backup state.

Exclude address  
Exclude IP addresses from VRRP packets. This option excluded-address is used when you want to set IPv4 + IPv6 addresses on the same virtual interface or when used more than 20 IP addresses.

set high-availability vrrp group Foo excluded-address '203.0.113.254/24' set high-availability vrrp group Foo excluded-address '2001:db8:aa::1/64' set high-availability vrrp group Foo excluded-address '2001:db8:22::1/64' Setting VRRP group priority  
VRRP priority can be set with priority option:

set high-availability vrrp group Foo priority 200   
The priority must be an integer number from 1 to 255. Higher priority value increases router’s precedence in the master elections.

Sync groups  
A sync group allows VRRP groups to transition together.

edit high-availability vrrp   
set sync-group MAIN member VLAN9   
set sync-group MAIN member VLAN20   
In the following example, when VLAN9 transitions, VLAN20 will also transition:

vrrp {  
 group VLAN9 {  
 interface eth0.9

address 10.9.1.1/24  
 priority 200  
 vrid 9  
 }  
 group VLAN20 {  
 interface eth0.20  
 priority 200  
 address 10.20.20.1/24  
 vrid 20  
 }  
 sync-group MAIN {  
 member VLAN20  
 member VLAN9  
 }   
}   
Warning

All items in a sync group should be similarly configured. If one VRRP group is set to a different preemption delay or priority, it would result in an endless transition loop.

Preemption  
VRRP can use two modes: preemptive and non-preemptive. In the preemptive mode, if a router with a higher priority fails and then comes back, routers with lower priority will give up their master status. In non-preemptive mode, the newly elected master will keep the master status and the virtual address indefinitely.

By default VRRP uses preemption. You can disable it with the “no-preempt” option:

set high-availability vrrp group Foo no-preempt   
You can also configure the time interval for preemption with the “preempt-delay” option. For example, to set the higher priority router to take over in 180 seconds, use:

set high-availability vrrp group Foo preempt-delay 180   
Track  
Track option to track non VRRP interface states. VRRP changes status to FAULT if one of the track interfaces in state down.

set high-availability vrrp group Foo track interface eth0 set high-availability vrrp group Foo track interface eth1 Ignore VRRP main interface faults

set high-availability vrrp group Foo track exclude-vrrp-interface Unicast VRRP

By default VRRP uses multicast packets. If your network does not support multicast for whatever reason, you can make VRRP use unicast communication instead.

set high-availability vrrp group Foo peer-address 192.0.2.10   
set high-availability vrrp group Foo hello-source-address 192.0.2.15   
rfc3768-compatibility  
RFC 3768 defines a virtual MAC address to each VRRP virtual router. This virtual router MAC address will be used as the source in all periodic VRRP messages sent by the active node. When the rfc3768-compatibility option is set, a new VRRP interface is created, to which the MAC address and the virtual IP address is automatically assigned.

set high-availability vrrp group Foo rfc3768-compatibility Verification

$show interfaces ethernet eth0v10   
eth0v10@eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000   
link/ether 00:00:5e:00:01:0a brd ff:ff:ff:ff:ff:ff   
inet 172.25.0.247/16 scope global eth0v10   
valid\_lft forever preferred\_lft forever   
Global options  
On most scenarios, there’s no need to change specific parameters, and using default configuration is enough. But there are cases were extra configuration is needed.

set high-availability vrrp global-parameters startup\_delay <1-600>   
This option specifies a delay in seconds before vrrp instances start up after keepalived starts.

Gratuitous ARP  
These configuration is not mandatory and in most cases there’s no need to configure it. But if necessary, Gratuitous ARP can be configured in global-parameters and/or in group section.

set high-availability vrrp global-parameters garp interval <0.000-1000> set high-availability vrrp group <name> garp interval <0.000-1000> Set delay between gratuitous ARP messages sent on an interface.

0 if not defined.

set high-availability vrrp global-parameters garp master-delay <1-255> set high-availability vrrp group <name> garp master-delay <1-255> Set delay for second set of gratuitous ARPs after transition to MASTER.

5 if not defined.

set high-availability vrrp global-parameters garp master-refresh <1-600>

set high-availability vrrp group <name> garp master-refresh <1-600> Set minimum time interval for refreshing gratuitous ARPs while MASTER.

0 if not defined, which means no refreshing.

set high-availability vrrp global-parameters garp master-refresh-repeat <1-600> set high-availability vrrp group <name> garp master-refresh-repeat <1-600> Set number of gratuitous ARP messages to send at a time while MASTER.

1 if not defined.

set high-availability vrrp global-parameters garp master-repeat <1-600>  
 set high-availability vrrp group <name> garp master-repeat <1-600>   
Set number of gratuitous ARP messages to send at a time after transition to MASTER.

5 if not defined.

Version  
 set high-availability vrrp global-parameters version 2|3   
Set the default VRRP version to use. This defaults to 2, but IPv6 instances will always use version 3.

Scripting  
VRRP functionality can be extended with scripts. VyOS supports two kinds of scripts: health check scripts and transition scripts. Health check scripts execute custom checks in addition to the master router reachability. Transition scripts are executed when VRRP state changes from master to backup or fault and vice versa and can be used to enable or disable certain services, for example.

Health check scripts  
This setup will make the VRRP process execute the /config/scripts/vrrp-check.sh script every 60 seconds, and transition the group to the fault state if it fails (i.e. exits with non-zero status) three times:

set high-availability vrrp group Foo health-check script /config/scripts/vrrp-check.sh   
set high-availability vrrp group Foo health-check interval 60   
set high-availability vrrp group Foo health-check failure-count 3   
Transition scripts  
Transition scripts can help you implement various fixups, such as starting and stopping services, or even modifying the VyOS config on VRRP transition. This setup will make the VRRP process execute the /config/scripts/vrrp-fail.sh with argument Foo when VRRP fails, and the   
/config/scripts/vrrp-master.sh when the router becomes the master:

set high-availability vrrp group Foo transition-script backup "/config/scripts/vrrp-fail.sh Foo"

set high-availability vrrp group Foo transition-script fault "/config/scripts/vrrp-fail.sh Foo" set high-availability vrrp group Foo transition-script master "/config/scripts/vrrp-master.sh Foo" To know more about scripting, check the Command Scripting section.

Virtual-server  
Call for Contributions   
This section needs improvements, examples and explanations.

Please take a look at the Contributing Guide for our Write Documentation.

Virtual Server allows to Load-balance traffic destination virtual-address:port between several real servers.

Algorithm  
Load-balancing schedule algorithm:   
round-robin   
weighted-round-robin   
least-connection   
weighted-least-connection   
source-hashing   
destination-hashing   
locality-based-least-connection   
set high-availability virtual-server 203.0.113.1 algorithm 'least-connection' Forward method  
NAT   
direct   
tunnel   
set high-availability virtual-server 203.0.113.1 forward-method 'nat' Health-check  
Custom health-check script allows checking real-server availability

set high-availability virtual-server 203.0.113.1 real-server 192.0.2.11 health-check script <path-to-script>   
Fwmark  
Firewall mark. It possible to loadbalancing traffic based on fwmark value

set high-availability virtual-server 203.0.113.1 fwmark '111' Real server  
Real server IP address and port

set high-availability virtual-server 203.0.113.1 real-server 192.0.2.11 port '80' Example  
Virtual-server can be configured with VRRP virtual address or without VRRP.

In the next example all traffic destined to 203.0.113.1 and port 8280 protocol TCP is balanced between 2 real servers 192.0.2.11 and 192.0.2.12 to port 80

Real server is auto-excluded if port check with this server fail.

set interfaces ethernet eth0 address '203.0.113.11/24'   
set interfaces ethernet eth1 address '192.0.2.1/24'   
set high-availability vrrp group FOO interface 'eth0'   
set high-availability vrrp group FOO no-preempt   
set high-availability vrrp group FOO priority '150'   
set high-availability vrrp group FOO address '203.0.113.1/24' set high-availability vrrp group FOO vrid '10'

set high-availability virtual-server 203.0.113.1 algorithm 'source-hashing'   
set high-availability virtual-server 203.0.113.1 delay-loop '10'   
set high-availability virtual-server 203.0.113.1 forward-method 'nat'   
set high-availability virtual-server 203.0.113.1 persistence-timeout '180'   
set high-availability virtual-server 203.0.113.1 port '8280'   
set high-availability virtual-server 203.0.113.1 protocol 'tcp'   
set high-availability virtual-server 203.0.113.1 real-server 192.0.2.11 port '80'   
set high-availability virtual-server 203.0.113.1 real-server 192.0.2.12 port '80'   
A firewall mark fwmark allows using multiple ports for high-availability virtual-server. It uses fwmark value.

In this example all traffic destined to ports “80, 2222, 8888” protocol TCP marks to fwmark “111” and balanced between 2 real servers. Port “0” is required if multiple ports are used.

set interfaces ethernet eth0 address 'dhcp'   
set interfaces ethernet eth0 description 'WAN' set interfaces ethernet eth1 address '192.0.2.1/24' set interfaces ethernet eth1 description 'LAN'

set policy route PR interface 'eth0'   
set policy route PR rule 10 destination port '80,2222,8888' set policy route PR rule 10 protocol 'tcp'   
set policy route PR rule 10 set mark '111'

set high-availability virtual-server vyos fwmark '111'   
set high-availability virtual-server vyos protocol 'tcp'   
set high-availability virtual-server vyos real-server 192.0.2.11 health-check script '/config/scripts/check-real-server-first.sh'   
set high-availability virtual-server vyos real-server 192.0.2.11 port '0'   
set high-availability virtual-server vyos real-server 192.0.2.12 health-check script '/config/scripts/check-real-server-second.sh'   
set high-availability virtual-server vyos real-server 192.0.2.12 port '0'

set nat source rule 100 outbound-interface 'eth0'   
set nat source rule 100 source address '192.0.2.0/24' set nat source rule 100 translation address 'masquerade' Op-mode check virtual-server status

vyos@r14:~$ run show virtual-server   
IP Virtual Server version 1.2.1 (size=4096)   
Prot LocalAddress:Port Scheduler Flags  
 -> RemoteAddress:Port Forward Weight ActiveConn InActConn FWM 111 lc persistent 300  
 -> 192.0.2.11:0 Masq 1 0 0  
 -> 192.0.2.12:0 Masq 1 1 0

WWAN - Wireless Wide-Area-Network  
The Wireless Wide-Area-Network interface provides access (through a wireless modem/wwan) to wireless networks provided by various cellular providers.

VyOS uses the interfaces wwan subsystem for configuration.

Configuration  
Common interface configuration  
 set interfaces wwan <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64

dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces wwan wwan0 address 192.0.2.1/24   
set interfaces wwan wwan0 address 2001:db8::1/64   
set interfaces wwan wwan0 address dhcp   
set interfaces wwan wwan0 address dhcpv6  
 set interfaces wwan <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces wwan wwan0 description 'This is an awesome interface running on VyOS' set interfaces wwan <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces wwan wwan0 disable  
 set interfaces wwan <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces wwan wwan0 disable-link-detect  
 set interfaces wwan <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:

set interfaces wwan wwan0 mtu 1600  
 set interfaces wwan <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wwan <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces wwan wwan0 ip arp-cache-timeout 180  
 set interfaces wwan <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces wwan wwan0 ip disable-arp-filter  
 set interfaces wwan <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces wwan wwan0 ip disable-forwarding  
 set interfaces wwan <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces wwan wwan0 ip enable-directed-broadcast  
 set interfaces wwan <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces wwan wwan0 ip enable-arp-accept  
 set interfaces wwan <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces wwan wwan0 ip enable-arp-announce set interfaces wwan <interface> ip enable-arp-ignore

Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces wwan wwan0 ip enable-arp-ignore  
 set interfaces wwan <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces wwan wwan0 ip enable-proxy-arp  
 set interfaces wwan <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces wwan <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos

attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces wwan <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces wwan wwan0 ipv6 address autoconf  
 set interfaces wwan <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces wwan wwan0 ipv6 address eui64 2001:db8:beef::/64 set interfaces wwan <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces wwan wwan0 ipv6 address no-default-link-local  
 set interfaces wwan <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces wwan wwan0 ipv6 disable-forwarding

set interfaces wwan <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wwan <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces wwan wwan0 ipv6 accept-dad 2  
 set interfaces wwan <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces wwan wwan0 ipv6 dup-addr-detect-transmits 5 set interfaces wwan <interface> vrf <vrf>   
Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces wwan wwan0 vrf red   
DHCP(v6)  
 set interfaces wwan <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces wwan wwan0 dhcp-options client-id 'foo-bar'  
 set interfaces wwan <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces wwan wwan0 dhcp-options host-name 'VyOS'  
 set interfaces wwan <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces wwan wwan0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces wwan <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces wwan wwan0 dhcp-options no-default-route  
 set interfaces wwan <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces wwan wwan0 dhcp-options default-route-distance 220

set interfaces wwan <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24

Example:

set interfaces wwan wwan0 dhcp-options reject 192.168.100.0/24  
 set interfaces wwan <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces wwan wwan0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces wwan <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces wwan wwan0 dhcpv6-options no-release  
 set interfaces wwan <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces wwan wwan0 dhcpv6-options parameters-only  
 set interfaces wwan <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces wwan wwan0 dhcpv6-options rapid-commit  
 set interfaces wwan <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces wwan wwan0 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces wwan <interface> dhcpv6-options pd <id> length <length>

Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces wwan wwan0 dhcpv6-options pd 0 length 56  
 set interfaces wwan <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces wwan wwan0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces wwan <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces wwan wwan0 dhcpv6-options pd 0 interface eth8 sla-id 1   
WirelessModem (WWAN) options  
 set interfaces wwan <interface> apn <apn>   
Every WWAN connection requires an APN which is used by the client to dial into the ISPs network. This is a mandatory parameter. Contact your Service Provider for correct APN.

Operation  
 show interfaces wwan <interface>   
Show detailed information on given <interface>

vyos@vyos:~$ show interfaces wwan wwan0   
wwan0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UNKNOWN group default qlen 1000  
 link/ether 02:c2:f3:00:01:02 brd ff:ff:ff:ff:ff:ff  
 inet 10.155.144.12/30 brd 10.155.144.15 scope global dynamic wwan0

valid\_lft 7012sec preferred\_lft 7012sec  
 inet6 fe80::c2:f3ff:fe00:0102/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 640 2 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions 3229 16 0 0 0 0  
 show interfaces wwan <interface> summary   
Show detailed information summary on given <interface>

vyos@vyos:~$ show interfaces wwan wwan0 summary  
 --------------------------------  
 General | dbus path: /org/freedesktop/ModemManager1/Modem/0  
 | device id: 79f4e9cc2e9fc8d4a3b8c8f6327c2e363170194d  
 --------------------------------  
 Hardware | manufacturer: Sierra Wireless, Incorporated  
 | model: MC7710  
 | revision: SWI9200X\_03.05.29.03ap r6485 CNSHZ-ED-XP0031 2014/12/02 17:53:15  
 | h/w revision: 1.0  
 | supported: gsm-umts, lte  
 | current: gsm-umts, lte  
 | equipment id: 358xxxxxxxxxxxx  
 --------------------------------  
 System | device: /sys/devices/pci0000:00/0000:00:13.0/usb3/3-1/3-1.3 | drivers: qcserial, qmi\_wwan  
 | plugin: Generic  
 | primary port: cdc-wdm0  
 | ports: ttyUSB0 (qcdm), ttyUSB2 (at), cdc-wdm0 (qmi), wwan0 (net) --------------------------------  
 Numbers | own: 4917xxxxxxxx  
 --------------------------------  
 Status | lock: sim-pin2  
 | unlock retries: sim-pin (3), sim-pin2 (3), sim-puk (10), sim-puk2 (10)  
 | state: connected  
 | power state: on  
 | access tech: lte  
 | signal quality: 63% (recent)  
 --------------------------------  
 Modes | supported: allowed: 2g; preferred: none  
 | allowed: 3g; preferred: none  
 | allowed: 4g; preferred: none  
 | allowed: 2g, 3g; preferred: 3g

| allowed: 2g, 3g; preferred: 2g  
 | allowed: 2g, 4g; preferred: 4g  
 | allowed: 2g, 4g; preferred: 2g  
 | allowed: 3g, 4g; preferred: 3g  
 | allowed: 3g, 4g; preferred: 4g  
 | allowed: 2g, 3g, 4g; preferred: 4g  
 | allowed: 2g, 3g, 4g; preferred: 3g  
 | allowed: 2g, 3g, 4g; preferred: 2g  
 | current: allowed: 2g, 3g, 4g; preferred: 2g  
 --------------------------------  
 Bands | supported: egsm, dcs, pcs, utran-1, utran-8, eutran-1, eutran-3, | eutran-7, eutran-8, eutran-20  
 | current: egsm, dcs, pcs, utran-1, utran-8, eutran-1, eutran-3, | eutran-7, eutran-8, eutran-20  
 --------------------------------  
 IP | supported: ipv4, ipv6, ipv4v6  
 --------------------------------  
 3GPP | imei: 358xxxxxxxxxxxx  
 | operator id: 26201  
 | operator name: Telekom.de  
 | registration: home  
 --------------------------------  
 3GPP EPS | ue mode of operation: ps-1  
 --------------------------------  
 SIM | dbus path: /org/freedesktop/ModemManager1/SIM/0  
 --------------------------------  
 Bearer | dbus path: /org/freedesktop/ModemManager1/Bearer/0 show interfaces wwan <interface> capabilities   
Show WWAN module hardware capabilities.

vyos@vyos:~$ show interfaces wwan wwan0 capabilities   
Max TX channel rate: '50000000'   
Max RX channel rate: '100000000'   
Data Service: 'simultaneous-cs-ps'   
SIM: 'supported'   
Networks: 'gsm, umts, lte'   
Bands: 'gsm-dcs-1800, gsm-900-extended, gsm-900-primary, gsm-pcs-1900, wcdma-2100, wcdma-900'   
LTE bands: '1, 3, 7, 8, 20'  
 show interfaces wwan <interface> firmware   
Show WWAN module firmware.

vyos@vyos:~$ show interfaces wwan wwan0 firmware Model: MC7710

Boot version: SWI9200X\_03.05.29.03bt r6485 CNSHZ-ED-XP0031 2014/12/02 17:33:08 AMSS version: SWI9200X\_03.05.29.03ap r6485 CNSHZ-ED-XP0031 2014/12/02 17:53:15 SKU ID: unknown   
Package ID: unknown   
Carrier ID: 0   
Config version: unknown  
 show interfaces wwan <interface> imei   
Show WWAN module IMEI.

vyos@vyos:~$ show interfaces wwan wwan0 imei ESN: '0'   
IMEI: '358xxxxxxxxxxxx'   
MEID: 'unknown'  
 show interfaces wwan <interface> imsi   
Show WWAN module IMSI.

vyos@vyos:~$ show interfaces wwan wwan0 imsi IMSI: '262xxxxxxxxxxxx'  
 show interfaces wwan <interface> model   
Show WWAN module model.

vyos@vyos:~$ show interfaces wwan wwan0 model Model: 'MC7710'  
 show interfaces wwan <interface> msisdn   
Show WWAN module MSISDN.

vyos@vyos:~$ show interfaces wwan wwan0 msisdn MSISDN: '4917xxxxxxxx'  
 show interfaces wwan <interface> revision   
Show WWAN module hardware revision.

vyos@vyos:~$ show interfaces wwan wwan0 revision   
Revision: 'SWI9200X\_03.05.29.03ap r6485 CNSHZ-ED-XP0031 2014/12/02 17:53:15' show interfaces wwan <interface> signal   
Show WWAN module signal strength.

vyos@vyos:~$ show interfaces wwan wwan0 signal LTE:   
RSSI: '-74 dBm'   
RSRQ: '-7 dB'   
RSRP: '-100 dBm'   
SNR: '13.0 dB'   
Radio Interface: 'lte'   
Active Band Class: 'eutran-3'

Active Channel: '1300'  
 show interfaces wwan <interface> sim   
Show WWAN module SIM card information.

vyos@vyos:~$ show interfaces wwan wwan0 sim   
Provisioning applications:   
Primary GW: slot '1', application '1'   
Primary 1X: session doesn't exist   
Secondary GW: session doesn't exist   
Secondary 1X: session doesn't exist   
Slot [1]:   
Card state: 'present'   
UPIN state: 'not-initialized'   
UPIN retries: '0'   
UPUK retries: '0'   
Application [1]:   
Application type: 'usim (2)'   
Application state: 'ready'   
Application ID:   
A0:00:00:00:87:10:02:FF:49:94:20:89:03:10:00:00   
Personalization state: 'ready'   
UPIN replaces PIN1: 'no'   
PIN1 state: 'disabled'   
PIN1 retries: '3'   
PUK1 retries: '10'   
PIN2 state: 'enabled-not-verified'   
PIN2 retries: '3'   
PUK2 retries: '10'   
Example  
The following example is based on a Sierra Wireless MC7710 miniPCIe card (only the form factor in reality it runs UBS) and Deutsche Telekom as ISP. The card is assembled into a PC Engines APU4.

set interfaces wwan wwan0 apn 'internet.telekom'   
set interfaces wwan wwan0 address 'dhcp'   
Supported Modules  
The following hardware modules have been tested successfully in an PC Engines APU4 board:

Sierra Wireless AirPrime MC7304 miniPCIe card (LTE)

Sierra Wireless AirPrime MC7430 miniPCIe card (LTE)

Sierra Wireless AirPrime MC7455 miniPCIe card (LTE)

Sierra Wireless AirPrime MC7710 miniPCIe card (LTE)   
Huawei ME909u-521 miniPCIe card (LTE)   
Huawei ME909s-120 miniPCIe card (LTE)   
Firmware Update  
All available WWAN cards have a build in, reprogrammable firmware. Most of the vendors provide a regular update to the firmware used in the baseband chip.

As VyOS makes use of the QMI interface to connect to the WWAN modem cards, also the firmware can be reprogrammed.

To update the firmware, VyOS also ships the qmi-firmware-update binary. To upgrade the firmware of an e.g. Sierra Wireless MC7710 module to the firmware provided in the file 9999999\_9999999\_9200\_03.05.14.00\_00\_generic\_000.000\_001\_SPKG\_MC.cwe use the following command:   
$ sudo qmi-firmware-update --update -d 1199:68a2 \  
 9999999\_9999999\_9200\_03.05.14.00\_00\_generic\_000.000\_001\_SPKG\_MC.cwe   
WLAN/WIFI - Wireless LAN  
WLAN interface provide 802.11 (a/b/g/n/ac) wireless support (commonly referred to as Wi-Fi) by means of compatible hardware. If your hardware supports it, VyOS supports multiple logical wireless interfaces per physical device.

There are three modes of operation for a wireless interface:   
WAP provides network access to connecting stations if the physical hardware supports acting as a WAP   
A station acts as a Wi-Fi client accessing the network through an available WAP   
Monitor, the system passively monitors any kind of wireless traffic   
If the system detects an unconfigured wireless device, it will be automatically added the configuration tree, specifying any detected settings (for example, its MAC address) and configured to run in monitor mode.

Configuration  
Common interface configuration  
 set interfaces wireless <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces wireless wlan0 address 192.0.2.1/24   
set interfaces wireless wlan0 address 2001:db8::1/64   
set interfaces wireless wlan0 address dhcp   
set interfaces wireless wlan0 address dhcpv6  
 set interfaces wireless <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces wireless wlan0 description 'This is an awesome interface running on VyOS' set interfaces wireless <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces wireless wlan0 disable  
 set interfaces wireless <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard.

A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces wireless wlan0 disable-flow-control  
 set interfaces wireless <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces wireless wlan0 disable-link-detect  
 set interfaces wireless <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces wireless wlan0 mac '00:01:02:03:04:05'  
 set interfaces wireless <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces wireless wlan0 mtu 1600  
 set interfaces wireless <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint

MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces wireless wlan0 ip arp-cache-timeout 180  
 set interfaces wireless <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces wireless wlan0 ip disable-arp-filter  
 set interfaces wireless <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces wireless wlan0 ip disable-forwarding  
 set interfaces wireless <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces wireless wlan0 ip enable-directed-broadcast  
 set interfaces wireless <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces wireless wlan0 ip enable-arp-accept  
 set interfaces wireless <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces wireless wlan0 ip enable-arp-announce  
 set interfaces wireless <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:

If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces wireless wlan0 ip enable-arp-ignore  
 set interfaces wireless <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces wireless wlan0 ip enable-proxy-arp  
 set interfaces wireless <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces wireless <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces wireless <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its

configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces wireless wlan0 ipv6 address autoconf  
 set interfaces wireless <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces wireless wlan0 ipv6 address eui64 2001:db8:beef::/64 set interfaces wireless <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces wireless wlan0 ipv6 address no-default-link-local  
 set interfaces wireless <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces wireless wlan0 ipv6 disable-forwarding  
 set interfaces wireless <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces wireless wlan0 ipv6 accept-dad 2  
 set interfaces wireless <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces wireless wlan0 ipv6 dup-addr-detect-transmits 5 set interfaces wireless <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces wireless wlan0 vrf red   
DHCP(v6)  
 set interfaces wireless <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces wireless wlan0 dhcp-options client-id 'foo-bar'  
 set interfaces wireless <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces wireless wlan0 dhcp-options host-name 'VyOS'  
 set interfaces wireless <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces wireless wlan0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces wireless <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces wireless wlan0 dhcp-options no-default-route  
 set interfaces wireless <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces wireless wlan0 dhcp-options default-route-distance 220  
 set interfaces wireless <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24   
Example:   
set interfaces wireless wlan0 dhcp-options reject 192.168.100.0/24  
 set interfaces wireless <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces wireless wlan0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces wireless <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces wireless wlan0 dhcpv6-options no-release  
 set interfaces wireless <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces wireless wlan0 dhcpv6-options parameters-only  
 set interfaces wireless <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces wireless wlan0 dhcpv6-options rapid-commit  
 set interfaces wireless <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces wireless wlan0 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces wireless <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces wireless wlan0 dhcpv6-options pd 0 length 56  
 set interfaces wireless <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces wireless wlan0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces wireless <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces wireless wlan0 dhcpv6-options pd 0 interface eth8 sla-id 1   
Wireless options  
 set interfaces wireless <interface> channel <number>   
Channel number (IEEE 802.11), for 2.4Ghz (802.11 b/g/n) channels range from 1-14. On 5Ghz (802.11 a/h/j/n/ac) channels available are 0, 34 to 173  
 set interfaces wireless <interface> country-code <cc>   
Country code (ISO/IEC 3166-1). Used to set regulatory domain. Set as needed to indicate country in which device is operating. This can limit available channels and transmit power.

Note   
This option is mandatory in Access-Point mode.

set interfaces wireless <interface> disable-broadcast-ssid   
Send empty SSID in beacons and ignore probe request frames that do not specify full SSID, i.e., require stations to know SSID.

set interfaces wireless <interface> expunge-failing-stations   
Disassociate stations based on excessive transmission failures or other indications of connection loss.

This depends on the driver capabilities and may not be available with all drivers.

set interfaces wireless <interface> isolate-stations   
Client isolation can be used to prevent low-level bridging of frames between associated stations in the BSS.

By default, this bridging is allowed.

set interfaces wireless <interface> max-stations   
Maximum number of stations allowed in station table. New stations will be rejected after the station table is full. IEEE 802.11 has a limit of 2007 different association IDs, so this number should not be larger than that.

This defaults to 2007.

set interfaces wireless <interface> mgmt-frame-protection Management Frame Protection (MFP) according to IEEE 802.11w set interfaces wireless <interface> mode <a | b | g | n | ac> Operation mode of wireless radio.

a - 802.11a - 54 Mbits/sec   
b - 802.11b - 11 Mbits/sec   
g - 802.11g - 54 Mbits/sec (default)   
n - 802.11n - 600 Mbits/sec   
ac - 802.11ac - 1300 Mbits/sec  
 set interfaces wireless <interface> physical-device <device> Wireless hardware device used as underlay radio.

This defaults to phy0.

set interfaces wireless <interface> reduce-transmit-power <number> Add Power Constraint element to Beacon and Probe Response frames.

This option adds Power Constraint element when applicable and Country element is added. Power Constraint element is required by Transmit Power Control.

Valid values are 0..255.

set interfaces wireless <interface> ssid <ssid>   
SSID to be used in IEEE 802.11 management frames  
 set interfaces wireless <interface> type <access-point | station | monitor> Wireless device type for this interface   
access-point - Access-point forwards packets between other nodes

station - Connects to another access point   
monitor - Passively monitor all packets on the frequency/channel  
 set interfaces wireless <interface> per-client-thread   
Provides a per-device control to enable/disable the threaded mode for all the NAPI instances of the given network device, without the need for a device up/down.

If CLI option is not specified, this feature is disabled.

Example:   
set interfaces wireless wlan0 per-client-thread   
PPDU  
 set interfaces wireless <interface> capabilities require-ht  
 set interfaces wireless <interface> capabilities require-hvt   
HT (High Throughput) capabilities (802.11n)  
 set interfaces wireless <interface> capabilities ht 40mhz-incapable   
Device is incapable of 40 MHz, do not advertise. This sets [40-INTOLERANT]  
 set interfaces wireless <interface> capabilities ht auto-powersave   
WMM-PS Unscheduled Automatic Power Save Delivery [U-APSD]  
 set interfaces wireless <interface> capabilities ht channel-set-width <ht20 | ht40+ | ht40-> Supported channel width set.

ht40- - Both 20 MHz and 40 MHz with secondary channel below the primary channel ht40+ - Both 20 MHz and 40 MHz with secondary channel above the primary channel Note   
There are limits on which channels can be used with HT40- and HT40+. Following table shows the channels that may be available for HT40- and HT40+ use per IEEE 802.11n Annex J: Depending on the location, not all of these channels may be available for use!

freq HT40- HT40+   
2.4 GHz 5-13 1-7 (1-9 in Europe/Japan) 5 GHz 40,48,56,64 36,44,52,60   
Note

40 MHz channels may switch their primary and secondary channels if needed or creation of 40 MHz channel maybe rejected based on overlapping BSSes. These changes are done   
automatically when hostapd is setting up the 40 MHz channel.

set interfaces wireless <interface> capabilities ht delayed-block-ack   
Enable HT-delayed Block Ack [DELAYED-BA]  
 set interfaces wireless <interface> capabilities ht dsss-cck-40   
DSSS/CCK Mode in 40 MHz, this sets [DSSS\_CCK-40]  
 set interfaces wireless <interface> capabilities ht greenfield   
This enables the greenfield option which sets the [GF] option  
 set interfaces wireless <interface> capabilities ht ldpc   
Enable LDPC coding capability  
 set interfaces wireless <interface> capabilities ht lsig-protection   
Enable L-SIG TXOP protection capability  
 set interfaces wireless <interface> capabilities ht max-amsdu <3839 | 7935>   
Maximum A-MSDU length 3839 (default) or 7935 octets  
 set interfaces wireless <interface> capabilities ht short-gi <20 | 40>   
Short GI capabilities for 20 and 40 MHz  
 set interfaces wireless <interface> capabilities ht smps <static | dynamic>   
Spatial Multiplexing Power Save (SMPS) settings  
 set interfaces wireless <interface> capabilities ht stbc rx <num>   
Enable receiving PPDU using STBC (Space Time Block Coding)  
 set interfaces wireless <interface> capabilities ht stbc tx   
Enable sending PPDU using STBC (Space Time Block Coding)   
VHT (Very High Throughput) capabilities (802.11ac)  
 set interfaces wireless <interface> capabilities vht antenna-count   
Number of antennas on this card  
 set interfaces wireless <interface> capabilities vht antenna-pattern-fixed   
Set if antenna pattern does not change during the lifetime of an association  
 set interfaces wireless <interface> capabilities vht beamform <single-user-beamformer | single-user-beamformee | multi-user-beamformer | multi-user-beamformee>   
Beamforming capabilities:

single-user-beamformer - Support for operation as single user beamformer   
single-user-beamformee - Support for operation as single user beamformee   
multi-user-beamformer - Support for operation as single user beamformer   
multi-user-beamformee - Support for operation as single user beamformer  
 set interfaces wireless <interface> capabilities vht center-channel-freq <freq-1 | freq-2> <number>   
VHT operating channel center frequency - center freq 1 (for use with 80, 80+80 and 160 modes) VHT operating channel center frequency - center freq 2 (for use with the 80+80 mode) <number> must be from 34 - 173. For 80 MHz channels it should be channel + 6.

set interfaces wireless <interface> capabilities vht channel-set-width <0 | 1 | 2 | 3> 0 - 20 or 40 MHz channel width (default)   
1 - 80 MHz channel width   
2 - 160 MHz channel width   
3 - 80+80 MHz channel width  
 set interfaces wireless <interface> capabilities vht ldpc   
Enable LDPC (Low Density Parity Check) coding capability  
 set interfaces wireless <interface> capabilities vht link-adaptation   
VHT link adaptation capabilities  
 set interfaces wireless <interface> capabilities vht max-mpdu <value>   
Increase Maximum MPDU length to 7991 or 11454 octets (default 3895 octets) set interfaces wireless <interface> capabilities vht max-mpdu-exp <value>   
Set the maximum length of A-MPDU pre-EOF padding that the station can receive set interfaces wireless <interface> capabilities vht short-gi <80 | 160>   
Short GI capabilities  
 set interfaces wireless <interface> capabilities vht stbc rx <num>   
Enable receiving PPDU using STBC (Space Time Block Coding)

set interfaces wireless <interface> capabilities vht stbc tx Enable sending PPDU using STBC (Space Time Block Coding)

set interfaces wireless <interface> capabilities vht tx-powersave Enable VHT TXOP Power Save Mode

set interfaces wireless <interface> capabilities vht vht-cf Station supports receiving VHT variant HT Control field

Wireless options (Station/Client)  
The example creates a wireless station (commonly referred to as Wi-Fi client) that accesses the network through the WAP defined in the above example. The default physical device (phy0) is used.

set interfaces wireless wlan0 type station   
set interfaces wireless wlan0 address dhcp   
set interfaces wireless wlan0 country-code de   
set interfaces wireless wlan0 ssid Test   
set interfaces wireless wlan0 security wpa passphrase '12345678' Resulting in

interfaces {  
 [...]  
 wireless wlan0 {  
 address dhcp  
 country-code de  
 security {  
 wpa {  
 passphrase "12345678"  
 }  
 }  
 ssid TEST  
 type station  
 }   
Security  
WPA and WPA2 Enterprise in combination with 802.1x based authentication can be used to authenticate users or computers in a domain.

The wireless client (supplicant) authenticates against the RADIUS server (authentication server) using an EAP method configured on the RADIUS server. The WAP (also referred to as   
authenticator) role is to send all authentication messages between the supplicant and the configured authentication server, thus the RADIUS server is responsible for authenticating the users.

The WAP in this example has the following characteristics:

IP address 192.168.2.1/24

Network ID (SSID) Enterprise-TEST

WPA passphrase 12345678

Use 802.11n protocol

Wireless channel 1

RADIUS server at 192.168.3.10 with shared-secret VyOSPassword

set interfaces wireless wlan0 address '192.168.2.1/24'   
set interfaces wireless wlan0 country-code de   
set interfaces wireless wlan0 type access-point   
set interfaces wireless wlan0 channel 1   
set interfaces wireless wlan0 mode n   
set interfaces wireless wlan0 ssid 'TEST'   
set interfaces wireless wlan0 security wpa mode wpa2   
set interfaces wireless wlan0 security wpa cipher CCMP   
set interfaces wireless wlan0 security wpa radius server 192.168.3.10 key 'VyOSPassword' set interfaces wireless wlan0 security wpa radius server 192.168.3.10 port 1812   
Resulting in

interfaces {  
 [...]  
 wireless wlan0 {  
 address 192.168.2.1/24  
 country-code de  
 channel 1  
 mode n  
 security {  
 wpa {  
 cipher CCMP  
 mode wpa2  
 radius {  
 server 192.168.3.10 {  
 key 'VyOSPassword'  
 port 1812  
 }  
 }  
 }

}  
 ssid "Enterprise-TEST"  
 type access-point  
 }   
}   
VLAN  
Regular VLANs (802.1q)  
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces wireless <interface> vif <vlan-id>   
Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note

Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces wireless <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64

dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces wireless wlan0 vif 10 address 192.0.2.1/24   
set interfaces wireless wlan0 vif 10 address 2001:db8::1/64   
set interfaces wireless wlan0 vif 10 address dhcp   
set interfaces wireless wlan0 vif 10 address dhcpv6  
 set interfaces wireless <interface> vif <vlan-id> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces wireless wlan0 vif 10 description 'This is an awesome interface running on VyOS' set interfaces wireless <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces wireless wlan0 vif 10 disable  
 set interfaces wireless <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces wireless wlan0 vif 10 disable-link-detect  
 set interfaces wireless <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces wireless wlan0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces wireless <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces wireless wlan0 vif 10 mtu 1600  
 set interfaces wireless <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> vif <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces wireless wlan0 vif 10 ip arp-cache-timeout 180  
 set interfaces wireless <interface> vif <vlan-id> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular

interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces wireless wlan0 vif 10 ip disable-arp-filter  
 set interfaces wireless <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces wireless wlan0 vif 10 ip disable-forwarding  
 set interfaces wireless <interface> vif <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces wireless wlan0 vif 10 ip enable-directed-broadcast  
 set interfaces wireless <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces wireless wlan0 vif 10 ip enable-arp-accept  
 set interfaces wireless <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces wireless wlan0 vif 10 ip enable-arp-announce  
 set interfaces wireless <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces wireless wlan0 vif 10 ip enable-arp-ignore  
 set interfaces wireless <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces wireless wlan0 vif 10 ip enable-proxy-arp  
 set interfaces wireless <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation

Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces wireless <interface> vif <vlan-id> ip source-validation <strict | loose | disable> Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces wireless <interface> vif <vlan-id> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces wireless wlan0 vif 10 ipv6 address autoconf  
 set interfaces wireless <interface> vif <vlan-id> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces wireless wlan0 vif 10 ipv6 address eui64 2001:db8:beef::/64 set interfaces wireless <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:

set interfaces wireless wlan0 vif 10 ipv6 address no-default-link-local  
 set interfaces wireless <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces wireless wlan0 vif 10 ipv6 disable-forwarding  
 set interfaces wireless <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces wireless wlan0 vif 10 ipv6 accept-dad 2  
 set interfaces wireless <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces wireless wlan0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces wireless <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces wireless wlan0 vif 10 vrf red   
DHCP(v6)  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces wireless wlan0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces wireless wlan0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id> The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces wireless wlan0 vif 10 dhcp-options vendor-class-id 'VyOS'  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:

set interfaces wireless wlan0 vif 10 dhcp-options no-default-route  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:

set interfaces wireless wlan0 vif 10 dhcp-options default-route-distance 220  
 set interfaces wireless <interface> vif <vlan-id> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24

Example:

set interfaces wireless wlan0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces wireless wlan0 vif 10 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d' set interfaces wireless <interface> vif <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces wireless wlan0 vif 10 dhcpv6-options no-release  
 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces wireless wlan0 vif 10 dhcpv6-options parameters-only  
 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces wireless wlan0 vif 10 dhcpv6-options rapid-commit  
 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options temporary

Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces wireless wlan0 vif 10 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces wireless <interface> vif <vlan-id> dhcpv6-options pd <id> length <length> Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces wireless wlan0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces wireless wlan0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534 set interfaces wireless <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces wireless wlan0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1 QinQ (802.1ad)  
Call for Contributions

This section needs improvements, examples and explanations.

Please take a look at the Contributing Guide for our Write Documentation.

IEEE 802.1ad was an Ethernet networking standard informally known as QinQ as an amendment to IEEE standard 802.1q VLAN interfaces as described above. 802.1ad was incorporated into the base 802.1q standard in 2011. The technique is also known as provider bridging, Stacked VLANs, or simply QinQ or Q-in-Q. “Q-in-Q” can for supported devices apply to C-tag stacking on C-tag (Ethernet Type = 0x8100).

The original 802.1q specification allows a single Virtual Local Area Network (VLAN) header to be inserted into an Ethernet frame. QinQ allows multiple VLAN tags to be inserted into a single frame, an essential capability for implementing Metro Ethernet network topologies. Just as QinQ extends 802.1Q, QinQ itself is extended by other Metro Ethernet protocols.

In a multiple VLAN header context, out of convenience the term “VLAN tag” or just “tag” for short is often used in place of “802.1q VLAN header”. QinQ allows multiple VLAN tags in an Ethernet frame; together these tags constitute a tag stack. When used in the context of an Ethernet frame, a QinQ frame is a frame that has 2 VLAN 802.1q headers (double-tagged). In VyOS the terms vif-s and vif-c stand for the ethertype tags that are used.

The inner tag is the tag which is closest to the payload portion of the frame. It is officially called C-TAG (customer tag, with ethertype 0x8100). The outer tag is the one closer/closest to the Ethernet header, its name is S-TAG (service tag with Ethernet Type = 0x88a8).

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> address <address | dhcp | dhcpv6>   
Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/

/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 address 192.0.2.1/24   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 address 2001:db8::1/64   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 address dhcp   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 address dhcpv6  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> description <description> Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 description 'This is an awesome interface running on VyOS'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 disable  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 disable-link-detect  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 mac '00:01:02:03:04:05'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 mtu 1600

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip arp-cache-timeout 180  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-arp-filter If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip disable-arp-filter  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip disable-forwarding  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-directed-broadcast Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip enable-directed-broadcast  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-accept Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip enable-arp-accept  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-announce Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip enable-arp-announce  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-ignore

Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip enable-arp-ignore  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ip enable-proxy-arp  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ip source-validation <strict | loose | disable>

Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 address autoconf  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address eui64 <prefix> EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 address eui64 2001:db8:beef::/64 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address no-default-link-local   
Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 address no-default-link-local  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 disable-forwarding Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 disable-forwarding  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 accept-dad 2  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 dup-addr-detect-transmits <n>   
The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:

set interfaces wireless wlan0 vif-s 1000 vif-c 20 ipv6 dup-addr-detect-transmits 5 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> vrf <vrf> Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 vrf red   
DHCP(v6)  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options client-id 'foo-bar'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options host-name 'VyOS'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options vendor-class-id 'VyOS'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options no-default-route  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options default-route-distance <distance>   
Set the distance for the default gateway sent by the DHCP server.

Example:

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options default-route-distance 220 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options reject <address> Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24

Example:

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcp-options reject 192.168.100.0/24  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options duid <duid> The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 duid   
'0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options no-release When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options no-release  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options parameters-only  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options rapid-commit When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options rapid-commit

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options temporary Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 length 56  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces wireless <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces wireless wlan0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 sla-id 1

Operation  
 show interfaces wireless info   
Use this command to view operational status and wireless-specific information about all wireless interfaces.

vyos@vyos:~$ show interfaces wireless info   
Interface Type SSID Channel   
wlan0 access-point VyOS-TEST-0 1  
 show interfaces wireless detail   
Use this command to view operational status and details wireless-specific information about all wireless interfaces.

vyos@vyos:~$ show interfaces wireless detail   
wlan0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000  
 link/ether XX:XX:XX:XX:XX:c3 brd XX:XX:XX:XX:XX:ff  
 inet xxx.xxx.99.254/24 scope global wlan0  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::xxxx:xxxx:fe54:2fc3/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 66072 282 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions 83413 430 0 0 0 0

wlan1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000  
 link/ether XX:XX:XX:XX:XX:c3 brd XX:XX:XX:XX:XX:ff  
 inet xxx.xxx.100.254/24 scope global wlan0  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::xxxx:xxxx:ffff:2ed3/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast  
 166072 5282 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 183413 5430 0 0 0 0  
 show interfaces wireless <wlanX>   
This command shows both status and statistics on the specified wireless interface. The wireless interface identifier can range from wlan0 to wlan999.

vyos@vyos:~$ show interfaces wireless wlan0

wlan0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000  
 link/ether XX:XX:XX:XX:XX:c3 brd XX:XX:XX:XX:XX:ff  
 inet xxx.xxx.99.254/24 scope global wlan0  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::xxxx:xxxx:fe54:2fc3/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast  
 66072 282 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 83413 430 0 0 0 0  
 show interfaces wireless <wlanX> brief   
This command gives a brief status overview of a specified wireless interface. The wireless interface identifier can range from wlan0 to wlan999.

vyos@vyos:~$ show interfaces wireless wlan0 brief   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
wlan0 192.168.2.254/24 u/u  
 show interfaces wireless <wlanX> queue   
Use this command to view wireless interface queue information. The wireless interface identifier can range from wlan0 to wlan999.

vyos@vyos:~$ show interfaces wireless wlan0 queue   
qdisc pfifo\_fast 0: root bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1  
 Sent 810323 bytes 6016 pkt (dropped 0, overlimits 0 requeues 0)  
 rate 0bit 0pps backlog 0b 0p requeues 0  
 show interfaces wireless <wlanX> scan   
This command is used to retrieve information about WAP within the range of your wireless interface. This command is useful on wireless interfaces configured in station mode.

Note

Scanning is not supported on all wireless drivers and wireless hardware. Refer to your driver and wireless hardware documentation for further details.

vyos@vyos:~$ show interfaces wireless wlan0 scan   
Address SSID Channel Signal (dbm) 00:53:3b:88:6e:d8 WLAN-576405 1 -64.00 00:53:3b:88:6e:da Telekom\_FON 1 -64.00 00:53:00:f2:c2:a4 BabyView\_F2C2A4 6 -60.00 00:53:3b:88:6e:d6 Telekom\_FON 100 -72.00

00:53:3b:88:6e:d4 WLAN-576405 100 -71.00   
00:53:44:a4:96:ec KabelBox-4DC8 56 -81.00   
00:53:d9:7a:67:c2 WLAN-741980 1 -75.00   
00:53:7c:99:ce:76 Vodafone Homespot 1 -86.00   
00:53:44:a4:97:21 KabelBox-4DC8 1 -78.00   
00:53:44:a4:97:21 Vodafone Hotspot 1 -79.00   
00:53:44:a4:97:21 Vodafone Homespot 1 -79.00   
00:53:86:40:30:da Telekom\_FON 1 -86.00   
00:53:7c:99:ce:76 Vodafone Hotspot 1 -86.00   
00:53:44:46:d2:0b Vodafone Hotspot 1 -87.00   
Examples  
The following example creates a WAP. When configuring multiple WAP interfaces, you must specify unique IP addresses, channels, Network IDs commonly referred to as SSID, and MAC addresses.

The WAP in this example has the following characteristics:

IP address 192.168.2.1/24

Network ID (SSID) TEST

WPA passphrase 12345678

Use 802.11n protocol

Wireless channel 1

set interfaces wireless wlan0 address '192.168.2.1/24'   
set interfaces wireless wlan0 type access-point   
set interfaces wireless wlan0 channel 1   
set interfaces wireless wlan0 mode n   
set interfaces wireless wlan0 ssid 'TEST'   
set interfaces wireless wlan0 security wpa mode wpa2   
set interfaces wireless wlan0 security wpa cipher CCMP   
set interfaces wireless wlan0 security wpa passphrase '12345678' set interfaces wireless wlan0 country-code de   
Resulting in

interfaces {  
 [...]  
 wireless wlan0 {  
 address 192.168.2.1/24  
 channel 1  
 country-code de

mode n  
 security {  
 wpa {  
 cipher CCMP  
 mode wpa2  
 passphrase "12345678"  
 }  
 }  
 ssid "TEST"  
 type access-point  
 }   
}   
system {  
 [...]  
 wifi-regulatory-domain DE   
}   
To get it to work as an access point with this configuration you will need to set up a DHCP server to work with that network. You can - of course - also bridge the Wireless interface with any configured bridge (Bridge) on the system.

Intel AX200  
The Intel AX200 card does not work out of the box in AP mode, see   
https://unix.stackexchange.com/questions/598275/intel-ax200-ap-mode. You can still put this card into AP mode using the following configuration:

set interfaces wireless wlan0 channel '1'   
set interfaces wireless wlan0 country-code 'us' set interfaces wireless wlan0 mode 'n'   
set interfaces wireless wlan0 physical-device 'phy0' set interfaces wireless wlan0 ssid 'VyOS'   
set interfaces wireless wlan0 type 'access-point'

VXLAN  
VXLAN is a network virtualization technology that attempts to address the scalability problems associated with large cloud computing deployments. It uses a VLAN-like encapsulation technique to encapsulate OSI layer 2 Ethernet frames within layer 4 UDP datagrams, using 4789 as the default IANA-assigned destination UDP port number. VXLAN endpoints, which terminate VXLAN tunnels and may be either virtual or physical switch ports, are known as VTEPs.

VXLAN is an evolution of efforts to standardize an overlay encapsulation protocol. It increases the scalability up to 16 million logical networks and allows for layer 2 adjacency across IP networks. Multicast or unicast with head-end replication (HER) is used to flood broadcast, unknown unicast, and multicast (BUM) traffic.

The VXLAN specification was originally created by VMware, Arista Networks and Cisco. Other backers of the VXLAN technology include Huawei, Broadcom, Citrix, Pica8, Big Switch Networks, Cumulus Networks, Dell EMC, Ericsson, Mellanox, FreeBSD, OpenBSD, Red Hat, Joyent, and Juniper Networks.

VXLAN was officially documented by the IETF in RFC 7348.

If configuring VXLAN in a VyOS virtual machine, ensure that MAC spoofing (Hyper-V) or Forged Transmits (ESX) are permitted, otherwise forwarded frames may be blocked by the hypervisor.

Note   
As VyOS is based on Linux and there was no official IANA port assigned for VXLAN, VyOS uses a default port of 8472. You can change the port on a per VXLAN interface basis to get it working across multiple vendors.

Configuration  
Common interface configuration  
 set interfaces vxlan <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
Example:   
set interfaces vxlan vxlan0 address 192.0.2.1/24   
set interfaces vxlan vxlan0 address 2001:db8::1/64  
 set interfaces vxlan <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces vxlan vxlan0 description 'This is an awesome interface running on VyOS' set interfaces vxlan <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces vxlan vxlan0 disable  
 set interfaces vxlan <interface> disable-flow-control

Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces vxlan vxlan0 disable-flow-control  
 set interfaces vxlan <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces vxlan vxlan0 disable-link-detect  
 set interfaces vxlan <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces vxlan vxlan0 mac '00:01:02:03:04:05'  
 set interfaces vxlan <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces vxlan vxlan0 mtu 1600  
 set interfaces vxlan <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces vxlan <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces vxlan vxlan0 ip arp-cache-timeout 180  
 set interfaces vxlan <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces vxlan vxlan0 ip disable-arp-filter  
 set interfaces vxlan <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces vxlan vxlan0 ip disable-forwarding

set interfaces vxlan <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces vxlan vxlan0 ip enable-directed-broadcast  
 set interfaces vxlan <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces vxlan vxlan0 ip enable-arp-accept  
 set interfaces vxlan <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces vxlan vxlan0 ip enable-arp-announce  
 set interfaces vxlan <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces vxlan vxlan0 ip enable-arp-ignore

set interfaces vxlan <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces vxlan vxlan0 ip enable-proxy-arp  
 set interfaces vxlan <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces vxlan <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces vxlan <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces vxlan vxlan0 ipv6 address autoconf  
 set interfaces vxlan <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces vxlan vxlan0 ipv6 address eui64 2001:db8:beef::/64 set interfaces vxlan <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces vxlan vxlan0 ipv6 address no-default-link-local  
 set interfaces vxlan <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces vxlan vxlan0 ipv6 disable-forwarding  
 set interfaces vxlan <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces vxlan <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces vxlan vxlan0 ipv6 accept-dad 2  
 set interfaces vxlan <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces vxlan vxlan0 ipv6 dup-addr-detect-transmits 5 set interfaces vxlan <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces vxlan vxlan0 vrf red   
VXLAN specific options

set interfaces vxlan <interface> vni <number>   
Each VXLAN segment is identified through a 24-bit segment ID, termed the VNI, This allows up to 16M VXLAN segments to coexist within the same administrative domain.

set interfaces vxlan <interface> port <port>   
Configure port number of remote VXLAN endpoint.

Note

As VyOS is Linux based the default port used is not using 4789 as the default IANA-assigned destination UDP port number. Instead VyOS uses the Linux default port of 8472.

set interfaces vxlan <interface> source-address <interface>   
Source IP address used for VXLAN underlay. This is mandatory when using VXLAN via L2VPN/EVPN.

set interfaces vxlan <interface> gpe   
Enables the Generic Protocol extension (VXLAN-GPE). Currently, this is only supported together with the external keyword.

set interfaces vxlan <interface> parameters external   
Specifies whether an external control plane (e.g. BGP L2VPN/EVPN) or the internal FDB should be used.

set interfaces vxlan <interface> parameters neighbor-suppress   
In order to minimize the flooding of ARP and ND messages in the VXLAN network, EVPN includes provisions RFC 7432#section-10 that allow participating VTEPs to suppress such messages in case they know the MAC-IP binding and can reply on behalf of the remote host.

set interfaces vxlan <interface> parameters nolearning   
Specifies if unknown source link layer addresses and IP addresses are entered into the VXLAN device forwarding database.

Unicast  
 set interfaces vxlan <interface> remote <address>   
IPv4/IPv6 remote address of the VXLAN tunnel. Alternative to multicast, the remote IPv4/IPv6 address can set directly.

Multicast  
 set interfaces vxlan <interface> source-interface <interface>   
Interface used for VXLAN underlay. This is mandatory when using VXLAN via a multicast network. VXLAN traffic will always enter and exit this interface.

set interfaces vxlan <interface> group <address>

Multicast group address for VXLAN interface. VXLAN tunnels can be built either via Multicast or via Unicast.

Both IPv4 and IPv6 multicast is possible.

Multicast VXLAN  
Topology: PC4 - Leaf2 - Spine1 - Leaf3 - PC5

PC4 has IP 10.0.0.4/24 and PC5 has IP 10.0.0.5/24, so they believe they are in the same broadcast domain.

Let’s assume PC4 on Leaf2 wants to ping PC5 on Leaf3. Instead of setting Leaf3 as our remote end manually, Leaf2 encapsulates the packet into a UDP-packet and sends it to its designated multicast-address via Spine1. When Spine1 receives this packet it forwards it to all other leaves who has joined the same multicast-group, in this case Leaf3. When Leaf3 receives the packet it forwards it, while at the same time learning that PC4 is reachable behind Leaf2, because the encapsulated packet had Leaf2’s IP address set as source IP.

PC5 receives the ping echo, responds with an echo reply that Leaf3 receives and this time forwards to Leaf2’s unicast address directly because it learned the location of PC4 above. When Leaf2 receives the echo reply from PC5 it sees that it came from Leaf3 and so remembers that PC5 is reachable via Leaf3.

Thanks to this discovery, any subsequent traffic between PC4 and PC5 will not be using the multicast-address between the leaves as they both know behind which Leaf the PCs are connected. This saves traffic as less multicast packets sent reduces the load on the network, which improves scalability when more leaves are added.

For optimal scalability, Multicast shouldn’t be used at all, but instead use BGP to signal all connected devices between leaves. Unfortunately, VyOS does not yet support this.

Single VXLAN device (SVD)  
FRR supports a new way of configuring VLAN-to-VNI mappings for EVPN-VXLAN, when working with the Linux kernel. In this new way, the mapping of a VLAN to a VNI is configured against a container VXLAN interface which is referred to as a SVD.

Multiple VLAN to VNI mappings can be configured against the same SVD. This allows for a significant scaling of the number of VNIs since a separate VXLAN interface is no longer required for each VNI.

set interfaces vxlan <interface> vlan-to-vni <vlan> vni <vni>   
Maps the VNI to the specified VLAN id. The VLAN can then be consumed by a bridge.

Sample configuration of SVD with VLAN to VNI mappings is shown below.

set interfaces bridge br0 member interface vxlan0 set interfaces vxlan vxlan0 parameters external set interfaces vxlan vxlan0 source-interface 'dum0' set interfaces vxlan vxlan0 vlan-to-vni 10 vni '10010' set interfaces vxlan vxlan0 vlan-to-vni 11 vni '10011' set interfaces vxlan vxlan0 vlan-to-vni 30 vni '10030' set interfaces vxlan vxlan0 vlan-to-vni 31 vni '10031' Example  
The setup is this: Leaf2 - Spine1 - Leaf3

Spine1 is a Cisco IOS router running version 15.4, Leaf2 and Leaf3 is each a VyOS router running 1.2.

This topology was built using GNS3.

Topology:

Spine1:   
fa0/2 towards Leaf2, IP-address: 10.1.2.1/24   
fa0/3 towards Leaf3, IP-address: 10.1.3.1/24

Leaf2:   
Eth0 towards Spine1, IP-address: 10.1.2.2/24   
Eth1 towards a vlan-aware switch

Leaf3:   
Eth0 towards Spine1, IP-address 10.1.3.3/24   
Eth1 towards a vlan-aware switch   
Spine1 Configuration:

conf t   
ip multicast-routing   
!

interface fastethernet0/2  
 ip address 10.1.2.1 255.255.255.0  
 ip pim sparse-dense-mode   
!

interface fastethernet0/3  
 ip address 10.1.3.1 255.255.255.0  
 ip pim sparse-dense-mode   
!

router ospf 1  
 network 10.0.0.0 0.255.255.255 area 0

Multicast-routing is required for the leaves to forward traffic between each other in a more scalable way. This also requires PIM to be enabled towards the leaves so that the Spine can learn what multicast groups each Leaf expects traffic from.

Leaf2 configuration:

set interfaces ethernet eth0 address '10.1.2.2/24' set protocols ospf area 0 network '10.0.0.0/8'

! Our first vxlan interface   
set interfaces bridge br241 address '172.16.241.1/24' set interfaces bridge br241 member interface 'eth1.241' set interfaces bridge br241 member interface 'vxlan241'

set interfaces vxlan vxlan241 group '239.0.0.241' set interfaces vxlan vxlan241 source-interface 'eth0' set interfaces vxlan vxlan241 vni '241'

! Our seconds vxlan interface   
set interfaces bridge br242 address '172.16.242.1/24' set interfaces bridge br242 member interface 'eth1.242' set interfaces bridge br242 member interface 'vxlan242'

set interfaces vxlan vxlan242 group '239.0.0.242' set interfaces vxlan vxlan242 source-interface 'eth0' set interfaces vxlan vxlan242 vni '242'   
Leaf3 configuration:

set interfaces ethernet eth0 address '10.1.3.3/24' set protocols ospf area 0 network '10.0.0.0/8'

! Our first vxlan interface   
set interfaces bridge br241 address '172.16.241.1/24' set interfaces bridge br241 member interface 'eth1.241' set interfaces bridge br241 member interface 'vxlan241'

set interfaces vxlan vxlan241 group '239.0.0.241' set interfaces vxlan vxlan241 source-interface 'eth0' set interfaces vxlan vxlan241 vni '241'

! Our seconds vxlan interface   
set interfaces bridge br242 address '172.16.242.1/24' set interfaces bridge br242 member interface 'eth1.242' set interfaces bridge br242 member interface 'vxlan242'

set interfaces vxlan vxlan242 group '239.0.0.242'   
set interfaces vxlan vxlan242 source-interface 'eth0'   
set interfaces vxlan vxlan242 vni '242'   
As you can see, Leaf2 and Leaf3 configuration is almost identical. There are lots of commands above, I’ll try to into more detail below, command descriptions are placed under the command boxes:

set interfaces bridge br241 address '172.16.241.1/24'   
This commands creates a bridge that is used to bind traffic on eth1 vlan 241 with the vxlan241-interface. The IP address is not required. It may however be used as a default gateway for each Leaf which allows devices on the vlan to reach other subnets. This requires that the subnets are redistributed by OSPF so that the Spine will learn how to reach it. To do this you need to change the OSPF network from ‘10.0.0.0/8’ to ‘0.0.0.0/0’ to allow 172.16/12-networks to be advertised.

set interfaces bridge br241 member interface 'eth1.241'   
set interfaces bridge br241 member interface 'vxlan241'   
Binds eth1.241 and vxlan241 to each other by making them both member interfaces of the same bridge.

set interfaces vxlan vxlan241 group '239.0.0.241'   
The multicast-group used by all leaves for this vlan extension. Has to be the same on all leaves that has this interface.

set interfaces vxlan vxlan241 source-interface 'eth0'   
Sets the interface to listen for multicast packets on. Could be a loopback, not yet tested.

set interfaces vxlan vxlan241 vni '241'   
Sets the unique id for this vxlan-interface. Not sure how it correlates with multicast-address.

set interfaces vxlan vxlan241 port 12345   
The destination port used for creating a VXLAN interface in Linux defaults to its pre-standard value of 8472 to preserve backward compatibility. A configuration directive to support a user-specified destination port to override that behavior is available using the above command.

Unicast VXLAN  
Alternative to multicast, the remote IPv4 address of the VXLAN tunnel can be set directly. Let’s change the Multicast example from above:

# leaf2 and leaf3   
delete interfaces vxlan vxlan241 group '239.0.0.241' delete interfaces vxlan vxlan241 source-interface 'eth0'

# leaf2

set interface vxlan vxlan241 remote 10.1.3.3

# leaf3   
set interface vxlan vxlan241 remote 10.1.2.2   
The default port udp is set to 8472. It can be changed with set interface vxlan <vxlanN> port <port>

VTI - Virtual Tunnel Interface  
Set Virtual Tunnel Interface

set interfaces vti vti0 address 192.168.2.249/30 set interfaces vti vti0 address 2001:db8:2::249/64 Results in:

vyos@vyos# show interfaces vti   
vti vti0 {  
 address 192.168.2.249/30  
 address 2001:db8:2::249/64  
 description "Description"   
}   
Warning

When using site-to-site IPsec with VTI interfaces, be sure to disable route autoinstall

set vpn ipsec options disable-route-autoinstall   
More details about the IPsec and VTI issue and option disable-route-autoinstall https://blog.vyos.io/vyos-1-dot-2-0-development-news-in-july

The root cause of the problem is that for VTI tunnels to work, their traffic selectors have to be set to 0.0.0.0/0 for traffic to match the tunnel, even though actual routing decision is made according to netfilter marks. Unless route insertion is disabled entirely, StrongSWAN thus mistakenly inserts a default route through the VTI peer address, which makes all traffic routed to nowhere.

Virtual Ethernet  
The veth devices are virtual Ethernet devices. They can act as tunnels between network namespaces to create a bridge to a physical network device in another namespace or VRF, but can also be used as standalone network devices.

Note

veth interfaces need to be created in pairs - it’s called the peer name

Configuration  
Common interface configuration  
 set interfaces virtual-ethernet <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces virtual-ethernet veth0 address 192.0.2.1/24   
set interfaces virtual-ethernet veth0 address 2001:db8::1/64   
set interfaces virtual-ethernet veth0 address dhcp   
set interfaces virtual-ethernet veth0 address dhcpv6  
 set interfaces virtual-ethernet <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces virtual-ethernet veth0 description 'This is an awesome interface running on VyOS' VLAN  
Regular VLANs (802.1q)  
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces virtual-ethernet <interface> vif <vlan-id>   
Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note   
Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces virtual-ethernet <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/

Example:   
set interfaces virtual-ethernet veth0 vif 10 address 192.0.2.1/24   
set interfaces virtual-ethernet veth0 vif 10 address 2001:db8::1/64   
set interfaces virtual-ethernet veth0 vif 10 address dhcp   
set interfaces virtual-ethernet veth0 vif 10 address dhcpv6  
 set interfaces virtual-ethernet <interface> vif <vlan-id> description <description> Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces virtual-ethernet veth0 vif 10 description 'This is an awesome interface running on VyOS'  
 set interfaces virtual-ethernet <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces virtual-ethernet veth0 vif 10 disable  
 set interfaces virtual-ethernet <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces virtual-ethernet veth0 vif 10 disable-link-detect  
 set interfaces virtual-ethernet <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces virtual-ethernet veth0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces virtual-ethernet <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces virtual-ethernet veth0 vif 10 mtu 1600  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces virtual-ethernet <interface> vif <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ip arp-cache-timeout 180  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:

set interfaces virtual-ethernet veth0 vif 10 ip disable-arp-filter  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces virtual-ethernet veth0 vif 10 ip disable-forwarding  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces virtual-ethernet veth0 vif 10 ip enable-directed-broadcast  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces virtual-ethernet veth0 vif 10 ip enable-arp-accept  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces virtual-ethernet veth0 vif 10 ip enable-arp-announce  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:

If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces virtual-ethernet veth0 vif 10 ip enable-arp-ignore  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ip enable-proxy-arp  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 address autoconf  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 address eui64 <prefix> EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 address eui64 2001:db8:beef::/64 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 address no-default-link-local  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 disable-forwarding  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 accept-dad 2  
 set interfaces virtual-ethernet <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces virtual-ethernet veth0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces virtual-ethernet <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces virtual-ethernet veth0 vif 10 vrf red   
DHCP(v6)  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options client-id <description> RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces virtual-ethernet veth0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options host-name <hostname> Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces virtual-ethernet veth0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces virtual-ethernet veth0 vif 10 dhcp-options vendor-class-id 'VyOS' set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces virtual-ethernet veth0 vif 10 dhcp-options no-default-route  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options default-route-distance <distance>   
Set the distance for the default gateway sent by the DHCP server.

Example:

set interfaces virtual-ethernet veth0 vif 10 dhcp-options default-route-distance 220  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcp-options reject <address> Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24

Example:

set interfaces virtual-ethernet veth0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces virtual-ethernet veth0 vif 10 duid   
'0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options no-release  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options parameters-only This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options parameters-only  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options rapid-commit When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options rapid-commit  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options temporary Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> length <length> Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534 set interfaces virtual-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces virtual-ethernet veth0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1 QinQ (802.1ad)  
Call for Contributions   
This section needs improvements, examples and explanations.

Please take a look at the Contributing Guide for our Write Documentation.

IEEE 802.1ad was an Ethernet networking standard informally known as QinQ as an amendment to IEEE standard 802.1q VLAN interfaces as described above. 802.1ad was incorporated into the base 802.1q standard in 2011. The technique is also known as provider bridging, Stacked VLANs,

or simply QinQ or Q-in-Q. “Q-in-Q” can for supported devices apply to C-tag stacking on C-tag (Ethernet Type = 0x8100).

The original 802.1q specification allows a single Virtual Local Area Network (VLAN) header to be inserted into an Ethernet frame. QinQ allows multiple VLAN tags to be inserted into a single frame, an essential capability for implementing Metro Ethernet network topologies. Just as QinQ extends 802.1Q, QinQ itself is extended by other Metro Ethernet protocols.

In a multiple VLAN header context, out of convenience the term “VLAN tag” or just “tag” for short is often used in place of “802.1q VLAN header”. QinQ allows multiple VLAN tags in an Ethernet frame; together these tags constitute a tag stack. When used in the context of an Ethernet frame, a QinQ frame is a frame that has 2 VLAN 802.1q headers (double-tagged). In VyOS the terms vif-s and vif-c stand for the ethertype tags that are used.

The inner tag is the tag which is closest to the payload portion of the frame. It is officially called C-TAG (customer tag, with ethertype 0x8100). The outer tag is the one closer/closest to the Ethernet header, its name is S-TAG (service tag with Ethernet Type = 0x88a8).

set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> address <address | dhcp | dhcpv6>   
Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 address 192.0.2.1/24   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 address 2001:db8::1/64   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 address dhcp

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 address dhcpv6  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 description 'This is an awesome interface running on VyOS'  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 disable  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> disable-link-detect Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 disable-link-detect  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 mac '00:01:02:03:04:05'  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 mtu 1600  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip arp-cache-timeout Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip arp-cache-timeout 180  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-arp-filter If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip disable-arp-filter  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-forwarding Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip disable-forwarding  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip enable-directed-broadcast  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-accept Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip enable-arp-accept  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-announce Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip enable-arp-announce  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-ignore Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:

If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip enable-arp-ignore  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-proxy-arp Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ip enable-proxy-arp  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip proxy-arp-pvlan Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address autoconf SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 address autoconf  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 address eui64 2001:db8:beef::/64 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address no-default-link-local   
Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 address no-default-link-local set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 disable-forwarding Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 disable-forwarding  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 accept-dad 2  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 dup-addr-detect-transmits <n>   
The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 ipv6 dup-addr-detect-transmits 5 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> vrf <vrf> Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 vrf red   
DHCP(v6)  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options client-id 'foo-bar'  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options host-name 'VyOS' set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options vendor-class-id 'VyOS' set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options no-default-route set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options default-route-distance <distance>

Set the distance for the default gateway sent by the DHCP server.

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options default-route-distance 220 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24

Example:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcp-options reject 192.168.100.0/24 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 duid   
'0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options no-release  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options parameters-only set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options rapid-commit

set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 length 56  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces virtual-ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces virtual-ethernet veth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 sla-id 1  
 set interfaces virtual-ethernet <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:

set interfaces virtual-ethernet veth0 disable  
 set interfaces virtual-ethernet <interface> vrf <vrf> Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:

set interfaces virtual-ethernet veth0 vrf red   
Operation  
 show interfaces virtual-ethernet   
Show brief interface information.

vyos@vyos:~$ show interfaces virtual-ethernet   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down Interface IP Address S/L Description--------- ---------- --- -----------  
veth10 100.64.0.0/31 u/u   
veth11 100.64.0.1/31 u/u  
 show interfaces virtual-ethernet <interface>   
Show detailed information on given <interface>

vyos@vyos:~$ show interfaces virtual-ethernet veth11   
10: veth11@veth10: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue master red state UP group default qlen 1000  
 link/ether b2:7b:df:47:e9:11 brd ff:ff:ff:ff:ff:ff  
 inet 100.64.0.1/31 scope global veth11  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::b07b:dfff:fe47:e911/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 0 0 0 0 0 0

TX: bytes packets errors dropped carrier collisions  
 1369707 4267 0 0 0 0   
Example  
Interconnect the global VRF with vrf “red” using the veth10 <-> veth 11 pair

set interfaces virtual-ethernet veth10 address '100.64.0.0/31' set interfaces virtual-ethernet veth10 peer-name 'veth11' set interfaces virtual-ethernet veth11 address '100.64.0.1/31' set interfaces virtual-ethernet veth11 peer-name 'veth10' set interfaces virtual-ethernet veth11 vrf 'red'   
set vrf name red table '1000'

vyos@vyos:~$ ping 100.64.0.1   
PING 100.64.0.1 (100.64.0.1) 56(84) bytes of data.

64 bytes from 100.64.0.1: icmp\_seq=1 ttl=64 time=0.080 ms 64 bytes from 100.64.0.1: icmp\_seq=2 ttl=64 time=0.119 ms

Tunnel  
This article touches on ‘classic’ IP tunneling protocols.

GRE is often seen as a one size fits all solution when it comes to classic IP tunneling protocols, and for a good reason. However, there are more specialized options, and many of them are supported by VyOS. There are also rather obscure GRE options that can be useful.

All those protocols are grouped under interfaces tunnel in VyOS. Let’s take a closer look at the protocols and options currently supported by VyOS.

Common interface configuration  
 set interfaces tunnel <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64

Example:

set interfaces tunnel tun0 address 192.0.2.1/24   
set interfaces tunnel tun0 address 2001:db8::1/64  
 set interfaces tunnel <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:

set interfaces tunnel tun0 description 'This is an awesome interface running on VyOS' set interfaces tunnel <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces tunnel tun0 disable  
 set interfaces tunnel <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces tunnel tun0 disable-flow-control  
 set interfaces tunnel <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces tunnel tun0 disable-link-detect  
 set interfaces tunnel <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces tunnel tun0 mtu 1600  
 set interfaces tunnel <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces tunnel <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces tunnel tun0 ip arp-cache-timeout 180  
 set interfaces tunnel <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces tunnel tun0 ip disable-arp-filter  
 set interfaces tunnel <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces tunnel tun0 ip disable-forwarding  
 set interfaces tunnel <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces tunnel tun0 ip enable-directed-broadcast  
 set interfaces tunnel <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces tunnel tun0 ip enable-arp-accept  
 set interfaces tunnel <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces tunnel tun0 ip enable-arp-announce  
 set interfaces tunnel <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces tunnel tun0 ip enable-arp-ignore  
 set interfaces tunnel <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces tunnel tun0 ip enable-proxy-arp  
 set interfaces tunnel <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces tunnel <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces tunnel <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces tunnel tun0 ipv6 address autoconf  
 set interfaces tunnel <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces tunnel tun0 ipv6 address eui64 2001:db8:beef::/64 set interfaces tunnel <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces tunnel tun0 ipv6 address no-default-link-local  
 set interfaces tunnel <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces tunnel tun0 ipv6 disable-forwarding  
 set interfaces tunnel <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces tunnel <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces tunnel tun0 ipv6 accept-dad 2  
 set interfaces tunnel <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces tunnel tun0 ipv6 dup-addr-detect-transmits 5 set interfaces tunnel <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:

set interfaces tunnel tun0 vrf red   
IPIP  
This is one of the simplest types of tunnels, as defined by RFC 2003. It takes an IPv4 packet and sends it as a payload of another IPv4 packet. For this reason, there are no other configuration options for this kind of tunnel.

An example:

set interfaces tunnel tun0 encapsulation ipip   
set interfaces tunnel tun0 source-address 192.0.2.10   
set interfaces tunnel tun0 remote 203.0.113.20   
set interfaces tunnel tun0 address 192.168.100.200/24   
IP6IP6  
This is the IPv6 counterpart of IPIP. I’m not aware of an RFC that defines this encapsulation specifically, but it’s a natural specific case of IPv6 encapsulation mechanisms described in :rfc:2473`.

It’s not likely that anyone will need it any time soon, but it does exist.

An example:

set interfaces tunnel tun0 encapsulation ip6ip6   
set interfaces tunnel tun0 source-address 2001:db8:aa::1   
set interfaces tunnel tun0 remote 2001:db8:aa::2   
set interfaces tunnel tun0 address 2001:db8:bb::1/64   
IPIP6  
In the future this is expected to be a very useful protocol (though there are other proposals).

As the name implies, it’s IPv4 encapsulated in IPv6, as simple as that.

An example:

set interfaces tunnel tun0 encapsulation ipip6   
set interfaces tunnel tun0 source-address 2001:db8:aa::1   
set interfaces tunnel tun0 remote 2001:db8:aa::2   
set interfaces tunnel tun0 address 192.168.70.80/24   
6in4 (SIT)  
6in4 uses tunneling to encapsulate IPv6 traffic over IPv4 links as defined in RFC 4213. The 6in4 traffic is sent over IPv4 inside IPv4 packets whose IP headers have the IP protocol number set to 41. This protocol number is specifically designated for IPv6 encapsulation, the IPv4 packet header is immediately followed by the IPv6 packet being carried. The encapsulation overhead is the size of the IPv4 header of 20 bytes, therefore with an MTU of 1500 bytes, IPv6 packets of 1480 bytes can be sent without fragmentation. This tunneling technique is frequently used by IPv6 tunnel brokers like Hurricane Electric.

An example:

set interfaces tunnel tun0 encapsulation sit   
set interfaces tunnel tun0 source-address 192.0.2.10   
set interfaces tunnel tun0 remote 192.0.2.20   
set interfaces tunnel tun0 address 2001:db8:bb::1/64   
A full example of a Tunnelbroker.net config can be found at here.

Generic Routing Encapsulation (GRE)  
A GRE tunnel operates at layer 3 of the OSI model and is represented by IP protocol 47. The main benefit of a GRE tunnel is that you are able to carry multiple protocols inside the same tunnel. GRE also supports multicast traffic and supports routing protocols that leverage multicast to form neighbor adjacencies.

A VyOS GRE tunnel can carry both IPv4 and IPv6 traffic and can also be created over either IPv4 (gre) or IPv6 (ip6gre).

Configuration  
A basic configuration requires a tunnel source (source-address), a tunnel destination (remote), an encapsulation type (gre), and an address (ipv4/ipv6). Below is a basic IPv4 only configuration example taken from a VyOS router and a Cisco IOS router. The main difference between these two configurations is that VyOS requires you explicitly configure the encapsulation type. The Cisco router defaults to GRE IP otherwise it would have to be configured as well.

VyOS Router:

set interfaces tunnel tun100 address '10.0.0.1/30'   
set interfaces tunnel tun100 encapsulation 'gre'   
set interfaces tunnel tun100 source-address '198.51.100.2' set interfaces tunnel tun100 remote '203.0.113.10'   
Cisco IOS Router:

interface Tunnel100   
ip address 10.0.0.2 255.255.255.252   
tunnel source 203.0.113.10   
tunnel destination 198.51.100.2   
Here is a second example of a dual-stack tunnel over IPv6 between a VyOS router and a Linux host using systemd-networkd.

VyOS Router:

set interfaces tunnel tun101 address '2001:db8:feed:beef::1/126' set interfaces tunnel tun101 address '192.168.5.1/30'

set interfaces tunnel tun101 encapsulation 'ip6gre'   
set interfaces tunnel tun101 source-address '2001:db8:babe:face::3afe:3' set interfaces tunnel tun101 remote '2001:db8:9bb:3ce::5'   
Linux systemd-networkd:

This requires two files, one to create the device (XXX.netdev) and one to configure the network on the device (XXX.network)

# cat /etc/systemd/network/gre-example.netdev [NetDev]   
Name=gre-example   
Kind=ip6gre   
MTUBytes=14180

[Tunnel]   
Remote=2001:db8:babe:face::3afe:3

# cat /etc/systemd/network/gre-example.network [Match]   
Name=gre-example

[Network]   
Address=2001:db8:feed:beef::2/126

[Address]   
Address=192.168.5.2/30   
Tunnel keys  
GRE is also the only classic protocol that allows creating multiple tunnels with the same source and destination due to its support for tunnel keys. Despite its name, this feature has nothing to do with security: it’s simply an identifier that allows routers to tell one tunnel from another.

An example:

set interfaces tunnel tun0 source-address 192.0.2.10 set interfaces tunnel tun0 remote 192.0.2.20   
set interfaces tunnel tun0 address 10.40.50.60/24 set interfaces tunnel tun0 parameters ip key 10   
set interfaces tunnel tun0 source-address 192.0.2.10 set interfaces tunnel tun0 remote 192.0.2.20   
set interfaces tunnel tun0 address 172.16.17.18/24 set interfaces tunnel tun0 parameters ip key 20   
GRETAP

While normal GRE is for layer 3, GRETAP is for layer 2. GRETAP can encapsulate Ethernet frames, thus it can be bridged with other interfaces to create datalink layer segments that span multiple remote sites.

set interfaces bridge br0 member interface eth0   
set interfaces bridge br0 member interface tun0   
set interfaces tunnel tun0 encapsulation gretap   
set interfaces tunnel tun0 source-address 198.51.100.2   
set interfaces tunnel tun0 remote 203.0.113.10   
Troubleshooting  
GRE is a well defined standard that is common in most networks. While not inherently difficult to configure there are a couple of things to keep in mind to make sure the configuration performs as expected. A common cause for GRE tunnels to fail to come up correctly include ACL or Firewall configurations that are discarding IP protocol 47 or blocking your source/destination traffic.

1. Confirm IP connectivity between tunnel source-address and remote:

vyos@vyos:~$ ping 203.0.113.10 interface 198.51.100.2 count 4   
PING 203.0.113.10 (203.0.113.10) from 198.51.100.2 : 56(84) bytes of data.

64 bytes from 203.0.113.10: icmp\_seq=1 ttl=254 time=0.807 ms 64 bytes from 203.0.113.10: icmp\_seq=2 ttl=254 time=1.50 ms 64 bytes from 203.0.113.10: icmp\_seq=3 ttl=254 time=0.624 ms 64 bytes from 203.0.113.10: icmp\_seq=4 ttl=254 time=1.41 ms

--- 203.0.113.10 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3007ms rtt min/avg/max/mdev = 0.624/1.087/1.509/0.381 ms   
2. Confirm the link type has been set to GRE:

vyos@vyos:~$ show interfaces tunnel tun100   
tun100@NONE: <POINTOPOINT,NOARP,UP,LOWER\_UP> mtu 1476 qdisc noqueue state UNKNOWN group default qlen 1000  
 link/gre 198.51.100.2 peer 203.0.113.10  
 inet 10.0.0.1/30 brd 10.0.0.3 scope global tun100  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::5efe:c612:2/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 2183 27 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions 836 9 0 0 0 0   
3. Confirm IP connectivity across the tunnel:

vyos@vyos:~$ ping 10.0.0.2 interface 10.0.0.1 count 4   
PING 10.0.0.2 (10.0.0.2) from 10.0.0.1 : 56(84) bytes of data.

64 bytes from 10.0.0.2: icmp\_seq=1 ttl=255 time=1.05 ms 64 bytes from 10.0.0.2: icmp\_seq=2 ttl=255 time=1.88 ms 64 bytes from 10.0.0.2: icmp\_seq=3 ttl=255 time=1.98 ms 64 bytes from 10.0.0.2: icmp\_seq=4 ttl=255 time=1.98 ms

--- 10.0.0.2 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3008ms rtt min/avg/max/mdev = 1.055/1.729/1.989/0.395 ms

SSTP Client  
SSTP is a form of VTP tunnel that provides a mechanism to transport PPP traffic through an SSL/TLS channel. SSL/TLS provides transport-level security with key negotiation, encryption and traffic integrity checking. The use of SSL/TLS over TCP port 443 (by default, port can be changed) allows SSTP to pass through virtually all firewalls and proxy servers except for authenticated web proxies.

Note

VyOS also comes with a build in SSTP server, see SSTP Server.

Configuration  
Common interface configuration  
 set interfaces sstpc <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:

set interfaces sstpc sstpc0 description 'This is an awesome interface running on VyOS' set interfaces sstpc <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:

set interfaces sstpc sstpc0 disable  
 set interfaces sstpc <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:

set interfaces sstpc sstpc0 mtu 1600  
 set interfaces sstpc <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces sstpc sstpc0 vrf red   
SSTP Client Options  
 set interfaces sstpc <interface> no-default-route   
Only request an address from the SSTP server but do not install any default route.

Example:   
set interfaces sstpc sstpc0 no-default-route   
Note   
This command got added in VyOS 1.4 and inverts the logic from the old default-route CLI option.

set interfaces sstpc <interface> default-route-distance <distance> Set the distance for the default gateway sent by the SSTP server.

Example:   
set interfaces sstpc sstpc0 default-route-distance 220  
 set interfaces sstpc <interface> no-peer-dns   
Use this command to not install advertised DNS nameservers into the local system.

set interfaces sstpc <interface> server <address>   
SSTP remote server to connect to. Can be either an IP address or FQDN.

set interfaces sstpc <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>

Hint

MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces sstpc <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces sstpc <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation

Operation  
 show interfaces sstpc <interface>   
Show detailed information on given <interface>

vyos@vyos:~$ show interfaces sstpc sstpc10   
sstpc10: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UNKNOWN group default qlen 3  
 link/ppp  
 inet 192.0.2.5 peer 192.0.2.254/32 scope global sstpc10  
 valid\_lft forever preferred\_lft forever  
 inet6 fe80::fd53:c7ff:fe8b:144f/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 215 9 0 0 0 0

TX: bytes packets errors dropped carrier collisions  
 539 14 0 0 0 0   
Connect/Disconnect  
 disconnect interface <interface>   
Test disconnecting given connection-oriented interface. <interface> can be sstpc0 as the example.

connect interface <interface>   
Test connecting given connection-oriented interface. <interface> can be sstpc0 as the example.

MACVLAN - Pseudo Ethernet  
Pseudo-Ethernet or MACVLAN interfaces can be seen as subinterfaces to regular ethernet interfaces. Each and every subinterface is created a different media access control (MAC) address, for a single physical Ethernet port. Pseudo- Ethernet interfaces have most of their application in virtualized environments,

By using Pseudo-Ethernet interfaces there will be less system overhead compared to running a traditional bridging approach. Pseudo-Ethernet interfaces can also be used to workaround the general limit of 4096 virtual LANs (VLANs) per physical Ethernet port, since that limit is with respect to a single MAC address.

Every Virtual Ethernet interfaces behaves like a real Ethernet interface. They can have IPv4/IPv6 addresses configured, or can request addresses by DHCP/ DHCPv6 and are associated/mapped with a real ethernet port. This also makes Pseudo-Ethernet interfaces interesting for testing purposes. A Pseudo-Ethernet device will inherit characteristics (speed, duplex, …) from its physical parent (the so called link) interface.

Once created in the system, Pseudo-Ethernet interfaces can be referenced in the exact same way as other Ethernet interfaces. Notes about using Pseudo- Ethernet interfaces:

Pseudo-Ethernet interfaces can not be reached from your internal host. This means that you can not try to ping a Pseudo-Ethernet interface from the host system on which it is defined. The ping will be lost.

Loopbacks occurs at the IP level the same way as for other interfaces, ethernet frames are not forwarded between Pseudo-Ethernet interfaces.

Pseudo-Ethernet interfaces may not work in environments which expect a NIC to only have a single address. This applies to: - VMware machines using default settings - Network switches with security settings allowing only a single MAC address - xDSL modems that try to learn the MAC address of the NIC

Configuration  
Common interface configuration

set interfaces pseudo-ethernet <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces pseudo-ethernet peth0 address 192.0.2.1/24   
set interfaces pseudo-ethernet peth0 address 2001:db8::1/64   
set interfaces pseudo-ethernet peth0 address dhcp   
set interfaces pseudo-ethernet peth0 address dhcpv6  
 set interfaces pseudo-ethernet <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces pseudo-ethernet peth0 description 'This is an awesome interface running on VyOS'  
 set interfaces pseudo-ethernet <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces pseudo-ethernet peth0 disable  
 set interfaces pseudo-ethernet <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces pseudo-ethernet peth0 disable-flow-control  
 set interfaces pseudo-ethernet <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces pseudo-ethernet peth0 disable-link-detect  
 set interfaces pseudo-ethernet <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces pseudo-ethernet peth0 mac '00:01:02:03:04:05'  
 set interfaces pseudo-ethernet <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces pseudo-ethernet peth0 mtu 1600  
 set interfaces pseudo-ethernet <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>

Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pseudo-ethernet <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces pseudo-ethernet peth0 ip arp-cache-timeout 180  
 set interfaces pseudo-ethernet <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces pseudo-ethernet peth0 ip disable-arp-filter  
 set interfaces pseudo-ethernet <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces pseudo-ethernet peth0 ip disable-forwarding  
 set interfaces pseudo-ethernet <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces pseudo-ethernet peth0 ip enable-directed-broadcast  
 set interfaces pseudo-ethernet <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces pseudo-ethernet peth0 ip enable-arp-accept  
 set interfaces pseudo-ethernet <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces pseudo-ethernet peth0 ip enable-arp-announce  
 set interfaces pseudo-ethernet <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces pseudo-ethernet peth0 ip enable-arp-ignore  
 set interfaces pseudo-ethernet <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for

destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces pseudo-ethernet peth0 ip enable-proxy-arp  
 set interfaces pseudo-ethernet <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces pseudo-ethernet <interface> ip source-validation <strict | loose | disable> Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces pseudo-ethernet <interface> ipv6 address autoconf

SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces pseudo-ethernet peth0 ipv6 address autoconf  
 set interfaces pseudo-ethernet <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces pseudo-ethernet peth0 ipv6 address eui64 2001:db8:beef::/64 set interfaces pseudo-ethernet <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces pseudo-ethernet peth0 ipv6 address no-default-link-local  
 set interfaces pseudo-ethernet <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces pseudo-ethernet peth0 ipv6 disable-forwarding  
 set interfaces pseudo-ethernet <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint

MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pseudo-ethernet <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces pseudo-ethernet peth0 ipv6 accept-dad 2  
 set interfaces pseudo-ethernet <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces pseudo-ethernet peth0 ipv6 dup-addr-detect-transmits 5 set interfaces pseudo-ethernet <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces pseudo-ethernet peth0 vrf red   
DHCP(v6)  
 set interfaces pseudo-ethernet <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client

identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces pseudo-ethernet peth0 dhcp-options client-id 'foo-bar'  
 set interfaces pseudo-ethernet <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces pseudo-ethernet peth0 dhcp-options host-name 'VyOS'  
 set interfaces pseudo-ethernet <interface> dhcp-options vendor-class-id <vendor-id> The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces pseudo-ethernet peth0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces pseudo-ethernet <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces pseudo-ethernet peth0 dhcp-options no-default-route  
 set interfaces pseudo-ethernet <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces pseudo-ethernet peth0 dhcp-options default-route-distance 220  
 set interfaces pseudo-ethernet <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24   
Example:   
set interfaces pseudo-ethernet peth0 dhcp-options reject 192.168.100.0/24  
 set interfaces pseudo-ethernet <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual

length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces pseudo-ethernet peth0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d' set interfaces pseudo-ethernet <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces pseudo-ethernet peth0 dhcpv6-options no-release  
 set interfaces pseudo-ethernet <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces pseudo-ethernet peth0 dhcpv6-options parameters-only  
 set interfaces pseudo-ethernet <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces pseudo-ethernet peth0 dhcpv6-options rapid-commit  
 set interfaces pseudo-ethernet <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces pseudo-ethernet peth0 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces pseudo-ethernet <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces pseudo-ethernet peth0 dhcpv6-options pd 0 length 56  
 set interfaces pseudo-ethernet <interface> dhcpv6-options pd <id> interface <delegatee> address <address>

Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces pseudo-ethernet peth0 dhcpv6-options pd 0 interface eth8 address 65534 set interfaces pseudo-ethernet <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces pseudo-ethernet peth0 dhcpv6-options pd 0 interface eth8 sla-id 1   
Pseudo Ethernet/MACVLAN options  
 set interfaces pseudo-ethernet <interface> source-interface <ethX>   
Specifies the physical <ethX> Ethernet interface associated with a Pseudo Ethernet <interface>.

VLAN  
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces pseudo-ethernet <interface> vif <vlan-id>   
Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note   
Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces pseudo-ethernet <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
dhcp interface address is received by DHCP from a DHCP server on this segment.

dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
/config/scripts/dhcp-client/pre-hooks.d/   
/config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces pseudo-ethernet peth0 vif 10 address 192.0.2.1/24   
set interfaces pseudo-ethernet peth0 vif 10 address 2001:db8::1/64   
set interfaces pseudo-ethernet peth0 vif 10 address dhcp   
set interfaces pseudo-ethernet peth0 vif 10 address dhcpv6  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:

set interfaces pseudo-ethernet peth0 vif 10 description 'This is an awesome interface running on VyOS'  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 disable  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 disable-link-detect  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 mtu 1600  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint

MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pseudo-ethernet <interface> vif <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ip arp-cache-timeout 180  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ip disable-arp-filter  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces pseudo-ethernet peth0 vif 10 ip disable-forwarding  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip enable-directed-broadcast Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces pseudo-ethernet peth0 vif 10 ip enable-directed-broadcast  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces pseudo-ethernet peth0 vif 10 ip enable-arp-accept  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces pseudo-ethernet peth0 vif 10 ip enable-arp-announce  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:

If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces pseudo-ethernet peth0 vif 10 ip enable-arp-ignore  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ip enable-proxy-arp  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
In RFC 3069 it is called VLAN Aggregation   
Cisco and Allied Telesyn call it Private VLAN   
Hewlett-Packard call it Source-Port filtering or port-isolation   
Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 address autoconf

SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 address autoconf  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 address eui64 <prefix> EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 address eui64 2001:db8:beef::/64 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 address no-default-link-local  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 disable-forwarding  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>

Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD   
1: Enable DAD (default)   
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 accept-dad 2  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces pseudo-ethernet peth0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces pseudo-ethernet <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 vrf red   
DHCP(v6)  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options client-id <description>

RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options host-name <hostname> Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 dhcp-options vendor-class-id 'VyOS' set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 dhcp-options no-default-route  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options default-route-distance <distance>   
Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces pseudo-ethernet peth0 vif 10 dhcp-options default-route-distance 220  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcp-options reject <address> Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:

set interfaces pseudo-ethernet peth0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces pseudo-ethernet peth0 vif 10 duid   
'0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options no-release  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options parameters-only This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options parameters-only  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options rapid-commit When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options rapid-commit  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options temporary Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options temporary DHCPv6 Prefix Delegation (PD)

VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation.

The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534 set interfaces pseudo-ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces pseudo-ethernet peth0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1

PPPoE  
PPPoE is a network protocol for encapsulating PPP frames inside Ethernet frames. It appeared in 1999, in the context of the boom of DSL as the solution for tunneling packets over the DSL connection to the ISPs IP network, and from there to the rest of the Internet. A 2005 networking book noted that “Most DSL providers use PPPoE, which provides authentication, encryption, and compression.” Typical use of PPPoE involves leveraging the PPP facilities for authenticating the user with a username and password, predominately via the PAP protocol and less often via CHAP.

Operating Modes  
VyOS supports setting up PPPoE in two different ways to a PPPoE internet connection. This is because most ISPs provide a modem that is also a wireless router.

Home Users  
In this method, the DSL Modem/Router connects to the ISP for you with your credentials preprogrammed into the device. This gives you an RFC 1918 address, such as 192.168.1.0/24 by default.

For a simple home network using just the ISP’s equipment, this is usually desirable. But if you want to run VyOS as your firewall and router, this will result in having a double NAT and firewall setup. This results in a few extra layers of complexity, particularly if you use some NAT or tunnel features.

Business Users  
In order to have full control and make use of multiple static public IP addresses, your VyOS will have to initiate the PPPoE connection and control it. In order for this method to work, you will have to figure out how to make your DSL Modem/Router switch into a Bridged Mode so it only acts as a DSL Transceiver device to connect between the Ethernet link of your VyOS and the phone cable. Once your DSL Transceiver is in Bridge Mode, you should get no IP address from it.

Please make sure you connect to the Ethernet Port 1 if your DSL Transceiver has a switch, as some of them only work this way.

Once you have an Ethernet device connected, i.e. eth0, then you can configure it to open the PPPoE session for you and your DSL Transceiver (Modem/Router) just acts to translate your messages in a way that vDSL/aDSL understands.

Configuration  
Common interface configuration  
 set interfaces pppoe <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:

set interfaces pppoe pppoe0 description 'This is an awesome interface running on VyOS' set interfaces pppoe <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:

set interfaces pppoe pppoe0 disable  
 set interfaces pppoe <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:

set interfaces pppoe pppoe0 mtu 1600  
 set interfaces pppoe <interface> vrf <vrf>   
Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces pppoe pppoe0 vrf red   
PPPoE options  
 set interfaces pppoe <interface> access-concentrator <name>   
Use this command to restrict the PPPoE session on a given access concentrator. Normally, a host sends a PPPoE initiation packet to start the PPPoE discovery process, a number of access concentrators respond with offer packets and the host selects one of the responding access concentrators to serve this session.

This command allows you to select a specific access concentrator when you know the access concentrators <name>.

set interfaces pppoe <interface> authentication username <username>   
Use this command to set the username for authenticating with a remote PPPoE endpoint. Authentication is optional from the system’s point of view but most service providers require it.

set interfaces pppoe <interface> authentication password <password>   
Use this command to set the password for authenticating with a remote PPPoE endpoint. Authentication is optional from the system’s point of view but most service providers require it.

set interfaces pppoe <interface> connect-on-demand When set the interface is enabled for “dial-on-demand”.

Use this command to instruct the system to establish a PPPoE connection automatically once traffic passes through the interface. A disabled on-demand connection is established at boot time and remains up. If the link fails for any reason, the link is brought back up immediately.

Enabled on-demand PPPoE connections bring up the link only when traffic needs to pass this link. If the link fails for any reason, the link is brought back up automatically once traffic passes the interface again. If you configure an on-demand PPPoE connection, you must also configure the idle timeout period, after which an idle PPPoE link will be disconnected. A non-zero idle timeout will never disconnect the link after it first came up.

set interfaces pppoe <interface> no-default-route   
Only request an address from the PPPoE server but do not install any default route.

Example:   
set interfaces pppoe pppoe0 no-default-route

Note   
This command got added in VyOS 1.4 and inverts the logic from the old default-route CLI option.

set interfaces pppoe <interface> default-route-distance <distance> Set the distance for the default gateway sent by the PPPoE server.

Example:   
set interfaces pppoe pppoe0 default-route-distance 220  
 set interfaces pppoe <interface> mru <mru>   
Set the MRU to mru. PPPd will ask the peer to send packets of no more than mru bytes. The value of mru must be between 128 and 16384.

A value of 296 works well on very slow links (40 bytes for TCP/IP header + 256 bytes of data). The default is 1492.

Note   
When using the IPv6 protocol, MRU must be at least 1280 bytes.

set interfaces pppoe <interface> idle-timeout <time>   
Use this command to set the idle timeout interval to be used with on-demand PPPoE sessions. When an on-demand connection is established, the link is brought up only when traffic is sent and is disabled when the link is idle for the interval specified.

If this parameter is not set or 0, an on-demand link will not be taken down when it is idle and after the initial establishment of the connection. It will stay up forever.

set interfaces pppoe <interface> holdoff <time>   
Use this command to set re-dial delay time to be used with persist PPPoE sessions. When the PPPoE session is terminated by peer, and on-demand option is not set, the router will attempt to re-establish the PPPoE link.

If this parameter is not set, the default holdoff time is 30 seconds.

set interfaces pppoe <interface> local-address <address>   
Use this command to set the IP address of the local endpoint of a PPPoE session. If it is not set it will be negotiated.

set interfaces pppoe <interface> no-peer-dns   
Use this command to not install advertised DNS nameservers into the local system.

set interfaces pppoe <interface> remote-address <address>   
Use this command to set the IP address of the remote endpoint of a PPPoE session. If it is not set it will be negotiated.

set interfaces pppoe <interface> service-name <name>   
Use this command to specify a service name by which the local PPPoE interface can select access concentrators to connect with. It will connect to any access concentrator if not set.

set interfaces pppoe <interface> source-interface <source-interface>   
Use this command to link the PPPoE connection to a physical interface. Each PPPoE connection must be established over a physical interface. Interfaces can be regular Ethernet interfaces, VIFs or bonding interfaces/VIFs.

set interfaces pppoe <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>

Hint

MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pppoe <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces pppoe <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation   
IPv6  
 set interfaces pppoe <interface> ipv6 address autoconf   
Use this command to enable acquisition of IPv6 address using stateless autoconfig (SLAAC).

set interfaces pppoe <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces pppoe <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces pppoe <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:

set interfaces pppoe pppoe0 dhcpv6-options pd 0 length 56  
 set interfaces pppoe <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces pppoe pppoe0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces pppoe <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces pppoe pppoe0 dhcpv6-options pd 0 interface eth8 sla-id 1 Operation  
 show interfaces pppoe <interface>   
Show detailed information on given <interface>

vyos@vyos:~$ show interfaces pppoe pppoe0   
pppoe0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER\_UP> mtu 1492 qdisc pfifo\_fast state UNKNOWN group default qlen 3  
 link/ppp  
 inet 192.0.2.1 peer 192.0.2.255/32 scope global pppoe0  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 7002658233 5064967 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions 533822843 1620173 0 0 0 0  
 show interfaces pppoe <interface> queue   
Displays queue information for a PPPoE interface.

vyos@vyos:~$ show interfaces pppoe pppoe0 queue

qdisc pfifo\_fast 0: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1  
 Sent 534625359 bytes 1626761 pkt (dropped 62, overlimits 0 requeues 0)  
 backlog 0b 0p requeues 0   
Connect/Disconnect  
 disconnect interface <interface>   
Test disconnecting given connection-oriented interface. <interface> can be pppoe0 as the example.

connect interface <interface>   
Test connecting given connection-oriented interface. <interface> can be pppoe0 as the example.

Example  
Requirements:   
Your ISPs modem is connected to port eth0 of your VyOS box.

No VLAN tagging required by your ISP.

You need your PPPoE credentials from your DSL ISP in order to configure this. The usual username is in the form of name@host.net but may vary depending on ISP.

The largest MTU size you can use with DSL is 1492 due to PPPoE overhead. If you are switching from a DHCP based ISP like cable then be aware that things like VPN links may need to have their MTU sizes adjusted to work within this limit.

With the name-server option set to none, VyOS will ignore the nameservers your ISP sends you and thus you can fully rely on the ones you have configured statically.

Note   
Syntax has changed from VyOS 1.2 (crux) and it will be automatically migrated during an upgrade.

Note   
A default route is automatically installed once the interface is up. To change this behavior use the no-default-route CLI option.

set interfaces pppoe pppoe0 authentication username 'userid'   
set interfaces pppoe pppoe0 authentication password 'secret'   
set interfaces pppoe pppoe0 source-interface 'eth0'   
You should add a firewall to your configuration above as well by assigning it to the pppoe0 itself as shown here:

set firewall interface pppoe0 in name NET-IN   
set firewall interface pppoe0 local name NET-LOCAL   
set firewall interface pppoe0 out name NET-OUT   
VLAN Example  
Some recent ISPs require you to build the PPPoE connection through a VLAN interface. One of those ISPs is e.g. Deutsche Telekom in Germany. VyOS can easily create a PPPoE session through an encapsulated VLAN interface. The following configuration will run your PPPoE connection through VLAN7 which is the default VLAN for Deutsche Telekom:

set interfaces pppoe pppoe0 authentication username 'userid'   
set interfaces pppoe pppoe0 authentication password 'secret'   
set interfaces pppoe pppoe0 source-interface 'eth0.7'   
IPv6 DHCPv6-PD Example  
The following configuration will assign a /64 prefix out of a /56 delegation to eth0. The IPv6 address assigned to eth0 will be <prefix>::ffff/64. If you do not know the prefix size delegated to you, start with sla-len 0.

set interfaces pppoe pppoe0 authentication username vyos   
set interfaces pppoe pppoe0 authentication password vyos   
set interfaces pppoe pppoe0 dhcpv6-options pd 0 interface eth0 address '1' set interfaces pppoe pppoe0 dhcpv6-options pd 0 interface eth0 sla-id '0' set interfaces pppoe pppoe0 dhcpv6-options pd 0 length '56'   
set interfaces pppoe pppoe0 ipv6 address autoconf   
set interfaces pppoe pppoe0 source-interface eth1

WireGuard  
WireGuard is an extremely simple yet fast and modern VPN that utilizes state-of-the-art cryptography. See https://www.wireguard.com for more information.

Site to Site VPN  
This diagram corresponds with the example site to site configuration below.

../../\_images/wireguard\_site2site\_diagram.jpg   
Keypairs  
WireGuard requires the generation of a keypair, which includes a private key to decrypt incoming traffic, and a public key for peer(s) to encrypt traffic.

Generate Keypair  
 generate pki wireguard key-pair   
It generates the keypair, which includes the public and private parts. The key is not stored on the system - only a keypair is generated.

vyos@vyos:~$ generate pki wireguard key-pair

Private key: iJJyEARGK52Ls1GYRCcFvPuTj7WyWYDo//BknoDU0XY=   
Public key: EKY0dxRrSD98QHjfHOK13mZ5PJ7hnddRZt5woB3szyw=  
 generate pki wireguard key-pair install interface <interface>   
Generates a keypair, which includes the public and private parts, and build a configuration command to install this key to interface.

vyos@vyos:~$ generate pki wireguard key-pair install interface wg10 "generate" CLI command executed from operational level.

Generated private-key is not stored to CLI, use configure mode commands to install key:

set interfaces wireguard wg10 private-key   
'4Krkv8h6NkAYMMaBWI957yYDJDMvj9URTHstdlOcDU0='

Corresponding public-key to use on peer system is: 'UxDsYT6EnpTIOKUzvMlw2p0sNOKQvFxEdSVrnNrX1Ro=' Note

If this command is invoked from configure mode with the run prefix the key is automatically installed to the appropriate interface:

vyos@vyos# run generate pki wireguard key-pair install interface wg10 "generate" CLI command executed from config session.   
Generated private-key was imported to CLI!

Use the following command to verify: show interfaces wireguard wg10 Corresponding public-key to use on peer system is:   
'7d9KwabjLhHpJiEJeIGd0CBlao/eTwFOh6xyCovTfG8='

vyos@vyos# compare   
[edit interfaces]   
+wireguard wg10 {   
+ private-key CJweb8FC6BU3Loj4PC2pn5V82cDjIPs7G1saW0ZfLWc= +}  
 show interfaces wireguard <interface> public-key   
Retrieve public key portion from configured WIreGuard interface.

vyos@vyos:~$ show interfaces wireguard wg01 public-key   
EKY0dxRrSD98QHjfHOK13mZ5PJ7hnddRZt5woB3szyw=   
Optional  
 generate pki wireguard preshared-key   
An additional layer of symmetric-key crypto can be used on top of the asymmetric crypto.

This is optional.

vyos@vyos:~$ generate pki wireguard preshared-key   
Pre-shared key: OHH2EwZfMNK+1L6BXbYw3bKCtMrfjpR4mCAEeBlFnRs=  
 generate pki wireguard preshared-key install interface <interface> peer <peer>   
An additional layer of symmetric-key crypto can be used on top of the asymmetric crypto. This command automatically creates for you the required CLI command to install this PSK for a given peer.

This is optional.

vyos@vyos:~$ generate pki wireguard preshared-key install interface wg10 peer foo "generate" CLI command executed from operational level.

Generated preshared-key is not stored to CLI, use configure mode commands to install key: set interfaces wireguard wg10 peer foo preshared-key   
'32vQ1w1yFKTna8n7Gu7EimubSe2Y63m8bafz55EG3Ro='   
Pre-shared key: +LuaZ8W6DjsDFJFX3jJzoNqrsXHhvq08JztM9z8LHCs=   
Note   
If this command is invoked from configure mode with the run prefix the key is automatically installed to the appropriate interface:   
Interface configuration  
The next step is to configure your local side as well as the policy based trusted destination addresses. If you only initiate a connection, the listen port and address/port is optional; however, if you act like a server and endpoints initiate the connections to your system, you need to define a port your clients can connect to, otherwise the port is randomly chosen and may make connection difficult with firewall rules, since the port may be different each time the system is rebooted.

You will also need the public key of your peer as well as the network(s) you want to tunnel (allowed-ips) to configure a WireGuard tunnel. The public key below is always the public key from your peer, not your local one.

local side - commands   
WireGuard interface itself uses address 10.1.0.1/30   
We only allow the 192.168.2.0/24 subnet to travel over the tunnel   
Our remote end of the tunnel for peer to-wg02 is reachable at 192.0.2.1 port 51820   
The remote peer to-wg02 uses XMrlPykaxhdAAiSjhtPlvi30NVkvLQliQuKP7AI7CyI= as its public key portion

We listen on port 51820   
We route all traffic for the 192.168.2.0/24 network to interface wg01   
set interfaces wireguard wg01 address '10.1.0.1/30'   
set interfaces wireguard wg01 description 'VPN-to-wg02'   
set interfaces wireguard wg01 peer to-wg02 allowed-ips '192.168.2.0/24'   
set interfaces wireguard wg01 peer to-wg02 address '192.0.2.1'   
set interfaces wireguard wg01 peer to-wg02 port '51820'   
set interfaces wireguard wg01 peer to-wg02 public-key   
'XMrlPykaxhdAAiSjhtPlvi30NVkvLQliQuKP7AI7CyI='   
set interfaces wireguard wg01 port '51820'   
set protocols static route 192.168.2.0/24 interface wg01   
The last step is to define an interface route for 192.168.2.0/24 to get through the WireGuard interface wg01. Multiple IPs or networks can be defined and routed. The last check is allowed-ips which either prevents or allows the traffic.

Warning   
You can not assign the same allowed-ips statement to multiple WireGuard peers. This a design decision. For more information please check the WireGuard mailing list.

set interfaces wireguard <interface> private-key <private-key>   
Associates the previously generated private key to a specific WireGuard interface. The private key can be generate via the command   
generate pki wireguard key-pair.

set interfaces wireguard wg01 private-key   
'iJJyEARGK52Ls1GYRCcFvPuTj7WyWYDo//BknoDU0XY='   
The command show interfaces wireguard wg01 public-key will then show the public key, which needs to be shared with the peer.

set interfaces wireguard <interface> per-client-thread   
Provides a per-device control to enable/disable the threaded mode for all the NAPI instances of the given network device, without the need for a device up/down.

If CLI option is not specified, this feature is disabled.

Example:   
set interfaces wireguard wg01 per-client-thread

remote side - commands

set interfaces wireguard wg01 address '10.1.0.2/30'   
set interfaces wireguard wg01 description 'VPN-to-wg01'   
set interfaces wireguard wg01 peer to-wg01 allowed-ips '192.168.1.0/24' set interfaces wireguard wg01 peer to-wg01 address '192.0.2.2'   
set interfaces wireguard wg01 peer to-wg01 port '51820'   
set interfaces wireguard wg01 peer to-wg01 public-key   
'EKY0dxRrSD98QHjfHOK13mZ5PJ7hnddRZt5woB3szyw='   
set interfaces wireguard wg01 port '51820'   
set interfaces wireguard wg01 private-key   
'OLTQY3HuK5qWDgVs6fJR093SwPgOmCKkDI1+vJLGoFU='

set protocols static route 192.168.1.0/24 interface wg01   
Firewall Exceptions  
For the WireGuard traffic to pass through the WAN interface, you must create a firewall exception.

set firewall ipv4 name OUTSIDE\_LOCAL rule 10 action accept   
set firewall ipv4 name OUTSIDE\_LOCAL rule 10 description 'Allow established/related'   
set firewall ipv4 name OUTSIDE\_LOCAL rule 10 state established enable   
set firewall ipv4 name OUTSIDE\_LOCAL rule 10 state related enable   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 action accept   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 description WireGuard\_IN   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 destination port 51820   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 log enable   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 protocol udp   
set firewall ipv4 name OUTSIDE\_LOCAL rule 20 source   
You should also ensure that the OUTSIDE\_LOCAL firewall group is applied to the WAN interface and a direction (local).

set interfaces ethernet eth0 firewall local name 'OUTSIDE-LOCAL'   
Assure that your firewall rules allow the traffic, in which case you have a working VPN using WireGuard.

wg01# ping 192.168.1.1   
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

64 bytes from 192.168.1.1: icmp\_seq=1 ttl=64 time=1.16 ms 64 bytes from 192.168.1.1: icmp\_seq=2 ttl=64 time=1.77 ms

wg02# ping 192.168.2.1   
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.

64 bytes from 192.168.2.1: icmp\_seq=1 ttl=64 time=4.40 ms 64 bytes from 192.168.2.1: icmp\_seq=2 ttl=64 time=1.02 ms

An additional layer of symmetric-key crypto can be used on top of the asymmetric crypto. This is optional.

vyos@vyos:~$ generate pki wireguard preshared-key   
Pre-shared key: rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc=   
Copy the key, as it is not stored on the local filesystem. Because it is a symmetric key, only you and your peer should have knowledge of its content. Make sure you distribute the key in a safe manner,

wg01# set interfaces wireguard wg01 peer to-wg02 preshared-key   
'rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc='   
wg02# set interfaces wireguard wg01 peer to-wg01 preshared-key   
'rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc='   
Remote Access “RoadWarrior” Example  
With WireGuard, a Road Warrior VPN config is similar to a site-to-site VPN. It just lacks the address and port statements.

In the following example, the IPs for the remote clients are defined in the peers. This allows the peers to interact with one another. In comparison to the site-to-site example the persistent-keepalive flag is set to 15 seconds to assure the connection is kept alive. This is mainly relevant if one of the peers is behind NAT and can’t be connected to if the connection is lost. To be effective this value needs to be lower than the UDP timeout.

wireguard wg01 {  
 address 10.172.24.1/24  
 address 2001:db8:470:22::1/64  
 description RoadWarrior  
 peer MacBook {  
 allowed-ips 10.172.24.30/32  
 allowed-ips 2001:db8:470:22::30/128  
 persistent-keepalive 15  
 pubkey F5MbW7ye7DsoxdOaixjdrudshjjxN5UdNV+pGFHqehc=  
 }  
 peer iPhone {  
 allowed-ips 10.172.24.20/32  
 allowed-ips 2001:db8:470:22::20/128  
 persistent-keepalive 15  
 pubkey BknHcLFo8nOo8Dwq2CjaC/TedchKQ0ebxC7GYn7Al00=  
 }  
 port 2224  
 private-key OLTQY3HuK5qWDgVs6fJR093SwPgOmCKkDI1+vJLGoFU=   
}   
The following is the config for the iPhone peer above. It’s important to note that the AllowedIPs wildcard setting directs all IPv4 and IPv6 traffic through the connection.

[Interface]   
PrivateKey = ARAKLSDJsadlkfjasdfiowqeruriowqeuasdf= Address = 10.172.24.20/24, 2001:db8:470:22::20/64 DNS = 10.0.0.53, 10.0.0.54

[Peer]   
PublicKey = RIbtUTCfgzNjnLNPQ/ulkGnnB2vMWHm7l2H/xUfbyjc=   
AllowedIPs = 0.0.0.0/0, ::/0   
Endpoint = 192.0.2.1:2224   
PersistentKeepalive = 25   
However, split-tunneling can be achieved by specifying the remote subnets. This ensures that only traffic destined for the remote site is sent over the tunnel. All other traffic is unaffected.

[Interface]   
PrivateKey = 8Iasdfweirousd1EVGUk5XsT+wYFZ9mhPnQhmjzaJE6Go= Address = 10.172.24.30/24, 2001:db8:470:22::30/64

[Peer]   
PublicKey = RIbtUTCfgzNjnLNPQ/ulkGnnB2vMWHm7l2H/xUfbyjc=   
AllowedIPs = 10.172.24.30/24, 2001:db8:470:22::/64   
Endpoint = 192.0.2.1:2224   
PersistentKeepalive = 25   
Operational Commands  
Status  
 show interfaces wireguard wg01 summary   
Show info about the Wireguard service. It also shows the latest handshake.

vyos@vyos:~$ show interfaces wireguard wg01 summary interface: wg01  
 public key:  
 private key: (hidden)  
 listening port: 51820

peer: <peer pubkey>  
 endpoint: <peer public IP>  
 allowed ips: 10.69.69.2/32  
 latest handshake: 23 hours, 45 minutes, 26 seconds ago transfer: 1.26 MiB received, 6.47 MiB sent  
 show interfaces wireguard   
Get a list of all wireguard interfaces

Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down Interface IP Address S/L Description

--------- ---------- --- -----------  
wg01 10.0.0.1/24 u/u  
 show interfaces wireguard <interface>   
Show general information about specific WireGuard interface

vyos@vyos:~$ show interfaces wireguard wg01   
interface: wg01  
 address: 10.0.0.1/24  
 public key: h1HkYlSuHdJN6Qv4Hz4bBzjGg5WUty+U1L7DJsZy1iE= private key: (hidden)  
 listening port: 41751

RX: bytes packets errors dropped overrun mcast  
 0 0 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 0 0 0 0 0 0   
Remote Access “RoadWarrior” clients  
Some users tend to connect their mobile devices using WireGuard to their VyOS router. To ease deployment one can generate a “per mobile” configuration from the VyOS CLI.

Warning

From a security perspective, it is not recommended to let a third party create and share the private key for a secured connection. You should create the private portion on your own and only hand out the public key. Please keep this in mind when using this convenience feature.

generate wireguard client-config <name> interface <interface> server <ip|fqdn> address <client-ip>   
Using this command, you will create a new client configuration which can connect to interface on this router. The public key from the specified interface is automatically extracted and embedded into the configuration.

The command also generates a configuration snipped which can be copy/pasted into the VyOS CLI if needed. The supplied <name> on the CLI will become the peer name in the snippet.

In addition you will specify the IP address or FQDN for the client where it will connect to. The address parameter can be used up to two times and is used to assign the clients specific IPv4 (/32) or IPv6 (/128) address.

OpenVPN  
Traditionally hardware routers implement IPsec exclusively due to relative ease of implementing it in hardware and insufficient CPU power for doing encryption in software. Since VyOS is a software router, this is less of a concern. OpenVPN has been widely used on UNIX platform for a

long time and is a popular option for remote access VPN, though it’s also capable of site-to-site connections.

Advantages of OpenVPN are:   
It uses a single TCP or UDP connection and does not rely on packet source addresses, so it will work even through a double NAT: perfect for public hotspots and such   
It’s easy to setup and offers very flexible split tunneling   
There’s a variety of client GUI frontends for any platform   
Disadvantages are:   
It’s slower than IPsec due to higher protocol overhead and the fact it runs in user mode while IPsec, on Linux, is in kernel mode   
None of the operating systems have client software installed by default   
In the VyOS CLI, a key point often overlooked is that rather than being configured using the set vpn stanza, OpenVPN is configured as a network interface using set interfaces openvpn.

Site-to-Site  
../../\_images/openvpn\_site2site\_diagram.jpg   
OpenVPN is popular for client-server setups, but its site-to-site mode remains a relatively obscure feature, and many router appliances still don’t support it. However, it’s very useful for quickly setting up tunnels between routers.

As of VyOS 1.4, OpenVPN site-to-site mode can use either pre-shared keys or x.509 certificates. The pre-shared key mode is deprecated and will be removed from future OpenVPN versions, so VyOS will have to remove support for that option as well. The reason is that using pre-shared keys is significantly less secure than using TLS.

We’ll configure OpenVPN using self-signed certificates, and then discuss the legacy pre-shared key mode.

In both cases, we will use the following settings:   
The public IP address of the local side of the VPN will be 198.51.100.10.

The public IP address of the remote side of the VPN will be 203.0.113.11.

The tunnel will use 10.255.1.1 for the local IP and 10.255.1.2 for the remote.

The local site will have a subnet of 10.0.0.0/16.

The remote site will have a subnet of 10.1.0.0/16.

The official port for OpenVPN is 1194, which we reserve for client VPN; we will use 1195 for site-to-site VPN.

The persistent-tunnel directive will allow us to configure tunnel-related attributes, such as firewall policy as we would on any normal network interface.

If known, the IP of the remote router can be configured using the remote-host directive; if unknown, it can be omitted. We will assume a dynamic IP for our remote router.

Setting up certificates  
Setting up a full-blown PKI with a CA certificate would arguably defeat the purpose of site-to-site OpenVPN, since its main goal is supposed to be configuration simplicity, compared to server setups that need to support multiple clients.

However, since VyOS 1.4, it is possible to verify self-signed certificates using certificate fingerprints.

On both sides, you need to generate a self-signed certificate, preferrably using the “ec” (elliptic curve) type. You can generate them by executing command run generate pki certificate self-signed install <name> in the configuration mode. Once the command is complete, it will add the certificate to the configuration session, to the pki subtree. You can then review the proposed changes and commit them.

vyos@vyos# run generate pki certificate self-signed install openvpn-local   
Enter private key type: [rsa, dsa, ec] (Default: rsa) ec   
Enter private key bits: (Default: 256)   
Enter country code: (Default: GB)   
Enter state: (Default: Some-State)   
Enter locality: (Default: Some-City)   
Enter organization name: (Default: VyOS)   
Enter common name: (Default: vyos.io)   
Do you want to configure Subject Alternative Names? [y/N]   
Enter how many days certificate will be valid: (Default: 365)   
Enter certificate type: (client, server) (Default: server)   
Note: If you plan to use the generated key on this router, do not encrypt the private key.

Do you want to encrypt the private key with a passphrase? [y/N]   
2 value(s) installed. Use "compare" to see the pending changes, and "commit" to apply.

[edit]

vyos@vyos# compare   
[pki]   
+ certificate openvpn-local {   
+ certificate "MIICJTCCAcugAwIBAgIUMXLfRNJ5iOjk/   
uAZqUe4phW8MdgwCgYIKoZIzj0EAwIwVzELMAkGA1UEBhMCR0IxEzARBgNVBAgMClNvbWUtU3 RhdGUxEjAQBgNVBAcMCVNvbWUtQ2l0eTENMAsGA1UECgwEVnlPUzEQMA4GA1UEAwwHdnlvc y5pbzAeFw0yMzA5MDcyMTQzMTNaFw0yNDA5MDYyMTQzMTNaMFcxCzAJBgNVBAYTAkdCMR MwEQYDVQQIDApTb21lLVN0YXRlMRIwEAYDVQQHDAlTb21lLUNpdHkxDTALBgNVBAoMBFZ5T1 MxEDAOBgNVBAMMB3Z5b3MuaW8wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASp7D0vE3SK SAWAzr/lw9Eq9Q89r247AJR6ec/GT26AIcVA1bsongV1YaWvRwzTPC/yi5pkzV/PcT/   
WU7JQIyMWo3UwczAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/   
wQEAwIHgDATBgNVHSUEDDAKBggrBgEFBQcDATAdBgNVHQ4EFgQUBrAxRdFppdG/   
UBRdo7qNyHutaTQwHwYDVR0jBBgwFoAUBrAxRdFppdG/   
UBRdo7qNyHutaTQwCgYIKoZIzj0EAwIDSAAwRQIhAI2+8C92z9wTcTWkQ/   
goRxs10EBC+h78O+vgo9k97z5iAiBSeqfaVr5taQTS31+McGTAK3cYWNTg0DlOBI8aKO2oRg==" + private {   
+ key   
"MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgtOeEb0dMb5P/2Exi09WW vk6Cvz0oOBoDuP68ZimS2LShRANCAASp7D0vE3SKSAWAzr/lw9Eq9Q89r247AJR6ec/   
GT26AIcVA1bsongV1YaWvRwzTPC/yi5pkzV/PcT/WU7JQIyMW"   
+ }   
+ }

[edit]

vyos@vyos# commit   
You do not need to copy the certificate to the other router. Instead, you need to retrieve its SHA-256 fingerprint. OpenVPN only supports SHA-256 fingerprints at the moment, so you need to use the following command:

vyos@vyos# run show pki certificate openvpn-local fingerprint sha256   
5C:B8:09:64:8B:59:51:DC:F4:DF:2C:12:5C:B7:03:D1:68:94:D7:5B:62:C2:E1:83:79:F1:F0:68:B2:81: 26:79   
Note: certificate names don’t matter, we use ‘openvpn-local’ and ‘openvpn-remote’ but they can be arbitrary.

Repeat the procedure on the other router.

Setting up OpenVPN  
Local Configuration:

Configure the tunnel:

set interfaces openvpn vtun1 mode site-to-site

set interfaces openvpn vtun1 protocol udp   
set interfaces openvpn vtun1 persistent-tunnel   
set interfaces openvpn vtun1 remote-host '203.0.113.11' # Public IP of the other side   
set interfaces openvpn vtun1 local-port '1195'   
set interfaces openvpn vtun1 remote-port '1195'   
set interfaces openvpn vtun1 local-address '10.255.1.1' # Local IP of vtun interface set interfaces openvpn vtun1 remote-address '10.255.1.2' # Remote IP of vtun interface   
set interfaces openvpn vtun1 tls certificate 'openvpn-local' # The self-signed certificate   
set interfaces openvpn vtun1 tls peer-fingerprint <remote cert fingerprint> # The output of 'run show pki certificate <name> fingerprint sha256  
 on the remote rout   
Remote Configuration:

set interfaces openvpn vtun1 mode site-to-site   
set interfaces openvpn vtun1 protocol udp   
set interfaces openvpn vtun1 persistent-tunnel   
set interfaces openvpn vtun1 remote-host '198.51.100.10' # Pub IP of other site set interfaces openvpn vtun1 local-port '1195'   
set interfaces openvpn vtun1 remote-port '1195'   
set interfaces openvpn vtun1 local-address '10.255.1.2' # Local IP of vtun interface set interfaces openvpn vtun1 remote-address '10.255.1.1' # Remote IP of vtun interface   
set interfaces openvpn vtun1 tls certificate 'openvpn-remote' # The self-signed certificate   
set interfaces openvpn vtun1 tls peer-fingerprint <local cert fingerprint> # The output of 'run show pki certificate <name> fingerprint sha256  
 on the local router   
Pre-shared keys  
Until VyOS 1.4, the only option for site-to-site OpenVPN without PKI was to use pre-shared keys.

That option is still available but it is deprecated and will be removed in the future. However, if you need to set up a tunnel to an older VyOS version or a system with older OpenVPN, you need to still need to know how to use it.

First, you need to generate a key by running run generate pki openvpn shared-secret install <name> from configuration mode. You can use any name, we will use s2s.

vyos@local# run generate pki openvpn shared-secret install s2s   
2 value(s) installed. Use "compare" to see the pending changes, and "commit" to apply.

[edit]   
vyos@local# compare   
[pki openvpn shared-secret]

+ s2s {   
+ key   
"7c73046a9da91e874d31c7ad894a32688cda054bde157c64270f28eceebc0bb2f44dbb70335fad 45148b0456aaa78cb34a34c0958eeed4f75e75fd99ff519ef940f7029a316c436d2366a2b0fb8ea1 d1c792a65f67d10a461af83ef4530adc25d1c872de6d9c7d5f338223d1f3b66dc3311bbbddc0e05 228c47b91c817c721aadc7ed18f0662df52ad14f898904372679e3d9697d062b0869d12de47ceb 2e626fa12e1926a3119be37dd29c9b0ad81997230f4038926900d5edb78522d2940cfe207f8e2b 948e0d459fa137ebb18064ac5982b28dd1899020b4f2b082a20d5d4eb65710fbb1e62b5e061df3 9620267eab429d3eedd9a1ae85957457c8e4655f3"   
+ version "1"   
+ }

[edit]

vyos@local# commit   
[edit]   
Then you need to install the key on the remote router:

vyos@remote# set pki openvpn shared-secret s2s key <generated key string> Then you need to set the key in your OpenVPN interface settings:

set interfaces openvpn vtun1 shared-secret-key s2s   
Firewall Exceptions  
For the OpenVPN traffic to pass through the WAN interface, you must create a firewall exception.

set firewall name OUTSIDE\_LOCAL rule 10 action accept   
set firewall name OUTSIDE\_LOCAL rule 10 description 'Allow established/related'   
set firewall name OUTSIDE\_LOCAL rule 10 state established enable   
set firewall name OUTSIDE\_LOCAL rule 10 state related enable   
set firewall name OUTSIDE\_LOCAL rule 20 action accept   
set firewall name OUTSIDE\_LOCAL rule 20 description OpenVPN\_IN   
set firewall name OUTSIDE\_LOCAL rule 20 destination port 1195   
set firewall name OUTSIDE\_LOCAL rule 20 log enable   
set firewall name OUTSIDE\_LOCAL rule 20 protocol udp   
set firewall name OUTSIDE\_LOCAL rule 20 source   
You should also ensure that the OUTISDE\_LOCAL firewall group is applied to the WAN interface and a direction (local).

set firewall interface eth0 local name 'OUTSIDE-LOCAL' Static Routing:

Static routes can be configured referencing the tunnel interface; for example, the local router will use a network of 10.0.0.0/16, while the remote has a network of 10.1.0.0/16:

Local Configuration:

set protocols static route 10.1.0.0/16 interface vtun1 Remote Configuration:

set protocols static route 10.0.0.0/16 interface vtun1   
The configurations above will default to using 256-bit AES in GCM mode for encryption (if both sides support NCP) and SHA-1 for HMAC authentication. SHA-1 is considered weak, but other hashing algorithms are available, as are encryption algorithms:

For Encryption:

This sets the cipher when NCP (Negotiable Crypto Parameters) is disabled or OpenVPN version < 2.4.0.

vyos@vyos# set interfaces openvpn vtun1 encryption cipher   
Possible completions:  
 des DES algorithm  
 3des DES algorithm with triple encryption  
 bf128 Blowfish algorithm with 128-bit key  
 bf256 Blowfish algorithm with 256-bit key  
 aes128 AES algorithm with 128-bit key CBC  
 aes128gcm AES algorithm with 128-bit key GCM  
 aes192 AES algorithm with 192-bit key CBC  
 aes192gcm AES algorithm with 192-bit key GCM  
 aes256 AES algorithm with 256-bit key CBC  
 aes256gcm AES algorithm with 256-bit key GCM   
This sets the accepted ciphers to use when version => 2.4.0 and NCP is enabled (which is the default). Default NCP cipher for versions >= 2.4.0 is aes256gcm. The first cipher in this list is what server pushes to clients.

vyos@vyos# set int open vtun0 encryption ncp-ciphers Possible completions:  
 des DES algorithm  
 3des DES algorithm with triple encryption  
 aes128 AES algorithm with 128-bit key CBC  
 aes128gcm AES algorithm with 128-bit key GCM aes192 AES algorithm with 192-bit key CBC  
 aes192gcm AES algorithm with 192-bit key GCM aes256 AES algorithm with 256-bit key CBC  
 aes256gcm AES algorithm with 256-bit key GCM For Hashing:

vyos@vyos# set interfaces openvpn vtun1 hash   
Possible completions:  
 md5 MD5 algorithm  
 sha1 SHA-1 algorithm  
 sha256 SHA-256 algorithm  
 sha512 SHA-512 algorithm   
If you change the default encryption and hashing algorithms, be sure that the local and remote ends have matching configurations, otherwise the tunnel will not come up.

Firewall policy can also be applied to the tunnel interface for local, in, and out directions and functions identically to ethernet interfaces.

If making use of multiple tunnels, OpenVPN must have a way to distinguish between different tunnels aside from the pre-shared-key. This is either by referencing IP address or port number.

One option is to dedicate a public IP to each tunnel. Another option is to dedicate a port number to each tunnel (e.g. 1195,1196,1197…).

OpenVPN status can be verified using the show openvpn operational commands. See the built-in help for a complete list of options.

Server  
Multi-client server is the most popular OpenVPN mode on routers. It always uses x.509 authentication and therefore requires a PKI setup. Refer this topic PKI to generate a CA certificate, a server certificate and key, a certificate revocation list, a Diffie-Hellman key exchange parameters file. You do not need client certificates and keys for the server setup.

In this example we will use the most complicated case: a setup where each client is a router that has its own subnet (think HQ and branch offices), since simpler setups are subsets of it.

Suppose you want to use 10.23.1.0/24 network for client tunnel endpoints and all client subnets belong to 10.23.0.0/20. All clients need access to the 192.168.0.0/16 network.

First we need to specify the basic settings. 1194/UDP is the default. The persistent-tunnel option is recommended, it prevents the TUN/TAP device from closing on connection resets or daemon reloads.

Note

Using openvpn-option -reneg-sec can be tricky. This option is used to renegotiate data channel after n seconds. When used at both server and client, the lower value will trigger the   
renegotiation. If you set it to 0 on one side of the connection (to disable it), the chosen value on the other side will determine when the renegotiation will occur.

set interfaces openvpn vtun10 mode server

set interfaces openvpn vtun10 local-port 1194   
set interfaces openvpn vtun10 persistent-tunnel   
set interfaces openvpn vtun10 protocol udp   
Then we need to generate, add and specify the names of the cryptographic materials. Each of the install command should be applied to the configuration and commited before using under the openvpn interface configuration.

run generate pki ca install ca-1 # Follow the instructions to generate CA cert.

Configure mode commands to install:   
set pki ca ca-1 certificate 'generated\_cert\_string' set pki ca ca-1 private key 'generated\_private\_key'

run generate pki certificate sign ca-1 install srv-1 # Follow the instructions to generate server cert.

Configure mode commands to install:   
set pki certificate srv-1 certificate 'generated\_server\_cert' set pki certificate srv-1 private key 'generated\_private\_key'

run generate pki dh install dh-1 # Follow the instructions to generate set of Diffie-Hellman parameters.

Generating parameters...

Configure mode commands to install DH parameters: set pki dh dh-1 parameters 'generated\_dh\_params\_set'

set interfaces openvpn vtun10 tls ca-certificate ca-1   
set interfaces openvpn vtun10 tls certificate srv-1   
set interfaces openvpn vtun10 tls dh-params dh-1   
Now we need to specify the server network settings. In all cases we need to specify the subnet for client tunnel endpoints. Since we want clients to access a specific network behind our router, we will use a push-route option for installing that route on clients.

set interfaces openvpn vtun10 server push-route 192.168.0.0/16   
set interfaces openvpn vtun10 server subnet 10.23.1.0/24   
Since it’s a HQ and branch offices setup, we will want all clients to have fixed addresses and we will route traffic to specific subnets through them. We need configuration for each client to achieve this.

Note

Clients are identified by the CN field of their x.509 certificates, in this example the CN is client0:

set interfaces openvpn vtun10 server client client0 ip 10.23.1.10   
set interfaces openvpn vtun10 server client client0 subnet 10.23.2.0/25

OpenVPN will not automatically create routes in the kernel for client subnets when they connect and will only use client-subnet association internally, so we need to create a route to the 10.23.0.0/20 network ourselves:

set protocols static route 10.23.0.0/20 interface vtun10   
Additionally, each client needs a copy of ca cert and its own client key and cert files. The files are plaintext so they may be copied either manually from the CLI. Client key and cert files should be signed with the proper ca cert and generated on the server side.

HQ’s router requires the following steps to generate crypto materials for the Branch 1:

run generate pki certificate sign ca-1 install branch-1 # Follow the instructions to generate client  
 cert for Branch 1   
Configure mode commands to install:   
Branch 1’s router might have the following lines:

set pki ca ca-1 certificate 'generated\_cert\_string' # CA cert generated on HQ router set pki certificate branch-1 certificate 'generated\_branch\_cert' # Client cert generated and signed on HQ router   
set pki certificate branch-1 private key 'generated\_private\_key' # Client cert key generated on HQ router

set interfaces openvpn vtun10 tls ca-cert ca-1   
set interfaces openvpn vtun10 tls certificate branch-1   
LDAP  
Enterprise installations usually ship a kind of directory service which is used to have a single password store for all employees. VyOS and OpenVPN support using LDAP/AD as single user backend.

Authentication is done by using the openvpn-auth-ldap.so plugin which is shipped with every VyOS installation. A dedicated configuration file is required. It is best practise to store it in /config to survive image updates

set interfaces openvpn vtun0 openvpn-option "--plugin /usr/lib/openvpn/openvpn-auth-ldap.so /config/auth/ldap-auth.config"   
The required config file may look like this:

<LDAP>   
# LDAP server URL   
URL ldap://ldap.example.com   
# Bind DN (If your LDAP server doesn't support anonymous binds) BindDN cn=LDAPUser,dc=example,dc=com   
# Bind Password password

Password S3cr3t   
# Network timeout (in seconds)   
Timeout 15   
</LDAP>

<Authorization>   
# Base DN   
BaseDN "ou=people,dc=example,dc=com"   
# User Search Filter   
SearchFilter "(&(uid=%u)(objectClass=shadowAccount))" # Require Group Membership - allow all users   
RequireGroup false   
</Authorization>   
Despite the fact that AD is a superset of LDAP

<LDAP>  
 # LDAP server URL  
 URL ldap://dc01.example.com  
 # Bind DN (If your LDAP server doesn’t support anonymous binds) BindDN CN=LDAPUser,DC=example,DC=com  
 # Bind Password  
 Password mysecretpassword  
 # Network timeout (in seconds)  
 Timeout 15  
 # Enable Start TLS  
 TLSEnable no  
 # Follow LDAP Referrals (anonymously)  
 FollowReferrals no   
</LDAP>

<Authorization>  
 # Base DN  
 BaseDN "DC=example,DC=com"  
 # User Search Filter, user must be a member of the VPN AD group SearchFilter "(&(sAMAccountName=%u)   
(memberOf=CN=VPN,OU=Groups,DC=example,DC=com))"  
 # Require Group Membership  
 RequireGroup false # already handled by SearchFilter  
 <Group>  
 BaseDN "OU=Groups,DC=example,DC=com"  
 SearchFilter "(|(cn=VPN))"  
 MemberAttribute memberOf  
 </Group>   
</Authorization>

If you only want to check if the user account is enabled and can authenticate (against the primary group) the following snipped is sufficient:

<LDAP>  
 URL ldap://dc01.example.com  
 BindDN CN=SA\_OPENVPN,OU=ServiceAccounts,DC=example,DC=com Password ThisIsTopSecret  
 Timeout 15  
 TLSEnable no  
 FollowReferrals no   
</LDAP>

<Authorization>  
 BaseDN "DC=example,DC=com"  
 SearchFilter "sAMAccountName=%u"  
 RequireGroup false   
</Authorization>   
A complete LDAP auth OpenVPN configuration could look like the following example:

vyos@vyos# show interfaces openvpn  
 openvpn vtun0 {  
 mode server  
 openvpn-option "--tun-mtu 1500 --fragment 1300 --mssfix"  
 openvpn-option "--plugin /usr/lib/openvpn/openvpn-auth-ldap.so /config/auth/ldap-auth.config"  
 openvpn-option "--push redirect-gateway"  
 openvpn-option --duplicate-cn  
 openvpn-option --client-cert-not-required  
 openvpn-option --comp-lzo  
 openvpn-option --persist-key  
 openvpn-option --persist-tun  
 server {  
 domain-name example.com  
 max-connections 5  
 name-server 203.0.113.0.10  
 name-server 198.51.100.3  
 subnet 172.18.100.128/29  
 }  
 tls {  
 ca-certificate ca.crt  
 certificate server.crt  
 dh-params dh1024.pem  
 }  
 }

Client  
VyOS can not only act as an OpenVPN site-to-site or server for multiple clients. You can indeed also configure any VyOS OpenVPN interface as an OpenVPN client connecting to a VyOS OpenVPN server or any other OpenVPN server.

Given the following example we have one VyOS router acting as OpenVPN server and another VyOS router acting as OpenVPN client. The server also pushes a static client IP address to the OpenVPN client. Remember, clients are identified using their CN attribute in the SSL certificate.

Server Side  
set interfaces openvpn vtun10 encryption cipher 'aes256'   
set interfaces openvpn vtun10 hash 'sha512'   
set interfaces openvpn vtun10 local-host '172.18.201.10'   
set interfaces openvpn vtun10 local-port '1194'   
set interfaces openvpn vtun10 mode 'server'   
set interfaces openvpn vtun10 persistent-tunnel   
set interfaces openvpn vtun10 protocol 'udp'   
set interfaces openvpn vtun10 server client client1 ip '10.10.0.10'   
set interfaces openvpn vtun10 server domain-name 'vyos.net'   
set interfaces openvpn vtun10 server max-connections '250'   
set interfaces openvpn vtun10 server name-server '172.16.254.30'   
set interfaces openvpn vtun10 server subnet '10.10.0.0/24'   
set interfaces openvpn vtun10 server topology 'subnet'   
set interfaces openvpn vtun10 tls ca-cert ca-1   
set interfaces openvpn vtun10 tls certificate srv-1   
set interfaces openvpn vtun10 tls crypt-key srv-1   
set interfaces openvpn vtun10 tls dh-params dh-1   
set interfaces openvpn vtun10 use-lzo-compression   
Client Side  
set interfaces openvpn vtun10 encryption cipher 'aes256'   
set interfaces openvpn vtun10 hash 'sha512'   
set interfaces openvpn vtun10 mode 'client'   
set interfaces openvpn vtun10 persistent-tunnel   
set interfaces openvpn vtun10 protocol 'udp'   
set interfaces openvpn vtun10 remote-host '172.18.201.10'   
set interfaces openvpn vtun10 remote-port '1194'   
set interfaces openvpn vtun10 tls ca-cert ca-1   
set interfaces openvpn vtun10 tls certificate client-1   
set interfaces openvpn vtun10 tls crypt-key client-1   
set interfaces openvpn vtun10 use-lzo-compression   
Options  
We do not have CLI nodes for every single OpenVPN option. If an option is missing, a feature request should be opened at Phabricator so all users can benefit from it (see Issues/Feature requests).

If you are a hacker or want to try on your own we support passing raw OpenVPN options to OpenVPN.

set interfaces openvpn vtun10 openvpn-option ‘persistent-key’  
Will add persistent-key at the end of the generated OpenVPN configuration. Please use this only as last resort - things might break and OpenVPN won’t start if you pass invalid options/syntax.

set interfaces openvpn vtun10 openvpn-option ‘push &quot;keepalive 1 10&quot;’Will add push "keepalive 1 10" to the generated OpenVPN config file.

Note   
Sometimes option lines in the generated OpenVPN configuration require quotes. This is done through a hack on our config generator. You can pass quotes using the &quot; statement.

OpenVPN Data Channel Offload (DCO)  
OpenVPN Data Channel Offload (DCO) enables significant performance enhancement in encrypted OpenVPN data processing. By minimizing context switching for each packet, DCO effectively reduces overhead. This optimization is achieved by keeping most data handling tasks within the kernel, avoiding frequent switches between kernel and user space for encryption and packet handling.

As a result, the processing of each packet becomes more efficient, potentially leveraging hardware encryption offloading support available in the kernel.

Note   
OpenVPN DCO is not full OpenVPN features supported , is currently considered experimental. Furthermore, there are certain OpenVPN features and use cases that remain incompatible with DCO. To get a comprehensive understanding of the limitations associated with DCO, refer to the list of known limitations in the documentation.

https://community.openvpn.net/openvpn/wiki/DataChannelOffload/Features   
DCO support is a per-tunnel option and it is not automatically enabled by default for new or upgraded tunnels. Existing tunnels will continue to function as they have in the past.

DCO can be enabled for both new and existing tunnels,VyOS adds an option in each tunnel configuration where we can enable this function .The current best practice is to create a new tunnel with DCO to minimize the chance of problems with existing clients.

set interfaces openvpn <name> offload dco

Enable OpenVPN Data Channel Offload feature by loading the appropriate kernel module. Disabled by default - no kernel module loaded.

Note   
Enable this feature causes an interface reset.

VyOS provides some operational commands on OpenVPN.

Check status  
The following commands let you check tunnel status.

show openvpn client   
Use this command to check the tunnel status for OpenVPN client interfaces.

show openvpn server   
Use this command to check the tunnel status for OpenVPN server interfaces.

show openvpn site-to-site   
Use this command to check the tunnel status for OpenVPN site-to-site interfaces.

Reset OpenVPN  
The following commands let you reset OpenVPN.

reset openvpn client <text>   
Use this command to reset the specified OpenVPN client.

reset openvpn interface <interface>   
Use this command to reset the OpenVPN process on a specific interface.

MACsec  
MACsec is an IEEE standard (IEEE 802.1AE) for MAC security, introduced in 2006. It defines a way to establish a protocol independent connection between two hosts with data   
confidentiality, authenticity and/or integrity, using GCM-AES-128. MACsec operates on the Ethernet layer and as such is a layer 2 protocol, which means it’s designed to secure traffic within a layer 2 network, including DHCP or ARP requests. It does not compete with other security solutions such as IPsec (layer 3) or TLS (layer 4), as all those solutions are used for their own specific use cases.

Configuration  
Common interface configuration  
 set interfaces macsec <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/  
• /config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces macsec macsec0 address 192.0.2.1/24   
set interfaces macsec macsec0 address 2001:db8::1/64   
set interfaces macsec macsec0 address dhcp   
set interfaces macsec macsec0 address dhcpv6  
 set interfaces macsec <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces macsec macsec0 description 'This is an awesome interface running on VyOS' set interfaces macsec <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces macsec macsec0 disable  
 set interfaces macsec <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces macsec macsec0 disable-flow-control  
 set interfaces macsec <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces macsec macsec0 disable-link-detect  
 set interfaces macsec <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces macsec macsec0 mac '00:01:02:03:04:05'

set interfaces macsec <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces macsec macsec0 mtu 1600  
 set interfaces macsec <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces macsec <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces macsec macsec0 ip arp-cache-timeout 180  
 set interfaces macsec <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces macsec macsec0 ip disable-arp-filter  
 set interfaces macsec <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces macsec macsec0 ip disable-forwarding  
 set interfaces macsec <interface> ip enable-directed-broadcast

Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces macsec macsec0 ip enable-directed-broadcast  
 set interfaces macsec <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces macsec macsec0 ip enable-arp-accept  
 set interfaces macsec <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces macsec macsec0 ip enable-arp-announce  
 set interfaces macsec <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces macsec macsec0 ip enable-arp-ignore  
 set interfaces macsec <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces macsec macsec0 ip enable-proxy-arp  
 set interfaces macsec <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream

router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation   
 Cisco and Allied Telesyn call it Private VLAN•  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces macsec <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces macsec <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces macsec macsec0 ipv6 address autoconf  
 set interfaces macsec <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces macsec macsec0 ipv6 address eui64 2001:db8:beef::/64 set interfaces macsec <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces macsec macsec0 ipv6 address no-default-link-local  
 set interfaces macsec <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces macsec macsec0 ipv6 disable-forwarding  
 set interfaces macsec <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces macsec <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces macsec macsec0 ipv6 accept-dad 2  
 set interfaces macsec <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces macsec macsec0 ipv6 dup-addr-detect-transmits 5 set interfaces macsec <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces macsec macsec0 vrf red   
DHCP(v6)  
 set interfaces macsec <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces macsec macsec0 dhcp-options client-id 'foo-bar'  
 set interfaces macsec <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:

set interfaces macsec macsec0 dhcp-options host-name 'VyOS'  
 set interfaces macsec <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces macsec macsec0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces macsec <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces macsec macsec0 dhcp-options no-default-route  
 set interfaces macsec <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces macsec macsec0 dhcp-options default-route-distance 220  
 set interfaces macsec <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces macsec macsec0 dhcp-options reject 192.168.100.0/24  
 set interfaces macsec <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces macsec macsec0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces macsec <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces macsec macsec0 dhcpv6-options no-release  
 set interfaces macsec <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces macsec macsec0 dhcpv6-options parameters-only  
 set interfaces macsec <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces macsec macsec0 dhcpv6-options rapid-commit  
 set interfaces macsec <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces macsec macsec0 dhcpv6-options temporary

DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces macsec <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces macsec macsec0 dhcpv6-options pd 0 length 56  
 set interfaces macsec <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces macsec macsec0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces macsec <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces macsec macsec0 dhcpv6-options pd 0 interface eth8 sla-id 1 MACsec options  
 set interfaces macsec <interface> security cipher <gcm-aes-128|gcm-aes-256> Select cipher suite used for cryptographic operations. This setting is mandatory.

set interfaces macsec <interface> security encrypt   
MACsec only provides authentication by default, encryption is optional. This command will enable encryption for all outgoing packets.

set interfaces macsec <interface> source-interface <physical-source>   
A physical interface is required to connect this MACsec instance to. Traffic leaving this interface will now be authenticated/encrypted.

Static Keys  
Static SAK mode can be configured manually on each device wishing to use MACsec. Keys must be set statically on all devices for traffic to flow properly. Key rotation is dependent on the administrator updating all keys manually across connected devices. Static SAK mode can not be used with MKA.

set interfaces macsec <interface> security static key <key>   
Set the device’s transmit (TX) key. This key must be a hex string that is 16-bytes (GCM-AES-128) or 32-bytes (GCM-AES-256).

set interfaces macsec <interface> security static peer <peer> mac <mac address>

Set the peer’s MAC address  
 set interfaces macsec <interface> security static peer <peer> key <key> Set the peer’s key used to receive (RX) traffic  
 set interfaces macsec <interface> security static peer <peer> disable Disable the peer configuration   
Key Management  
MKA is used to synchronize keys between individual peers.

set interfaces macsec <interface> security mka cak <key>   
IEEE 802.1X/MACsec pre-shared key mode. This allows configuring MACsec with a pre-shared key using a CAK and CKN pair.

set interfaces macsec <interface> security mka ckn <key>   
CKN key  
 set interfaces macsec <interface> security mka priority <priority>   
The peer with lower priority will become the key server and start distributing SAKs.

Replay protection  
 set interfaces macsec <interface> security replay-window <window>   
IEEE 802.1X/MACsec replay protection window. This determines a window in which replay is tolerated, to allow receipt of frames that have been misordered by the network.

• 0: No replay window, strict check  
• 1-4294967295: Number of packets that could be misordered Operation  
 run generate macsec mka cak <gcm-aes-128|gcm-aes-256>   
Generate MKA CAK key 128 or 256 bits.

vyos@vyos:~$ generate macsec mka cak gcm-aes-128 20693b6e08bfa482703a563898c9e3ad  
 run generate macsec mka ckn   
Generate MKA CAK key.

vyos@vyos:~$ generate macsec mka ckn   
88737efef314ee319b2cbf30210a5f164957d884672c143aefdc0f5f6bc49eb2 show interfaces macsec   
List all MACsec interfaces.

vyos@vyos:~$ show interfaces macsec   
17: macsec1: protect on validate strict sc off sa off encrypt on send\_sci on end\_station off scb off replay off  
 cipher suite: GCM-AES-128, using ICV length 16  
 TXSC: 005056bfefaa0001 on SA 0   
20: macsec0: protect on validate strict sc off sa off encrypt off send\_sci on end\_station off scb off replay off  
 cipher suite: GCM-AES-128, using ICV length 16  
 TXSC: 005056bfefaa0001 on SA 0  
 show interfaces macsec <interface>   
Show specific MACsec interface information   
vyos@vyos:~$ show interfaces macsec macsec1

17: macsec1: protect on validate strict sc off sa off encrypt on send\_sci on end\_station off scb off replay off  
 cipher suite: GCM-AES-128, using ICV length 16  
 TXSC: 005056bfefaa0001 on SA 0   
Examples  
• Two routers connected both via eth1 through an untrusted switch   
 R1 has 192.0.2.1/24 & 2001:db8::1/64  
•• R2 has 192.0.2.2/24 & 2001:db8::2/64   
R1   
set interfaces macsec macsec1 address '192.0.2.1/24'   
set interfaces macsec macsec1 address '2001:db8::1/64'   
set interfaces macsec macsec1 security cipher 'gcm-aes-128'   
set interfaces macsec macsec1 security encrypt   
set interfaces macsec macsec1 security mka cak '232e44b7fda6f8e2d88a07bf78a7aff4' set interfaces macsec macsec1 security mka ckn   
'40916f4b23e3d548ad27eedd2d10c6f98c2d21684699647d63d41b500dfe8836'   
set interfaces macsec macsec1 source-interface 'eth1'   
R2   
set interfaces macsec macsec1 address '192.0.2.2/24'   
set interfaces macsec macsec1 address '2001:db8::2/64'   
set interfaces macsec macsec1 security cipher 'gcm-aes-128'   
set interfaces macsec macsec1 security encrypt   
set interfaces macsec macsec1 security mka cak '232e44b7fda6f8e2d88a07bf78a7aff4' set interfaces macsec macsec1 security mka ckn   
'40916f4b23e3d548ad27eedd2d10c6f98c2d21684699647d63d41b500dfe8836'   
set interfaces macsec macsec1 source-interface 'eth1'   
Pinging (IPv6) the other host and intercepting the traffic in eth1 will show you the content is encrypted.

17:35:44.586668 00:50:56:bf:ef:aa > 00:50:56:b3:ad:d6, ethertype Unknown (0x88e5), length 150:  
 0x0000: 2c00 0000 000a 0050 56bf efaa 0001 d9fb ,......PV.......

0x0010: 920a 8b8d 68ed 9609 29dd e767 25a4 4466 ....h...)..g%.Df 0x0020: 5293 487b 9990 8517 3b15 22c7 ea5c ac83 R.H{....;."..\..

0x0030: 4c6e 13cf 0743 f917 2c4e 694e 87d1 0f09 Ln...C..,NiN....

0x0040: 0f77 5d53 ed75 cfe1 54df 0e5a c766 93cb .w]S.u..T..Z.f..

0x0050: c4f2 6e23 f200 6dfe 3216 c858 dcaa a73b ..n#..m.2..X...; 0x0060: 4dd1 9358 d9e4 ed0e 072f 1acc 31c4 f669 M..X...../..1..i 0x0070: e93a 9f38 8a62 17c6 2857 6ac5 ec11 8b0e .:.8.b..(Wj.....

0x0080: 6b30 92a5 7ccc 720b k0..|.r.

Disabling the encryption on the link by removing security encrypt will show the unencrypted but authenticated content.

17:37:00.746155 00:50:56:bf:ef:aa > 00:50:56:b3:ad:d6, ethertype Unknown (0x88e5), length 150:  
 0x0000: 2000 0000 0009 0050 56bf efaa 0001 86dd .......PV.......

0x0010: 6009 86f3 0040 3a40 2001 0db8 0000 0000 `....@:@........

0x0020: 0000 0000 0000 0001 2001 0db8 0000 0000 ................

0x0030: 0000 0000 0000 0002 8100 d977 0f30 0003 ...........w.0..

0x0040: 1ca0 c65e 0000 0000 8d93 0b00 0000 0000 ...^............

0x0050: 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f ................

0x0060: 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f .!"#$%&'()\*+,-./  
 0x0070: 3031 3233 3435 3637 87d5 eed3 3a39 d52b 01234567....:9.+  
 0x0080: a282 c842 5254 ef28 ...BRT.(   
R1 Static Key   
set interfaces macsec macsec1 address '192.0.2.1/24'   
set interfaces macsec macsec1 address '2001:db8::1/64'   
set interfaces macsec macsec1 security cipher 'gcm-aes-128'   
set interfaces macsec macsec1 security encrypt   
set interfaces macsec macsec1 security static key 'ddd6f4a7be4d8bbaf88b26f10e1c05f7' set interfaces macsec macsec1 security static peer R2 mac 00:11:22:33:44:02   
set interfaces macsec macsec1 security static peer R2 key   
'eadcc0aa9cf203f3ce651b332bd6e6c7'   
set interfaces macsec macsec1 source-interface 'eth1'   
R2 Static Key   
set interfaces macsec macsec1 address '192.0.2.2/24'   
set interfaces macsec macsec1 address '2001:db8::2/64'   
set interfaces macsec macsec1 security cipher 'gcm-aes-128'   
set interfaces macsec macsec1 security encrypt   
set interfaces macsec macsec1 security static key 'eadcc0aa9cf203f3ce651b332bd6e6c7' set interfaces macsec macsec1 security static peer R2 mac 00:11:22:33:44:01   
set interfaces macsec macsec1 security static peer R2 key   
'ddd6f4a7be4d8bbaf88b26f10e1c05f7'   
set interfaces macsec macsec1 source-interface 'eth1'   
Loopback  
The loopback networking interface is a virtual network device implemented entirely in software. All traffic sent to it “loops back” and just targets services on your local machine.

Note   
There can only be one loopback lo interface on the system. If you need multiple interfaces, please use the Dummy interface type.

Hint   
A lookback interface is always up, thus it could be used for management traffic or as   
source/destination for and IGP like BGP so your internal BGP link is not dependent on physical link states and multiple routes can be chosen to the destination. A Dummy Interface should always be preferred over a Loopback interface.

Configuration  
Common interface configuration  
 set interfaces loopback <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
Example:   
set interfaces loopback lo address 192.0.2.1/24   
set interfaces loopback lo address 2001:db8::1/64  
 set interfaces loopback <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces loopback lo description 'This is an awesome interface running on VyOS' Operation  
 show interfaces loopback   
Show brief interface information.

vyos@vyos:~$ show interfaces loopback   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down Interface IP Address S/L Description  
--------- ---------- --- -----------  
lo 127.0.0.1/8 u/u  
 ::1/128  
 show interfaces loopback lo   
Show detailed information on the given loopback interface lo.

vyos@vyos:~$ show interfaces loopback lo   
lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
 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  
 inet6 ::1/128 scope host  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast  
 300 6 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 300 6 0 0 0 0   
L2TPv3  
Layer 2 Tunnelling Protocol Version 3 is an IETF standard related to L2TP that can be used as an alternative protocol to MPLS for encapsulation of multiprotocol Layer 2 communications traffic over IP networks. Like L2TP, L2TPv3 provides a pseudo-wire service but is scaled to fit carrier requirements.

L2TPv3 can be regarded as being to MPLS what IP is to ATM: a simplified version of the same concept, with much of the same benefit achieved at a fraction of the effort, at the cost of losing some technical features considered less important in the market.

In the case of L2TPv3, the features lost are teletraffic engineering features considered important in MPLS. However, there is no reason these features could not be re-engineered in or on top of L2TPv3 in later products.

The protocol overhead of L2TPv3 is also significantly bigger than MPLS.

L2TPv3 is described in RFC 3931.

Configuration  
Common interface configuration  
 set interfaces l2tpv3 <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
Example:   
set interfaces l2tpv3 l2tpeth0 address 192.0.2.1/24   
set interfaces l2tpv3 l2tpeth0 address 2001:db8::1/64  
 set interfaces l2tpv3 <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces l2tpv3 l2tpeth0 description 'This is an awesome interface running on VyOS' set interfaces l2tpv3 <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces l2tpv3 l2tpeth0 disable  
 set interfaces l2tpv3 <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces l2tpv3 l2tpeth0 disable-flow-control  
 set interfaces l2tpv3 <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces l2tpv3 l2tpeth0 disable-link-detect  
 set interfaces l2tpv3 <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces l2tpv3 l2tpeth0 mac '00:01:02:03:04:05'

set interfaces l2tpv3 <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces l2tpv3 l2tpeth0 mtu 1600  
 set interfaces l2tpv3 <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces l2tpv3 <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces l2tpv3 l2tpeth0 ip arp-cache-timeout 180  
 set interfaces l2tpv3 <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces l2tpv3 l2tpeth0 ip disable-arp-filter  
 set interfaces l2tpv3 <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces l2tpv3 l2tpeth0 ip disable-forwarding  
 set interfaces l2tpv3 <interface> ip enable-directed-broadcast

Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces l2tpv3 l2tpeth0 ip enable-directed-broadcast  
 set interfaces l2tpv3 <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces l2tpv3 l2tpeth0 ip enable-arp-accept  
 set interfaces l2tpv3 <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces l2tpv3 l2tpeth0 ip enable-arp-announce  
 set interfaces l2tpv3 <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces l2tpv3 l2tpeth0 ip enable-arp-ignore  
 set interfaces l2tpv3 <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces l2tpv3 l2tpeth0 ip enable-proxy-arp  
 set interfaces l2tpv3 <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream

router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation   
 Cisco and Allied Telesyn call it Private VLAN•  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces l2tpv3 <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces l2tpv3 <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 address autoconf  
 set interfaces l2tpv3 <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 address eui64 2001:db8:beef::/64 set interfaces l2tpv3 <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 address no-default-link-local  
 set interfaces l2tpv3 <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 disable-forwarding  
 set interfaces l2tpv3 <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces l2tpv3 <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 accept-dad 2  
 set interfaces l2tpv3 <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces l2tpv3 l2tpeth0 ipv6 dup-addr-detect-transmits 5 set interfaces l2tpv3 <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces l2tpv3 l2tpeth0 vrf red   
L2TPv3 options  
 set interfaces l2tpv3 <interface> encapsulation <udp | ip>   
Set the encapsulation type of the tunnel. Valid values for encapsulation are: udp, ip.

This defaults to UDP  
 set interfaces l2tpv3 <interface> source-address <address> Set the IP address of the local interface to be used for the tunnel.

This address must be the address of a local interface. It may be specified as an IPv4 address or an IPv6 address.

set interfaces l2tpv3 <interface> remote <address>   
Set the IP address of the remote peer. It may be specified as an IPv4 address or an IPv6 address.

set interfaces l2tpv3 <interface> session-id <id>

Set the session id, which is a 32-bit integer value. Uniquely identifies the session being created. The value used must match the peer\_session\_id value being used at the peer.

set interfaces l2tpv3 <interface> peer-session-id <id>   
Set the peer-session-id, which is a 32-bit integer value assigned to the session by the peer. The value used must match the session\_id value being used at the peer.

set interfaces l2tpv3 <interface> tunnel-id <id>   
Set the tunnel id, which is a 32-bit integer value. Uniquely identifies the tunnel into which the session will be created.

set interfaces l2tpv3 <interface> peer-tunnel-id <id>   
Set the tunnel id, which is a 32-bit integer value. Uniquely identifies the tunnel into which the session will be created.

Example  
Over IP  
# show interfaces l2tpv3   
l2tpv3 l2tpeth10 {  
 address 192.168.37.1/27  
 encapsulation ip  
 source-address 192.0.2.1  
 peer-session-id 100  
 peer-tunnel-id 200  
 remote 203.0.113.24  
 session-id 100  
 tunnel-id 200   
}   
The inverse configuration has to be applied to the remote side.

Over UDP  
UDP mode works better with NAT:  
• Set source-address to your local IP (LAN).

• Add a forwarding rule matching UDP port on your internet router.

# show interfaces l2tpv3   
l2tpv3 l2tpeth10 {  
 address 192.168.37.1/27  
 destination-port 9001  
 encapsulation udp  
 source-address 192.0.2.1  
 peer-session-id 100  
 peer-tunnel-id 200  
 remote 203.0.113.24  
 session-id 100  
 source-port 9000  
 tunnel-id 200   
}   
To create more than one tunnel, use distinct UDP ports.

Over IPSec, L2 VPN (bridge)

This is the LAN extension use case. The eth0 port of the distant VPN peers will be directly connected like if there was a switch between them.

IPSec:   
set vpn ipsec authentication psk <pre-shared-name> id '%any'   
set vpn ipsec authentication psk <pre-shared-name> secret <pre-shared-key>   
set vpn ipsec interface <VPN-interface>   
set vpn ipsec esp-group test-ESP-1 lifetime '3600'   
set vpn ipsec esp-group test-ESP-1 mode 'transport'   
set vpn ipsec esp-group test-ESP-1 pfs 'enable'   
set vpn ipsec esp-group test-ESP-1 proposal 1 encryption 'aes128'   
set vpn ipsec esp-group test-ESP-1 proposal 1 hash 'sha1'   
set vpn ipsec ike-group test-IKE-1 key-exchange 'ikev1'   
set vpn ipsec ike-group test-IKE-1 lifetime '3600'   
set vpn ipsec ike-group test-IKE-1 proposal 1 dh-group '5'   
set vpn ipsec ike-group test-IKE-1 proposal 1 encryption 'aes128'   
set vpn ipsec ike-group test-IKE-1 proposal 1 hash 'sha1'   
set vpn ipsec site-to-site peer <connection-name> authentication mode 'pre-shared-secret' set vpn ipsec site-to-site peer <connection-name> connection-type 'initiate'   
set vpn ipsec site-to-site peer <connection-name> ike-group 'test-IKE-1'   
set vpn ipsec site-to-site peer <connection-name> ikev2-reauth 'inherit'   
set vpn ipsec site-to-site peer <connection-name> local-address <local-ip>   
set vpn ipsec site-to-site peer <connection-name> tunnel 1 esp-group 'test-ESP-1'   
set vpn ipsec site-to-site peer <connection-name> tunnel 1 protocol 'l2tp'   
Bridge:   
set interfaces bridge br0 description 'L2 VPN Bridge'   
# remote side in this example:   
# set interfaces bridge br0 address '172.16.30.18/30'   
set interfaces bridge br0 address '172.16.30.17/30'   
set interfaces bridge br0 member interface eth0   
set interfaces ethernet eth0 description 'L2 VPN Physical port'   
L2TPv3:   
set interfaces bridge br0 member interface 'l2tpeth0'   
set interfaces l2tpv3 l2tpeth0 description 'L2 VPN Tunnel'   
set interfaces l2tpv3 l2tpeth0 destination-port '5000'   
set interfaces l2tpv3 l2tpeth0 encapsulation 'ip'   
set interfaces l2tpv3 l2tpeth0 source-address <local-ip>   
set interfaces l2tpv3 l2tpeth0 mtu '1500'   
set interfaces l2tpv3 l2tpeth0 peer-session-id '110'   
set interfaces l2tpv3 l2tpeth0 peer-tunnel-id '10'   
set interfaces l2tpv3 l2tpeth0 remote <peer-ip>   
set interfaces l2tpv3 l2tpeth0 session-id '110'   
set interfaces l2tpv3 l2tpeth0 source-port '5000'   
set interfaces l2tpv3 l2tpeth0 tunnel-id '10'   
GENEVE

GENEVE supports all of the capabilities of VXLAN, NVGRE, and STT and was designed to overcome their perceived limitations. Many believe GENEVE could eventually replace these earlier formats entirely.

GENEVE is designed to support network virtualization use cases, where tunnels are typically established to act as a backplane between the virtual switches residing in hypervisors, physical switches, or middleboxes or other appliances. An arbitrary IP network can be used as an underlay although Clos networks - A technique for composing network fabrics larger than a single switch while maintaining non-blocking bandwidth across connection points. ECMP is used to divide traffic across the multiple links and switches that constitute the fabric. Sometimes termed “leaf and spine” or “fat tree” topologies.

Geneve Header:   
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Ver| Opt Len |O|C| Rsvd. | Protocol Type |   
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | Virtual Network Identifier (VNI) | Reserved |   
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | Variable Length Options |   
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Configuration  
Common interface configuration  
 set interfaces geneve <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
Example:   
set interfaces geneve gnv0 address 192.0.2.1/24   
set interfaces geneve gnv0 address 2001:db8::1/64  
 set interfaces geneve <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces geneve gnv0 description 'This is an awesome interface running on VyOS' set interfaces geneve <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces geneve gnv0 disable  
 set interfaces geneve <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces geneve gnv0 disable-flow-control  
 set interfaces geneve <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces geneve gnv0 disable-link-detect  
 set interfaces geneve <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces geneve gnv0 mac '00:01:02:03:04:05'  
 set interfaces geneve <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces geneve gnv0 mtu 1600  
 set interfaces geneve <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces geneve <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces geneve gnv0 ip arp-cache-timeout 180  
 set interfaces geneve <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces geneve gnv0 ip disable-arp-filter  
 set interfaces geneve <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces geneve gnv0 ip disable-forwarding  
 set interfaces geneve <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces geneve gnv0 ip enable-directed-broadcast  
 set interfaces geneve <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces geneve gnv0 ip enable-arp-accept  
 set interfaces geneve <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces geneve gnv0 ip enable-arp-announce  
 set interfaces geneve <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces geneve gnv0 ip enable-arp-ignore  
 set interfaces geneve <interface> ip enable-proxy-arp

Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces geneve gnv0 ip enable-proxy-arp  
 set interfaces geneve <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces geneve <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces geneve <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces geneve gnv0 ipv6 address autoconf  
 set interfaces geneve <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces geneve gnv0 ipv6 address eui64 2001:db8:beef::/64

set interfaces geneve <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces geneve gnv0 ipv6 address no-default-link-local  
 set interfaces geneve <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces geneve gnv0 ipv6 disable-forwarding  
 set interfaces geneve <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces geneve <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces geneve gnv0 ipv6 accept-dad 2  
 set interfaces geneve <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces geneve gnv0 ipv6 dup-addr-detect-transmits 5 set interfaces geneve <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces geneve gnv0 vrf red   
GENEVE options  
 set interfaces geneve gnv0 remote <address>

Configure GENEVE tunnel far end/remote tunnel endpoint.

set interfaces geneve gnv0 vni <vni>   
VNI is an identifier for a unique element of a virtual network. In many situations this may represent an L2 segment, however, the control plane defines the forwarding semantics of decapsulated packets. The VNI MAY be used as part of ECMP forwarding decisions or MAY be used as a mechanism to distinguish between overlapping address spaces contained in the encapsulated packet when load balancing across CPUs.

Ethernet  
This will be the most widely used interface on a router carrying traffic to the real world.

Configuration  
Common interface configuration  
 set interfaces ethernet <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:   
 /config/scripts/dhcp-client/pre-hooks.d/•  
• /config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces ethernet eth0 address 192.0.2.1/24   
set interfaces ethernet eth0 address 2001:db8::1/64   
set interfaces ethernet eth0 address dhcp   
set interfaces ethernet eth0 address dhcpv6  
 set interfaces ethernet <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces ethernet eth0 description 'This is an awesome interface running on VyOS' set interfaces ethernet <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces ethernet eth0 disable  
 set interfaces ethernet <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces ethernet eth0 disable-flow-control  
 set interfaces ethernet <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces ethernet eth0 disable-link-detect  
 set interfaces ethernet <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces ethernet eth0 mac '00:01:02:03:04:05'  
 set interfaces ethernet <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces ethernet eth0 mtu 1600  
 set interfaces ethernet <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces ethernet eth0 ip arp-cache-timeout 180  
 set interfaces ethernet <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces ethernet eth0 ip disable-arp-filter  
 set interfaces ethernet <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces ethernet eth0 ip disable-forwarding  
 set interfaces ethernet <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces ethernet eth0 ip enable-directed-broadcast  
 set interfaces ethernet <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces ethernet eth0 ip enable-arp-accept  
 set interfaces ethernet <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces ethernet eth0 ip enable-arp-announce  
 set interfaces ethernet <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces ethernet eth0 ip enable-arp-ignore  
 set interfaces ethernet <interface> ip enable-proxy-arp

Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces ethernet eth0 ip enable-proxy-arp  
 set interfaces ethernet <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces ethernet <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces ethernet <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces ethernet eth0 ipv6 address autoconf  
 set interfaces ethernet <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces ethernet eth0 ipv6 address eui64 2001:db8:beef::/64

set interfaces ethernet <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces ethernet eth0 ipv6 address no-default-link-local  
 set interfaces ethernet <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces ethernet eth0 ipv6 disable-forwarding  
 set interfaces ethernet <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces ethernet eth0 ipv6 accept-dad 2  
 set interfaces ethernet <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces ethernet eth0 ipv6 dup-addr-detect-transmits 5 set interfaces ethernet <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces ethernet eth0 vrf red   
DHCP(v6)  
 set interfaces ethernet <interface> dhcp-options client-id <description>

RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces ethernet eth0 dhcp-options client-id 'foo-bar'  
 set interfaces ethernet <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces ethernet eth0 dhcp-options host-name 'VyOS'  
 set interfaces ethernet <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces ethernet eth0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces ethernet <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces ethernet eth0 dhcp-options no-default-route  
 set interfaces ethernet <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces ethernet eth0 dhcp-options default-route-distance 220  
 set interfaces ethernet <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces ethernet eth0 dhcp-options reject 192.168.100.0/24  
 set interfaces ethernet <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces ethernet eth0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces ethernet <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces ethernet eth0 dhcpv6-options no-release  
 set interfaces ethernet <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is

useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces ethernet eth0 dhcpv6-options parameters-only  
 set interfaces ethernet <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces ethernet eth0 dhcpv6-options rapid-commit  
 set interfaces ethernet <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces ethernet eth0 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces ethernet <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces ethernet eth0 dhcpv6-options pd 0 length 56  
 set interfaces ethernet <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces ethernet eth0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces ethernet <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces ethernet eth0 dhcpv6-options pd 0 interface eth8 sla-id 1 Ethernet options  
 set interfaces ethernet <interface> duplex <auto | full | half>   
Configure physical interface duplex setting.

• auto - interface duplex setting is auto-negotiated• full - always use full-duplex  
• half - always use half-duplex   
VyOS default will be auto.

set interfaces ethernet <interface> speed <auto | 10 | 100 | 1000 | 2500 | 5000 | 10000 | 25000 | 40000 | 50000 | 100000>   
Configure physical interface speed setting.

• auto - interface speed is auto-negotiated  
• 10 - 10 MBit/s   
 100 - 100 MBit/s•  
• 1000 - 1 GBit/s  
• 2500 - 2.5 GBit/s  
• 5000 - 5 GBit/s   
 10000 - 10 GBit/s•  
• 25000 - 25 GBit/s  
• 40000 - 40 GBit/s  
• 50000 - 50 GBit/s   
 100000 - 100 GBit/s•  
VyOS default will be auto.

Offloading  
 set interfaces ethernet <interface> offload <gro | gso | lro | rps | sg | tso> Enable different types of hardware offloading on the given NIC.

GSO is a pure software offload that is meant to deal with cases where device drivers cannot perform the offloads described above. What occurs in GSO is that a given skbuff will have its data broken out over multiple skbuffs that have been resized to match the MSS provided via skb\_shinfo()->gso\_size.

Before enabling any hardware segmentation offload a corresponding software offload is required in GSO. Otherwise it becomes possible for a frame to be re-routed between devices and end up being unable to be transmitted.

GRO is the complement to GSO. Ideally any frame assembled by GRO should be segmented to create an identical sequence of frames using GSO, and any sequence of frames segmented by GSO should be able to be reassembled back to the original by GRO. The only exception to this is IPv4 ID in the case that the DF bit is set for a given IP header. If the value of the IPv4 ID is not sequentially incrementing it will be altered so that it is when a frame assembled via GRO is segmented via GSO.

RPS is logically a software implementation of RSS. Being in software, it is necessarily called later in the datapath. Whereas RSS selects the queue and hence CPU that will run the hardware interrupt handler, RPS selects the CPU to perform protocol processing above the interrupt handler. This is accomplished by placing the packet on the desired CPU’s backlog queue and waking up the CPU for processing. RPS has some advantages over RSS:   
 it can be used with any NIC,•  
•  
 software filters can easily be added to hash over new protocols,• it does not increase hardware device interrupt rate (although it does introduce inter-processor interrupts (IPIs)).

Note   
In order to use TSO/LRO with VMXNET3 adaters one must also enable the SG offloading option.

Authentication (EAPoL)

EAP over LAN (EAPoL) is a network port authentication protocol used in IEEE 802.1X (Port Based Network Access Control) developed to give a generic network sign-on to access network resources.

EAPoL comes with an identify option. We automatically use the interface MAC address as identity parameter.

set interfaces ethernet <interface> eapol ca-certificate <name>   
Set the name of the SSL CA PKI entry used for authentication of the remote side. If an   
intermediate CA certificate is specified, then all parent CA certificates that exist in the PKI, such as the root CA or additional intermediate CAs, will automatically be used during certificate validation to ensure that the full chain of trust is available.

Example:   
set pki ca eapol-server-intermediate-ca <Server intermediate CA contents>   
set pki ca eapol-server-root-ca <Server root CA contents>   
set interfaces ethernet eth0 eapol ca-certificate eapol-server-intermediate-ca  
 set interfaces ethernet <interface> eapol certificate <name>   
Set the name of the x509 client keypair used to authenticate against the 802.1x system. All parent CA certificates of the client certificate, such as intermediate and root CAs, will be sent as part of the EAP-TLS handshake.

Example:   
set pki ca eapol-client-intermediate-ca <Client intermediate CA contents>   
set pki ca eapol-client-root-ca <Client root CA contents>   
set pki certificate eapol-client certificate <Client certificate contents>   
set pki certificate eapol-client private key <Client private key contents>   
set interfaces ethernet eth0 eapol certificate eapol-client   
VLAN  
Regular VLANs (802.1q)  
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces ethernet <interface> vif <vlan-id>

Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note   
Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces ethernet <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/  
• /config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces ethernet eth0 vif 10 address 192.0.2.1/24   
set interfaces ethernet eth0 vif 10 address 2001:db8::1/64   
set interfaces ethernet eth0 vif 10 address dhcp   
set interfaces ethernet eth0 vif 10 address dhcpv6  
 set interfaces ethernet <interface> vif <vlan-id> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces ethernet eth0 vif 10 description 'This is an awesome interface running on VyOS' set interfaces ethernet <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces ethernet eth0 vif 10 disable  
 set interfaces ethernet <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces ethernet eth0 vif 10 disable-link-detect  
 set interfaces ethernet <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces ethernet eth0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces ethernet <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces ethernet eth0 vif 10 mtu 1600  
 set interfaces ethernet <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> vif <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces ethernet eth0 vif 10 ip arp-cache-timeout 180  
 set interfaces ethernet <interface> vif <vlan-id> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces ethernet eth0 vif 10 ip disable-arp-filter  
 set interfaces ethernet <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces ethernet eth0 vif 10 ip disable-forwarding  
 set interfaces ethernet <interface> vif <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces ethernet eth0 vif 10 ip enable-directed-broadcast  
 set interfaces ethernet <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces ethernet eth0 vif 10 ip enable-arp-accept  
 set interfaces ethernet <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces ethernet eth0 vif 10 ip enable-arp-announce  
 set interfaces ethernet <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces ethernet eth0 vif 10 ip enable-arp-ignore  
 set interfaces ethernet <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces ethernet eth0 vif 10 ip enable-proxy-arp  
 set interfaces ethernet <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:

• In RFC 3069 it is called VLAN Aggregation  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
• Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces ethernet <interface> vif <vlan-id> ip source-validation <strict | loose | disable> Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces ethernet eth0 vif 10 ipv6 address autoconf  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces ethernet eth0 vif 10 ipv6 address eui64 2001:db8:beef::/64  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces ethernet eth0 vif 10 ipv6 address no-default-link-local  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces ethernet eth0 vif 10 ipv6 disable-forwarding  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces ethernet eth0 vif 10 ipv6 accept-dad 2  
 set interfaces ethernet <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces ethernet eth0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces ethernet <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces ethernet eth0 vif 10 vrf red   
DHCP(v6)  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces ethernet eth0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces ethernet eth0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id> The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:

set interfaces ethernet eth0 vif 10 dhcp-options vendor-class-id 'VyOS'  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces ethernet eth0 vif 10 dhcp-options no-default-route  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces ethernet eth0 vif 10 dhcp-options default-route-distance 220  
 set interfaces ethernet <interface> vif <vlan-id> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces ethernet eth0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces ethernet eth0 vif 10 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d' set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces ethernet eth0 vif 10 dhcpv6-options no-release  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces ethernet eth0 vif 10 dhcpv6-options parameters-only  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces ethernet eth0 vif 10 dhcpv6-options rapid-commit  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces ethernet eth0 vif 10 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> length <length>

Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces ethernet eth0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces ethernet eth0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces ethernet <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces ethernet eth0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1 QinQ (802.1ad)  
Call for Contributions   
This section needs improvements, examples and explanations.

Please take a look at the Contributing Guide for our Write Documentation.

IEEE 802.1ad was an Ethernet networking standard informally known as QinQ as an amendment to IEEE standard 802.1q VLAN interfaces as described above. 802.1ad was incorporated into the base 802.1q standard in 2011. The technique is also known as provider bridging, Stacked VLANs, or simply QinQ or Q-in-Q. “Q-in-Q” can for supported devices apply to C-tag stacking on C-tag (Ethernet Type = 0x8100).

The original 802.1q specification allows a single Virtual Local Area Network (VLAN) header to be inserted into an Ethernet frame. QinQ allows multiple VLAN tags to be inserted into a single frame, an essential capability for implementing Metro Ethernet network topologies. Just as QinQ extends 802.1Q, QinQ itself is extended by other Metro Ethernet protocols.

In a multiple VLAN header context, out of convenience the term “VLAN tag” or just “tag” for short is often used in place of “802.1q VLAN header”. QinQ allows multiple VLAN tags in an Ethernet frame; together these tags constitute a tag stack. When used in the context of an Ethernet frame, a QinQ frame is a frame that has 2 VLAN 802.1q headers (double-tagged). In VyOS the terms vif-s and vif-c stand for the ethertype tags that are used.

The inner tag is the tag which is closest to the payload portion of the frame. It is officially called C-TAG (customer tag, with ethertype 0x8100). The outer tag is the one closer/closest to the Ethernet header, its name is S-TAG (service tag with Ethernet Type = 0x88a8).

set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> address <address | dhcp | dhcpv6>   
Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/   
 /config/scripts/dhcp-client/post-hooks.d/•  
Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 address 192.0.2.1/24   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 address 2001:db8::1/64   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 address dhcp   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 address dhcpv6  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> description <description> Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 description 'This is an awesome interface running on VyOS'  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 disable  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 disable-link-detect  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 mac '00:01:02:03:04:05'  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 mtu 1600  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu>

As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip arp-cache-timeout Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip arp-cache-timeout 180  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-arp-filter If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip disable-arp-filter  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip disable-forwarding  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-directed-broadcast Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip enable-directed-broadcast  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-accept

Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip enable-arp-accept  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-announce Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip enable-arp-announce  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-arp-ignore Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip enable-arp-ignore  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip enable-proxy-arp Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ip enable-proxy-arp  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation• Cisco and Allied Telesyn call it Private VLAN

• Hewlett-Packard call it Source-Port filtering or port-isolation  
• Ericsson call it MAC-Forced Forwarding (RFC Draft)  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address autoconf SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 address autoconf  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address eui64 <prefix> EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 address eui64 2001:db8:beef::/64 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 address no-default-link-local   
Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 address no-default-link-local  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 disable-forwarding Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 disable-forwarding  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note

This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 accept-dad 2  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> ipv6 dup-addr-detect-transmits <n>   
The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 ipv6 dup-addr-detect-transmits 5 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> vrf <vrf> Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 vrf red   
DHCP(v6)  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options client-id 'foo-bar'  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options host-name 'VyOS'

set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options vendor-class-id 'VyOS'  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options no-default-route  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options default-route-distance <distance>   
Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options default-route-distance 220 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcp-options reject <address> Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcp-options reject 192.168.100.0/24  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options duid <duid> The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 duid   
'0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options no-release When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options no-release  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options parameters-only  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options rapid-commit When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options rapid-commit  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options temporary

Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 length 56  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces ethernet <interface> vif-s <vlan-id> vif-c <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces ethernet eth0 vif-s 1000 vif-c 20 dhcpv6-options pd 0 interface eth8 sla-id 1 Port Mirror (SPAN)  
SPAN port mirroring can copy the inbound/outbound traffic of the interface to the specified interface, usually the interface can be connected to some special equipment, such as behavior control system, intrusion detection system and traffic collector, and can copy all related traffic from this port. The benefit of mirroring the traffic is that the application is isolated from the source traffic and so application processing does not affect the traffic or the system   
performance.

VyOS uses the mirror option to configure port mirroring. The configuration is divided into 2 different directions. Destination ports should be configured for different traffic directions.

set interfaces ethernet <interface> mirror ingress <monitor-interface>   
Configure port mirroring for interface inbound traffic and copy the traffic to monitor-interface Example: Mirror the inbound traffic of eth1 port to eth3

set interfaces ethernet eth1 mirror ingress eth3  
 set interfaces ethernet <interface> mirror egress <monitor-interface>   
Configure port mirroring for interface outbound traffic and copy the traffic to monitor-interface Example: Mirror the outbound traffic of eth1 port to eth3   
set interfaces ethernet eth1 mirror egress eth3   
Operation  
 show interfaces ethernet   
Show brief interface information.

vyos@vyos:~$ show interfaces ethernet   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 172.18.201.10/24 u/u LAN   
eth1 172.18.202.11/24 u/u WAN   
eth2 - u/D  
 show interfaces ethernet <interface>   
Show detailed information on given <interface>   
vyos@vyos:~$ show interfaces ethernet eth0   
eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000  
 link/ether 00:50:44:00:f5:c9 brd ff:ff:ff:ff:ff:ff  
 inet6 fe80::250:44ff:fe00:f5c9/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast 56735451 179841 0 0 0 142380  
 TX: bytes packets errors dropped carrier collisions 5601460 62595 0 0 0 0  
 show interfaces ethernet <interface> physical   
Show information about physical <interface>   
vyos@vyos:~$ show interfaces ethernet eth0 physical   
Settings for eth0:  
 Supported ports: [ TP ]  
 Supported link modes: 1000baseT/Full  
 10000baseT/Full  
 Supported pause frame use: No  
 Supports auto-negotiation: No  
 Supported FEC modes: Not reported  
 Advertised link modes: Not reported  
 Advertised pause frame use: No  
 Advertised auto-negotiation: No  
 Advertised FEC modes: Not reported  
 Speed: 10000Mb/s  
 Duplex: Full

Port: Twisted Pair  
 PHYAD: 0  
 Transceiver: internal  
 Auto-negotiation: off  
 MDI-X: Unknown  
 Supports Wake-on: uag  
 Wake-on: d  
 Link detected: yes   
driver: vmxnet3   
version: 1.4.16.0-k-NAPI   
firmware-version:   
expansion-rom-version:   
bus-info: 0000:0b:00.0   
supports-statistics: yes   
supports-test: no   
supports-eeprom-access: no   
supports-register-dump: yes   
supports-priv-flags: no  
 show interfaces ethernet <interface> physical offload   
Show available offloading functions on given <interface> vyos@vyos:~$ show interfaces ethernet eth0 physical offload rx-checksumming on   
tx-checksumming on   
tx-checksum-ip-generic on   
scatter-gather off   
tx-scatter-gather off   
tcp-segmentation-offload off   
tx-tcp-segmentation off   
tx-tcp-mangleid-segmentation off   
tx-tcp6-segmentation off   
udp-fragmentation-offload off   
generic-segmentation-offload off   
generic-receive-offload off   
large-receive-offload off   
rx-vlan-offload on   
tx-vlan-offload on   
ntuple-filters off   
receive-hashing on   
tx-gre-segmentation on   
tx-gre-csum-segmentation on   
tx-udp\_tnl-segmentation on   
tx-udp\_tnl-csum-segmentation on   
tx-gso-partial on   
tx-nocache-copy off

rx-all off  
 show interfaces ethernet <interface> transceiver   
Show transceiver information from plugin modules, e.g SFP+, QSFP vyos@vyos:~$ show interfaces ethernet eth5 transceiver  
 Identifier : 0x03 (SFP)  
 Extended identifier : 0x04 (GBIC/SFP defined by 2-wire interface ID) Connector : 0x07 (LC)  
 Transceiver codes : 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 Transceiver type : Ethernet: 1000BASE-SX  
 Encoding : 0x01 (8B/10B)  
 BR, Nominal : 1300MBd  
 Rate identifier : 0x00 (unspecified)  
 Length (SMF,km) : 0km  
 Length (SMF) : 0m  
 Length (50um) : 550m  
 Length (62.5um) : 270m  
 Length (Copper) : 0m  
 Length (OM3) : 0m  
 Laser wavelength : 850nm  
 Vendor name : CISCO-FINISAR  
 Vendor OUI : 00:90:65  
 Vendor PN : FTRJ-8519-7D-CS4  
 Vendor rev : A  
 Option values : 0x00 0x1a  
 Option : RX\_LOS implemented  
 Option : TX\_FAULT implemented  
 Option : TX\_DISABLE implemented  
 BR margin, max : 0%  
 BR margin, min : 0%  
 Vendor SN : FNS092xxxxx  
 Date code : 0506xx

Dummy  
The dummy interface is really a little exotic, but rather useful nevertheless. Dummy interfaces are much like the Loopback interface, except you can have as many as you want.

Note   
Dummy interfaces can be used as interfaces that always stay up (in the same fashion to loopbacks in Cisco IOS), or for testing purposes.

Hint   
On systems with multiple redundant uplinks and routes, it’s a good idea to use a dedicated address for management and dynamic routing protocols. However, assigning that address to a physical link is risky: if that link goes down, that address will become inaccessible. A common solution is to assign the management address to a loopback or a dummy interface and advertise

that address via all physical links, so that it’s reachable through any of them. Since in Linux-based systems, there can be only one loopback interface, it’s better to use a dummy interface for that purpose, since they can be added, removed, and taken up and down independently.

Configuration  
Common interface configuration  
 set interfaces dummy <interface> address <address>   
Configure interface <interface> with one or more interface addresses.

address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 •  
and/or 2001:db8::1/64   
Example:   
set interfaces dummy dum0 address 192.0.2.1/24   
set interfaces dummy dum0 address 2001:db8::1/64  
 set interfaces dummy <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces dummy dum0 description 'This is an awesome interface running on VyOS' set interfaces dummy <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces dummy dum0 disable  
 set interfaces dummy <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces dummy dum0 vrf red   
Operation  
 show interfaces dummy   
Show brief interface information.

vyos@vyos:~$ show interfaces dummy   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
dum0 172.18.254.201/32 u/u  
 show interfaces dummy <interface>   
Show detailed information on given <interface>   
vyos@vyos:~$ show interfaces dummy dum0   
dum0: <BROADCAST,NOARP,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000  
 link/ether 26:7c:8e:bc:fc:f5 brd ff:ff:ff:ff:ff:ff  
 inet 172.18.254.201/32 scope global dum0  
 valid\_lft forever preferred\_lft forever

inet6 fe80::247c:8eff:febc:fcf5/64 scope link  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast  
 0 0 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 1369707 4267 0 0 0 0   
Bridge  
A Bridge is a way to connect two Ethernet segments together in a protocol independent way. Packets are forwarded based on Ethernet address, rather than IP address (like a router). Since forwarding is done at Layer 2, all protocols can go transparently through a bridge. The Linux bridge code implements a subset of the ANSI/IEEE 802.1d standard.

Note   
Spanning Tree Protocol is not enabled by default in VyOS. STP Parameter can be easily enabled if needed.

Configuration  
Common interface configuration  
 set interfaces bridge <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/  
• /config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces bridge br0 address 192.0.2.1/24   
set interfaces bridge br0 address 2001:db8::1/64   
set interfaces bridge br0 address dhcp   
set interfaces bridge br0 address dhcpv6  
 set interfaces bridge <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces bridge br0 description 'This is an awesome interface running on VyOS' set interfaces bridge <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces bridge br0 disable  
 set interfaces bridge <interface> disable-flow-control

Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces bridge br0 disable-flow-control  
 set interfaces bridge <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces bridge br0 disable-link-detect  
 set interfaces bridge <interface> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces bridge br0 mac '00:01:02:03:04:05'  
 set interfaces bridge <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces bridge br0 mtu 1600  
 set interfaces bridge <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bridge <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:

set interfaces bridge br0 ip arp-cache-timeout 180  
 set interfaces bridge <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces bridge br0 ip disable-arp-filter  
 set interfaces bridge <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces bridge br0 ip disable-forwarding  
 set interfaces bridge <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces bridge br0 ip enable-directed-broadcast  
 set interfaces bridge <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces bridge br0 ip enable-arp-accept  
 set interfaces bridge <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces bridge br0 ip enable-arp-announce set interfaces bridge <interface> ip enable-arp-ignore

Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces bridge br0 ip enable-arp-ignore  
 set interfaces bridge <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces bridge br0 ip enable-proxy-arp  
 set interfaces bridge <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
 In RFC 3069 it is called VLAN Aggregation•  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces bridge <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the •  
source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces bridge <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note

This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces bridge br0 ipv6 address autoconf  
 set interfaces bridge <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces bridge br0 ipv6 address eui64 2001:db8:beef::/64 set interfaces bridge <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces bridge br0 ipv6 address no-default-link-local  
 set interfaces bridge <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces bridge br0 ipv6 disable-forwarding  
 set interfaces bridge <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bridge <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD•  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces bridge br0 ipv6 accept-dad 2  
 set interfaces bridge <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces bridge br0 ipv6 dup-addr-detect-transmits 5 set interfaces bridge <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces bridge br0 vrf red   
DHCP(v6)  
 set interfaces bridge <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces bridge br0 dhcp-options client-id 'foo-bar'  
 set interfaces bridge <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces bridge br0 dhcp-options host-name 'VyOS'  
 set interfaces bridge <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces bridge br0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces bridge <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces bridge br0 dhcp-options no-default-route  
 set interfaces bridge <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces bridge br0 dhcp-options default-route-distance 220  
 set interfaces bridge <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces bridge br0 dhcp-options reject 192.168.100.0/24  
 set interfaces bridge <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces bridge br0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d' set interfaces bridge <interface> dhcpv6-options no-release

When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces bridge br0 dhcpv6-options no-release  
 set interfaces bridge <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces bridge br0 dhcpv6-options parameters-only  
 set interfaces bridge <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces bridge br0 dhcpv6-options rapid-commit  
 set interfaces bridge <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces bridge br0 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces bridge <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces bridge br0 dhcpv6-options pd 0 length 56  
 set interfaces bridge <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces bridge br0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces bridge <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces bridge br0 dhcpv6-options pd 0 interface eth8 sla-id 1 Member Interfaces

set interfaces bridge <interface> member interface <member>   
Assign <member> interface to bridge <interface>. A completion helper will help you with all allowed interfaces which can be bridged. This includes Ethernet, Bond / Link Aggregation, L2TPv3, OpenVPN, VXLAN, WLAN/WIFI - Wireless LAN, Tunnel and GENEVE.

set interfaces bridge <interface> member interface <member> priority <priority> Configure individual bridge port <priority>.

Each bridge has a relative priority and cost. Each interface is associated with a port (number) in the STP code. Each has a priority and a cost, that is used to decide which is the shortest path to forward a packet. The lowest cost path is always used unless the other path is down. If you have multiple bridges and interfaces then you may need to adjust the priorities to achieve optimum performance.

set interfaces bridge <interface> member interface <member> cost <cost>   
Path <cost> value for Spanning Tree Protocol. Each interface in a bridge could have a different speed and this value is used when deciding which link to use. Faster interfaces should have lower costs.

Bridge Options  
 set interfaces bridge <interface> aging <time>   
MAC address aging <time> in seconds (default: 300).

set interfaces bridge <interface> max-age <time>   
Bridge maximum aging <time> in seconds (default: 20).

If an another bridge in the spanning tree does not send out a hello packet for a long period of time, it is assumed to be dead.

set interfaces bridge <interface> igmp querier   
Enable IGMP and MLD querier.

set interfaces bridge <interface> igmp snooping   
Enable IGMP and MLD snooping.

STP Parameter  
STP is a network protocol that builds a loop-free logical topology for Ethernet networks. The basic function of STP is to prevent bridge loops and the broadcast radiation that results from them. Spanning tree also allows a network design to include backup links providing fault tolerance if an active link fails.

set interfaces bridge <interface> stp   
Enable spanning tree protocol. STP is disabled by default.

set interfaces bridge <interface> forwarding-delay <delay>   
Spanning Tree Protocol forwarding <delay> in seconds (default: 15).

The forwarding delay time is the time spent in each of the listening and learning states before the Forwarding state is entered. This delay is so that when a new bridge comes onto a busy network it looks at some traffic before participating.

set interfaces bridge <interface> hello-time <interval>   
Spanning Tree Protocol hello advertisement <interval> in seconds (default: 2).

Periodically, a hello packet is sent out by the Root Bridge and the Designated Bridges. Hello packets are used to communicate information about the topology throughout the entire Bridged Local Area Network.

VLAN

Enable VLAN-Aware Bridge  
 set interfaces bridge <interface> enable-vlan   
To activate the VLAN aware bridge, you must activate this setting to use VLAN settings for the bridge   
VLAN Options  
Note   
It is not valid to use the vif 1 option for VLAN aware bridges because VLAN aware bridges assume that all unlabeled packets belong to the default VLAN 1 member and that the VLAN ID of the bridge’s parent interface is always 1   
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces bridge <interface> vif <vlan-id>   
Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note   
Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces bridge <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/   
 /config/scripts/dhcp-client/post-hooks.d/•

Example:   
set interfaces bridge br0 vif 10 address 192.0.2.1/24   
set interfaces bridge br0 vif 10 address 2001:db8::1/64   
set interfaces bridge br0 vif 10 address dhcp   
set interfaces bridge br0 vif 10 address dhcpv6  
 set interfaces bridge <interface> vif <vlan-id> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces bridge br0 vif 10 description 'This is an awesome interface running on VyOS' set interfaces bridge <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces bridge br0 vif 10 disable  
 set interfaces bridge <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces bridge br0 vif 10 disable-link-detect  
 set interfaces bridge <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces bridge br0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces bridge <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces bridge br0 vif 10 mtu 1600  
 set interfaces bridge <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bridge <interface> vif <vlan-id> ip arp-cache-timeout

Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces bridge br0 vif 10 ip arp-cache-timeout 180  
 set interfaces bridge <interface> vif <vlan-id> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces bridge br0 vif 10 ip disable-arp-filter  
 set interfaces bridge <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces bridge br0 vif 10 ip disable-forwarding  
 set interfaces bridge <interface> vif <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces bridge br0 vif 10 ip enable-directed-broadcast  
 set interfaces bridge <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces bridge br0 vif 10 ip enable-arp-accept  
 set interfaces bridge <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When

we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces bridge br0 vif 10 ip enable-arp-announce  
 set interfaces bridge <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces bridge br0 vif 10 ip enable-arp-ignore  
 set interfaces bridge <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces bridge br0 vif 10 ip enable-proxy-arp  
 set interfaces bridge <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces bridge <interface> vif <vlan-id> ip source-validation <strict | loose | disable> Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

disable: No source validation•  
 set interfaces bridge <interface> vif <vlan-id> ipv6 address autoconf

SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces bridge br0 vif 10 ipv6 address autoconf  
 set interfaces bridge <interface> vif <vlan-id> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces bridge br0 vif 10 ipv6 address eui64 2001:db8:beef::/64  
 set interfaces bridge <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces bridge br0 vif 10 ipv6 address no-default-link-local  
 set interfaces bridge <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces bridge br0 vif 10 ipv6 disable-forwarding  
 set interfaces bridge <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bridge <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD   
 1: Enable DAD (default)•  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces bridge br0 vif 10 ipv6 accept-dad 2  
 set interfaces bridge <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n>

The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces bridge br0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces bridge <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces bridge br0 vif 10 vrf red   
DHCP(v6)  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces bridge br0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces bridge br0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id> The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces bridge br0 vif 10 dhcp-options vendor-class-id 'VyOS'  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces bridge br0 vif 10 dhcp-options no-default-route  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces bridge br0 vif 10 dhcp-options default-route-distance 220  
 set interfaces bridge <interface> vif <vlan-id> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces bridge br0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options duid <duid>

The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces bridge br0 vif 10 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces bridge br0 vif 10 dhcpv6-options no-release  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces bridge br0 vif 10 dhcpv6-options parameters-only  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces bridge br0 vif 10 dhcpv6-options rapid-commit  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces bridge br0 vif 10 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces bridge <interface> vif <vlan-id> dhcpv6-options pd <id> length <length> Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces bridge br0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces bridge br0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces bridge <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>

Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces bridge br0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1  
 set interfaces bridge <interface> member interface <member> native-vlan <vlan-id>   
Set the native VLAN ID flag of the interface. When a data packet without a VLAN tag enters the port, the data packet will be forced to add a tag of a specific vlan id. When the vlan id flag flows out, the tag of the vlan id will be stripped   
Example: Set eth0 member port to be native VLAN 2   
set interfaces bridge br1 member interface eth0 native-vlan 2  
 set interfaces bridge <interface> member interface <member> allowed-vlan <vlan-id>   
Allows specific VLAN IDs to pass through the bridge member interface. This can either be an individual VLAN id or a range of VLAN ids delimited by a hyphen.

Example: Set eth0 member port to be allowed VLAN 4   
set interfaces bridge br1 member interface eth0 allowed-vlan 4   
Example: Set eth0 member port to be allowed VLAN 6-8   
set interfaces bridge br1 member interface eth0 allowed-vlan 6-8   
Port Mirror (SPAN)  
SPAN port mirroring can copy the inbound/outbound traffic of the interface to the specified interface, usually the interface can be connected to some special equipment, such as behavior control system, intrusion detection system and traffic collector, and can copy all related traffic from this port. The benefit of mirroring the traffic is that the application is isolated from the source traffic and so application processing does not affect the traffic or the system   
performance.

VyOS uses the mirror option to configure port mirroring. The configuration is divided into 2 different directions. Destination ports should be configured for different traffic directions.

set interfaces bridge <interface> mirror ingress <monitor-interface>   
Configure port mirroring for interface inbound traffic and copy the traffic to monitor-interface Example: Mirror the inbound traffic of br1 port to eth3   
set interfaces bridge br1 mirror ingress eth3  
 set interfaces bridge <interface> mirror egress <monitor-interface>   
Configure port mirroring for interface outbound traffic and copy the traffic to monitor-interface Example: Mirror the outbound traffic of br1 port to eth3   
set interfaces bridge br1 mirror egress eth3   
Examples  
Create a basic bridge  
Creating a bridge interface is very simple. In this example, we will have:   
 A bridge named br100•  
• Member interfaces eth1 and VLAN 10 on interface eth2  
•  
 Enable STP• Bridge answers on IP address 192.0.2.1/24 and 2001:db8::ffff/64   
set interfaces bridge br100 address 192.0.2.1/24

set interfaces bridge br100 address 2001:db8::ffff/64   
set interfaces bridge br100 member interface eth1   
set interfaces bridge br100 member interface eth2.10   
set interfaces bridge br100 stp   
This results in the active configuration:   
vyos@vyos# show interfaces bridge br100  
 address 192.0.2.1/24  
 address 2001:db8::ffff/64  
 member {  
 interface eth1 {  
 }  
 interface eth2.10 {  
 }  
 }  
 stp   
Using VLAN aware Bridge  
An example of creating a VLAN-aware bridge is as follows:  
• A bridge named br100  
• The member interface eth1 is a trunk that allows VLAN 10 to pass• VLAN 10 on member interface eth2 (ACCESS mode)  
•  
 Enable STP• Bridge answers on IP address 192.0.2.1/24 and 2001:db8::ffff/64 set interfaces bridge br100 enable-vlan   
set interfaces bridge br100 member interface eth1 allowed-vlan 10 set interfaces bridge br100 member interface eth2 native-vlan 10   
set interfaces bridge br100 vif 10 address 192.0.2.1/24   
set interfaces bridge br100 vif 10 address 2001:db8::ffff/64   
set interfaces bridge br100 stp   
This results in the active configuration:   
vyos@vyos# show interfaces bridge br100  
 enable-vlan  
 member {  
 interface eth1 {  
 allowed-vlan 10  
 }  
 interface eth2 {  
 native-vlan 10  
 }  
 }  
 stp  
 vif 10 {  
 address 192.0.2.1/24  
 address 2001:db8::ffff/64  
 }

Using the operation mode command to view Bridge Information  
 show bridge   
The show bridge operational command can be used to display configured bridges:   
vyos@vyos:~$ show bridge   
3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 master br0 state forwarding priority 32 cost 100   
4: eth2: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 master br0 state forwarding priority 32 cost 100  
 show bridge <name> fdb   
Show bridge <name> fdb displays the current forwarding table:   
vyos@vyos:~$ show bridge br0 fdb   
50:00:00:08:00:01 dev eth1 vlan 20 master br0 permanent   
50:00:00:08:00:01 dev eth1 vlan 10 master br0 permanent   
50:00:00:08:00:01 dev eth1 master br0 permanent   
33:33:00:00:00:01 dev eth1 self permanent   
33:33:00:00:00:02 dev eth1 self permanent   
01:00:5e:00:00:01 dev eth1 self permanent   
50:00:00:08:00:02 dev eth2 vlan 20 master br0 permanent   
50:00:00:08:00:02 dev eth2 vlan 10 master br0 permanent   
50:00:00:08:00:02 dev eth2 master br0 permanent   
33:33:00:00:00:01 dev eth2 self permanent   
33:33:00:00:00:02 dev eth2 self permanent   
01:00:5e:00:00:01 dev eth2 self permanent   
33:33:00:00:00:01 dev br0 self permanent   
33:33:00:00:00:02 dev br0 self permanent   
33:33:ff:08:00:01 dev br0 self permanent   
01:00:5e:00:00:6a dev br0 self permanent   
33:33:00:00:00:6a dev br0 self permanent   
01:00:5e:00:00:01 dev br0 self permanent   
33:33:ff:00:00:00 dev br0 self permanent  
 show bridge <name> mdb   
Show bridge <name> mdb displays the current multicast group membership table.The table is populated by IGMP and MLD snooping in the bridge driver automatically.

vyos@vyos:~$ show bridge br0 mdb   
dev br0 port br0 grp ff02::1:ff00:0 temp vid 1   
dev br0 port br0 grp ff02::2 temp vid 1   
dev br0 port br0 grp ff02::1:ff08:1 temp vid 1   
dev br0 port br0 grp ff02::6a temp vid 1   
Bond / Link Aggregation  
The bonding interface provides a method for aggregating multiple network interfaces into a single logical “bonded” interface, or LAG, or ether-channel, or port-channel. The behavior of the bonded interfaces depends upon the mode; generally speaking, modes provide either hot standby or load balancing services. Additionally, link integrity monitoring may be performed.

Configuration

Common interface configuration  
 set interfaces bonding <interface> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64  
• dhcp interface address is received by DHCP from a DHCP server on this segment.• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/   
 /config/scripts/dhcp-client/post-hooks.d/•  
Example:   
set interfaces bonding bond0 address 192.0.2.1/24   
set interfaces bonding bond0 address 2001:db8::1/64   
set interfaces bonding bond0 address dhcp   
set interfaces bonding bond0 address dhcpv6  
 set interfaces bonding <interface> description <description>   
Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces bonding bond0 description 'This is an awesome interface running on VyOS' set interfaces bonding <interface> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces bonding bond0 disable  
 set interfaces bonding <interface> disable-flow-control   
Ethernet flow control is a mechanism for temporarily stopping the transmission of data on Ethernet family computer networks. The goal of this mechanism is to ensure zero packet loss in the presence of network congestion.

The first flow control mechanism, the pause frame, was defined by the IEEE 802.3x standard. A sending station (computer or network switch) may be transmitting data faster than the other end of the link can accept it. Using flow control, the receiving station can signal the sender requesting suspension of transmissions until the receiver catches up.

Use this command to disable the generation of Ethernet flow control (pause frames).

Example:   
set interfaces bonding bond0 disable-flow-control  
 set interfaces bonding <interface> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces bonding bond0 disable-link-detect  
 set interfaces bonding <interface> mac <xx:xx:xx:xx:xx:xx>

Configure user defined MAC address on given <interface>.

Example:   
set interfaces bonding bond0 mac '00:01:02:03:04:05'  
 set interfaces bonding <interface> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces bonding bond0 mtu 1600  
 set interfaces bonding <interface> ip adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bonding <interface> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces bonding bond0 ip arp-cache-timeout 180  
 set interfaces bonding <interface> ip disable-arp-filter   
If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces bonding bond0 ip disable-arp-filter  
 set interfaces bonding <interface> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces bonding bond0 ip disable-forwarding  
 set interfaces bonding <interface> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces bonding bond0 ip enable-directed-broadcast  
 set interfaces bonding <interface> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces bonding bond0 ip enable-arp-accept  
 set interfaces bonding <interface> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces bonding bond0 ip enable-arp-announce  
 set interfaces bonding <interface> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:   
If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces bonding bond0 ip enable-arp-ignore  
 set interfaces bonding <interface> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces bonding bond0 ip enable-proxy-arp  
 set interfaces bonding <interface> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:   
 In RFC 3069 it is called VLAN Aggregation•  
• Cisco and Allied Telesyn call it Private VLAN  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces bonding <interface> ip source-validation <strict | loose | disable>   
Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

loose: Each incoming packet’s source address is also tested against the FIB and if the •  
source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces bonding <interface> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:   
set interfaces bonding bond0 ipv6 address autoconf  
 set interfaces bonding <interface> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces bonding bond0 ipv6 address eui64 2001:db8:beef::/64 set interfaces bonding <interface> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces bonding bond0 ipv6 address no-default-link-local  
 set interfaces bonding <interface> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces bonding bond0 ipv6 disable-forwarding

set interfaces bonding <interface> ipv6 adjust-mss <mss | clamp-mss-to-pmtu>   
As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bonding <interface> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

0: Disable DAD•  
• 1: Enable DAD (default)  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces bonding bond0 ipv6 accept-dad 2  
 set interfaces bonding <interface> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces bonding bond0 ipv6 dup-addr-detect-transmits 5 set interfaces bonding <interface> vrf <vrf>   
Place interface in given VRF instance.

See also   
There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces bonding bond0 vrf red   
DHCP(v6)  
 set interfaces bonding <interface> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces bonding bond0 dhcp-options client-id 'foo-bar'  
 set interfaces bonding <interface> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces bonding bond0 dhcp-options host-name 'VyOS'  
 set interfaces bonding <interface> dhcp-options vendor-class-id <vendor-id>   
The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces bonding bond0 dhcp-options vendor-class-id 'VyOS'  
 set interfaces bonding <interface> dhcp-options no-default-route   
Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces bonding bond0 dhcp-options no-default-route  
 set interfaces bonding <interface> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces bonding bond0 dhcp-options default-route-distance 220  
 set interfaces bonding <interface> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces bonding bond0 dhcp-options reject 192.168.100.0/24  
 set interfaces bonding <interface> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server.

It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces bonding bond0 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d'  
 set interfaces bonding <interface> dhcpv6-options no-release   
When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces bonding bond0 dhcpv6-options no-release  
 set interfaces bonding <interface> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces bonding bond0 dhcpv6-options parameters-only  
 set interfaces bonding <interface> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces bonding bond0 dhcpv6-options rapid-commit  
 set interfaces bonding <interface> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces bonding bond0 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces bonding <interface> dhcpv6-options pd <id> length <length>   
Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces bonding bond0 dhcpv6-options pd 0 length 56  
 set interfaces bonding <interface> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces bonding bond0 dhcpv6-options pd 0 interface eth8 address 65534  
 set interfaces bonding <interface> dhcpv6-options pd <id> interface <delegatee> sla-id <id> Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces bonding bond0 dhcpv6-options pd 0 interface eth8 sla-id 1 Member Interfaces  
 set interfaces bonding <interface> member interface <member>   
Enslave <member> interface to bond <interface>.

Bond options  
 set interfaces bonding <interface> mode <802.3ad | active-backup | broadcast | round-robin | transmit-load-balance | adaptive-load-balance | xor-hash>   
Specifies one of the bonding policies. The default is 802.3ad. Possible values are:  
• 802.3ad - IEEE 802.3ad Dynamic link aggregation. Creates aggregation groups that share the same speed and duplex settings. Utilizes all slaves in the active aggregator according to the 802.3ad specification.

Slave selection for outgoing traffic is done according to the transmit hash policy, which may be changed from the default simple XOR policy via the hash-policy option, documented below.

Note   
Not all transmit policies may be 802.3ad compliant, particularly in regards to the packet misordering requirements of section 43.2.4 of the 802.3ad standard.

• active-backup - Active-backup policy: Only one slave in the bond is active. A different slave becomes active if, and only if, the active slave fails. The bond’s MAC address is externally visible on only one port (network adapter) to avoid confusing the switch.

When a failover occurs in active-backup mode, bonding will issue one or more gratuitous ARPs on the newly active slave. One gratuitous ARP is issued for the bonding master interface and each VLAN interfaces configured above it, provided that the interface has at least one IP address configured. Gratuitous ARPs issued for VLAN interfaces are tagged with the appropriate VLAN id.

This mode provides fault tolerance. The primary option, documented below, affects the behavior of this mode.

• broadcast - Broadcast policy: transmits everything on all slave interfaces.

This mode provides fault tolerance.

• round-robin - Round-robin policy: Transmit packets in sequential order from the first available slave through the last.

This mode provides load balancing and fault tolerance.

• transmit-load-balance - Adaptive transmit load balancing: channel bonding that does not require any special switch support.

Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed receiving slave.

adaptive-load-balance - Adaptive load balancing: includes transmit-load-balance plus •  
receive load balancing for IPV4 traffic, and does not require any special switch support. The receive load balancing is achieved by ARP negotiation. The bonding driver intercepts the ARP Replies sent by the local system on their way out and overwrites the source hardware address with the unique hardware address of one of the slaves in the bond such that different peers use different hardware addresses for the server.

Receive traffic from connections created by the server is also balanced. When the local system sends an ARP Request the bonding driver copies and saves the peer’s IP information from the ARP packet. When the ARP Reply arrives from the peer, its hardware address is retrieved and the bonding driver initiates an ARP reply to this peer assigning it to one of the slaves in the bond. A problematic outcome of using ARP negotiation for balancing is that each time that an ARP request is broadcast it uses the hardware address of the bond. Hence, peers learn the hardware address of the bond and the balancing of receive traffic collapses to the current slave. This is handled by sending updates (ARP Replies) to all the peers with their individually assigned hardware address such that the traffic is redistributed. Receive traffic is also redistributed when a new slave is added to the bond and when an inactive slave is re-activated. The receive load is distributed sequentially (round robin) among the group of highest speed slaves in the bond.

When a link is reconnected or a new slave joins the bond the receive traffic is redistributed among all active slaves in the bond by initiating ARP Replies with the selected MAC address to each of the clients. The updelay parameter (detailed below) must be set to a value equal or greater than the switch’s forwarding delay so that the ARP Replies sent to the peers will not be blocked by the switch.

• xor-hash - XOR policy: Transmit based on the selected transmit hash policy. The default policy is a simple [(source MAC address XOR’d with destination MAC address XOR packet type

ID) modulo slave count]. Alternate transmit policies may be selected via the hash-policy option, described below.

This mode provides load balancing and fault tolerance.

set interfaces bonding <interface> min-links <0-16>   
Specifies the minimum number of links that must be active before asserting carrier. It is similar to the Cisco EtherChannel min-links feature. This allows setting the minimum number of member ports that must be up (link-up state) before marking the bond device as up (carrier on).

This is useful for situations where higher level services such as clustering want to ensure a minimum number of low bandwidth links are active before switchover.

This option only affects 802.3ad mode.

The default value is 0. This will cause the carrier to be asserted (for 802.3ad mode) whenever there is an active aggregator, regardless of the number of available links in that aggregator.

Note   
Because an aggregator cannot be active without at least one available link, setting this option to 0 or to 1 has the exact same effect.

set interfaces bonding <interface> lacp-rate <slow|fast>   
Option specifying the rate in which we’ll ask our link partner to transmit LACPDU packets in 802.3ad mode.

This option only affects 802.3ad mode.

• slow: Request partner to transmit LACPDUs every 30 seconds• fast: Request partner to transmit LACPDUs every 1 second The default value is slow.

set interfaces bonding <interface> hash-policy <policy>  
• layer2 - Uses XOR of hardware MAC addresses and packet type ID field to generate the hash. The formula is  
• hash = source MAC XOR destination MAC XOR packet type ID  
• slave number = hash modulo slave count   
This algorithm will place all traffic to a particular network peer on the same slave.

This algorithm is 802.3ad compliant.

• layer2+3 - This policy uses a combination of layer2 and layer3 protocol information to generate the hash. Uses XOR of hardware MAC addresses and IP addresses to generate the hash. The formula is:  
• hash = source MAC XOR destination MAC XOR packet type ID  
• hash = hash XOR source IP XOR destination IP  
• hash = hash XOR (hash RSHIFT 16)  
• hash = hash XOR (hash RSHIFT 8)   
And then hash is reduced modulo slave count.

If the protocol is IPv6 then the source and destination addresses are first hashed using ipv6\_addr\_hash.

This algorithm will place all traffic to a particular network peer on the same slave. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy.

This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations.

This algorithm is 802.3ad compliant.

• layer3+4 - This policy uses upper layer protocol information, when available, to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.

The formula for unfragmented TCP and UDP packets is hash = source port, destination port (as in the header) hash = hash XOR source IP XOR destination IP   
hash = hash XOR (hash RSHIFT 16)   
hash = hash XOR (hash RSHIFT 8)   
And then hash is reduced modulo slave count.

If the protocol is IPv6 then the source and destination addresses are first hashed using ipv6\_addr\_hash.

For fragmented TCP or UDP packets and all other IPv4 and IPv6 protocol traffic, the source and destination port information is omitted. For non-IP traffic, the formula is the same as for the layer2 transmit hash policy.

This algorithm is not fully 802.3ad compliant. A single TCP or UDP conversation containing both fragmented and unfragmented packets will see packets striped across two interfaces. This may result in out of order delivery. Most traffic types will not meet these criteria, as TCP rarely fragments traffic, and most UDP traffic is not involved in extended conversations. Other implementations of 802.3ad may or may not tolerate this noncompliance.

set interfaces bonding <interface> primary <interface>   
An <interface> specifying which slave is the primary device. The specified device will always be the active slave while it is available. Only when the primary is off-line will alternate devices be used. This is useful when one slave is preferred over another, e.g., when one slave has higher throughput than another.

The primary option is only valid for active-backup, transmit-load-balance, and adaptive-load-balance mode.

set interfaces bonding <interface> arp-monitor interval <time> Specifies the ARP link monitoring <time> in seconds.

The ARP monitor works by periodically checking the slave devices to determine whether they have sent or received traffic recently (the precise criteria depends upon the bonding mode, and the state of the slave). Regular traffic is generated via ARP probes issued for the addresses specified by the arp-monitor target option.

If ARP monitoring is used in an etherchannel compatible mode (modes round-robin and xor-hash), the switch should be configured in a mode that evenly distributes packets across all links.

If the switch is configured to distribute the packets in an XOR fashion, all replies from the ARP targets will be received on the same link which could cause the other team members to fail. A value of 0 disables ARP monitoring. The default value is 0.

set interfaces bonding <interface> arp-monitor target <address>   
Specifies the IP addresses to use as ARP monitoring peers when arp-monitor interval option is > 0. These are the targets of the ARP request sent to determine the health of the link to the targets.

Multiple target IP addresses can be specified. At least one IP address must be given for ARP monitoring to function.

The maximum number of targets that can be specified is 16. The default value is no IP address.

VLAN  
IEEE 802.1q, often referred to as Dot1q, is the networking standard that supports virtual LANs (VLANs) on an IEEE 802.3 Ethernet network. The standard defines a system of VLAN tagging for Ethernet frames and the accompanying procedures to be used by bridges and switches in handling such frames. The standard also contains provisions for a quality-of-service   
prioritization scheme commonly known as IEEE 802.1p and defines the Generic Attribute Registration Protocol.

Portions of the network which are VLAN-aware (i.e., IEEE 802.1q conformant) can include VLAN tags. When a frame enters the VLAN-aware portion of the network, a tag is added to represent the VLAN membership. Each frame must be distinguishable as being within exactly one VLAN. A frame in the VLAN-aware portion of the network that does not contain a VLAN tag is assumed to be flowing on the native VLAN.

The standard was developed by IEEE 802.1, a working group of the IEEE 802 standards committee, and continues to be actively revised. One of the notable revisions is 802.1Q-2014 which incorporated IEEE 802.1aq (Shortest Path Bridging) and much of the IEEE 802.1d standard.

802.1q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is vif.

set interfaces bonding <interface> vif <vlan-id>   
Create a new VLAN interface on interface <interface> using the VLAN number provided via <vlan-id>.

You can create multiple VLAN interfaces on a physical interface. The VLAN ID range is from 0 to 4094.

Note   
Only 802.1Q-tagged packets are accepted on Ethernet vifs.

set interfaces bonding <interface> vif <vlan-id> address <address | dhcp | dhcpv6> Configure interface <interface> with one or more interface addresses.

• address can be specified multiple times as IPv4 and/or IPv6 address, e.g. 192.0.2.1/24 and/or 2001:db8::1/64   
 dhcp interface address is received by DHCP from a DHCP server on this segment.•  
• dhcpv6 interface address is received by DHCPv6 from a DHCPv6 server on this segment.

Note   
When using DHCP to retrieve IPv4 address and if local customizations are needed, they should be possible using the enter and exit hooks provided. The hook dirs are:  
• /config/scripts/dhcp-client/pre-hooks.d/  
• /config/scripts/dhcp-client/post-hooks.d/   
Example:   
set interfaces bonding bond0 vif 10 address 192.0.2.1/24   
set interfaces bonding bond0 vif 10 address 2001:db8::1/64   
set interfaces bonding bond0 vif 10 address dhcp   
set interfaces bonding bond0 vif 10 address dhcpv6  
 set interfaces bonding <interface> vif <vlan-id> description <description>

Set a human readable, descriptive alias for this connection. Alias is used by e.g. the show interfaces command or SNMP based monitoring tools.

Example:   
set interfaces bonding bond0 vif 10 description 'This is an awesome interface running on VyOS' set interfaces bonding <interface> vif <vlan-id> disable   
Disable given <interface>. It will be placed in administratively down (A/D) state.

Example:   
set interfaces bonding bond0 vif 10 disable  
 set interfaces bonding <interface> vif <vlan-id> disable-link-detect   
Use this command to direct an interface to not detect any physical state changes on a link, for example, when the cable is unplugged.

Default is to detects physical link state changes.

Example:   
set interfaces bonding bond0 vif 10 disable-link-detect  
 set interfaces bonding <interface> vif <vlan-id> mac <xx:xx:xx:xx:xx:xx> Configure user defined MAC address on given <interface>.

Example:   
set interfaces bonding bond0 vif 10 mac '00:01:02:03:04:05'  
 set interfaces bonding <interface> vif <vlan-id> mtu <mtu>   
Configure MTU on given <interface>. It is the size (in bytes) of the largest ethernet frame sent on this link.

Example:   
set interfaces bonding bond0 vif 10 mtu 1600  
 set interfaces bonding <interface> vif <vlan-id> ip adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss <value>   
Hint   
MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bonding <interface> vif <vlan-id> ip arp-cache-timeout   
Once a neighbor has been found, the entry is considered to be valid for at least for this specific time. An entry’s validity will be extended if it receives positive feedback from higher level protocols.

This defaults to 30 seconds.

Example:   
set interfaces bonding bond0 vif 10 ip arp-cache-timeout 180 set interfaces bonding <interface> vif <vlan-id> ip disable-arp-filter

If set the kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

If not set (default) allows you to have multiple network interfaces on the same subnet, and have the ARPs for each interface be answered based on whether or not the kernel would route a packet from the ARP’d IP out that interface (therefore you must use source based routing for this to work).

In other words it allows control of which cards (usually 1) will respond to an arp request.

Example:   
set interfaces bonding bond0 vif 10 ip disable-arp-filter  
 set interfaces bonding <interface> vif <vlan-id> ip disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

set interfaces bonding bond0 vif 10 ip disable-forwarding  
 set interfaces bonding <interface> vif <vlan-id> ip enable-directed-broadcast   
Define different modes for IP directed broadcast forwarding as described in RFC 1812 and RFC 2644.

If configured, incoming IP directed broadcast packets on this interface will be forwarded.

If this option is unset (default), incoming IP directed broadcast packets will not be forwarded.

set interfaces bonding bond0 vif 10 ip enable-directed-broadcast  
 set interfaces bonding <interface> vif <vlan-id> ip enable-arp-accept   
Define behavior for gratuitous ARP frames who’s IP is not already present in the ARP table. If configured create new entries in the ARP table.

Both replies and requests type gratuitous arp will trigger the ARP table to be updated, if this setting is on.

If the ARP table already contains the IP address of the gratuitous arp frame, the arp table will be updated regardless if this setting is on or off.

set interfaces bonding bond0 vif 10 ip enable-arp-accept  
 set interfaces bonding <interface> vif <vlan-id> ip enable-arp-announce   
Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface.

Use any local address, configured on any interface if this is not set.

If configured, try to avoid local addresses that are not in the target’s subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.

set interfaces bonding bond0 vif 10 ip enable-arp-announce  
 set interfaces bonding <interface> vif <vlan-id> ip enable-arp-ignore   
Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:

If configured, reply only if the target IP address is local address configured on the incoming interface.

If this option is unset (default), reply for any local target IP address, configured on any interface.

set interfaces bonding bond0 vif 10 ip enable-arp-ignore  
 set interfaces bonding <interface> vif <vlan-id> ip enable-proxy-arp   
Use this command to enable proxy Address Resolution Protocol (ARP) on this interface. Proxy ARP allows an Ethernet interface to respond with its own MAC address to ARP requests for destination IP addresses on subnets attached to other interfaces on the system. Subsequent packets sent to those destination IP addresses are forwarded appropriately by the system.

Example:   
set interfaces bonding bond0 vif 10 ip enable-proxy-arp  
 set interfaces bonding <interface> vif <vlan-id> ip proxy-arp-pvlan   
Private VLAN proxy arp. Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy\_arp’ing.

Note   
Does not need to be used together with proxy\_arp.

This technology is known by different names:  
• In RFC 3069 it is called VLAN Aggregation   
 Cisco and Allied Telesyn call it Private VLAN•  
• Hewlett-Packard call it Source-Port filtering or port-isolation  
•  
 Ericsson call it MAC-Forced Forwarding (RFC Draft) set interfaces bonding <interface> vif <vlan-id> ip source-validation <strict | loose | disable> Enable policy for source validation by reversed path, as specified in RFC 3704. Current recommended practice in RFC 3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended.

• strict: Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.

• loose: Each incoming packet’s source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.

• disable: No source validation  
 set interfaces bonding <interface> vif <vlan-id> ipv6 address autoconf   
SLAAC RFC 4862. IPv6 hosts can configure themselves automatically when connected to an IPv6 network using the Neighbor Discovery Protocol via ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; routers respond to such a request with a router advertisement packet that contains Internet Layer configuration parameters.

Note   
This method automatically disables IPv6 traffic forwarding on the interface in question.

Example:

set interfaces bonding bond0 vif 10 ipv6 address autoconf  
 set interfaces bonding <interface> vif <vlan-id> ipv6 address eui64 <prefix>   
EUI-64 as specified in RFC 4291 allows a host to assign iteslf a unique 64-Bit IPv6 address.

Example:   
set interfaces bonding bond0 vif 10 ipv6 address eui64 2001:db8:beef::/64 set interfaces bonding <interface> vif <vlan-id> ipv6 address no-default-link-local Do not assign a link-local IPv6 address to this interface.

Example:   
set interfaces bonding bond0 vif 10 ipv6 address no-default-link-local  
 set interfaces bonding <interface> vif <vlan-id> ipv6 disable-forwarding   
Configure interface-specific Host/Router behaviour. If set, the interface will switch to host mode and IPv6 forwarding will be disabled on this interface.

Example:   
set interfaces bonding bond0 vif 10 ipv6 disable-forwarding  
 set interfaces bonding <interface> vif <vlan-id> ipv6 adjust-mss <mss | clamp-mss-to-pmtu> As Internet wide PMTU discovery rarely works, we sometimes need to clamp our TCP MSS value to a specific value. This is a field in the TCP options part of a SYN packet. By setting the MSS value, you are telling the remote side unequivocally ‘do not try to send me packets bigger than this value’.

Note   
This command was introduced in VyOS 1.4 - it was previously called: set firewall options interface <name> adjust-mss6 <value>   
Hint   
MSS value = MTU - 40 (IPv6 header) - 20 (TCP header), resulting in 1432 bytes on a 1492 byte MTU.

Instead of a numerical MSS value clamp-mss-to-pmtu can be used to automatically set the proper value.

set interfaces bonding <interface> vif <vlan-id> ipv6 accept-dad <1-3> Whether to accept DAD (Duplicate Address Detection).

• 0: Disable DAD   
 1: Enable DAD (default)•  
• 2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

Example:   
set interfaces bonding bond0 vif 10 ipv6 accept-dad 2  
 set interfaces bonding <interface> vif <vlan-id> ipv6 dup-addr-detect-transmits <n> The amount of Duplicate Address Detection probes to send.

Default: 1   
Example:   
set interfaces bonding bond0 vif 10 ipv6 dup-addr-detect-transmits 5 set interfaces bonding <interface> vif <vlan-id> vrf <vrf>   
Place interface in given VRF instance.

See also

There is an entire chapter about how to configure a VRF, please check this for additional information.

Example:   
set interfaces bonding bond0 vif 10 vrf red   
DHCP(v6)  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options client-id <description>   
RFC 2131 states: The client MAY choose to explicitly provide the identifier through the ‘client identifier’ option. If the client supplies a ‘client identifier’, the client MUST use the same ‘client identifier’ in all subsequent messages, and the server MUST use that identifier to identify the client.

Example:   
set interfaces bonding bond0 vif 10 dhcp-options client-id 'foo-bar'  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options host-name <hostname>   
Instead of sending the real system hostname to the DHCP server, overwrite the host-name with this given-value.

Example:   
set interfaces bonding bond0 vif 10 dhcp-options host-name 'VyOS'  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options vendor-class-id <vendor-id> The vendor-class-id option can be used to request a specific class of vendor options from the server.

Example:   
set interfaces bonding bond0 vif 10 dhcp-options vendor-class-id 'VyOS'  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options no-default-route Only request an address from the DHCP server but do not request a default gateway.

Example:   
set interfaces bonding bond0 vif 10 dhcp-options no-default-route  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options default-route-distance <distance> Set the distance for the default gateway sent by the DHCP server.

Example:   
set interfaces bonding bond0 vif 10 dhcp-options default-route-distance 220  
 set interfaces bonding <interface> vif <vlan-id> dhcp-options reject <address>   
Reject DHCP leases from a given address or range. This is useful when a modem gives a local IP when first starting.

• address can be specified multiple times, e.g. 192.168.100.1 and/or 192.168.100.0/24 Example:   
set interfaces bonding bond0 vif 10 dhcp-options reject 192.168.100.0/24  
 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options duid <duid>   
The DHCP unique identifier (DUID) is used by a client to get an IP address from a DHCPv6 server. It has a 2-byte DUID type field, and a variable-length identifier field up to 128 bytes. Its actual length depends on its type. The server compares the DUID with its database and delivers configuration data (address, lease times, DNS servers, etc.) to the client.

set interfaces bonding bond0 vif 10 duid '0e:00:00:01:00:01:27:71:db:f0:00:50:56:bf:c5:6d' set interfaces bonding <interface> vif <vlan-id> dhcpv6-options no-release

When no-release is specified, dhcp6c will send a release message on client exit to prevent losing an assigned address or prefix.

set interfaces bonding bond0 vif 10 dhcpv6-options no-release  
 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options parameters-only   
This statement specifies dhcp6c to only exchange informational configuration parameters with servers. A list of DNS server addresses is an example of such parameters. This statement is useful when the client does not need stateful configuration parameters such as IPv6 addresses or prefixes.

set interfaces bonding bond0 vif 10 dhcpv6-options parameters-only  
 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options rapid-commit   
When rapid-commit is specified, dhcp6c will include a rapid-commit option in solicit messages and wait for an immediate reply instead of advertisements.

set interfaces bonding bond0 vif 10 dhcpv6-options rapid-commit  
 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options temporary   
Request only a temporary address and not form an IA\_NA (Identity Association for Non-temporary Addresses) partnership.

set interfaces bonding bond0 vif 10 dhcpv6-options temporary   
DHCPv6 Prefix Delegation (PD)   
VyOS 1.3 (equuleus) supports DHCPv6-PD (RFC 3633). DHCPv6 Prefix Delegation is supported by most ISPs who provide native IPv6 for consumers on fixed networks.

set interfaces bonding <interface> vif <vlan-id> dhcpv6-options pd <id> length <length> Some ISPs by default only delegate a /64 prefix. To request for a specific prefix size use this option to request for a bigger delegation for this pd <id>. This value is in the range from 32 - 64 so you could request up to a /32 prefix (if your ISP allows this) down to a /64 delegation. The default value corresponds to 64.

To request a /56 prefix from your ISP use:   
set interfaces bonding bond0 vif 10 dhcpv6-options pd 0 length 56  
 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> address <address>   
Specify the interface address used locally on the interface where the prefix has been delegated to. ID must be a decimal integer.

It will be combined with the delegated prefix and the sla-id to form a complete interface address. The default is to use the EUI-64 address of the interface.

Example: Delegate a /64 prefix to interface eth8 which will use a local address on this router of <prefix>::ffff, as the address 65534 will correspond to ffff in hexadecimal notation.

set interfaces bonding bond0 vif 10 dhcpv6-options pd 0 interface eth8 address 65534 set interfaces bonding <interface> vif <vlan-id> dhcpv6-options pd <id> interface <delegatee> sla-id <id>   
Specify the identifier value of the site-level aggregator (SLA) on the interface. ID must be a decimal number greater then 0 which fits in the length of SLA IDs (see below).

Example: If ID is 1 and the client is delegated an IPv6 prefix 2001:db8:ffff::/48, dhcp6c will combine the two values into a single IPv6 prefix, 2001:db8:ffff:1::/64, and will configure the prefix on the specified interface.

set interfaces bonding bond0 vif 10 dhcpv6-options pd 0 interface eth8 sla-id 1

Port Mirror (SPAN)  
SPAN port mirroring can copy the inbound/outbound traffic of the interface to the specified interface, usually the interface can be connected to some special equipment, such as behavior control system, intrusion detection system and traffic collector, and can copy all related traffic from this port. The benefit of mirroring the traffic is that the application is isolated from the source traffic and so application processing does not affect the traffic or the system   
performance.

VyOS uses the mirror option to configure port mirroring. The configuration is divided into 2 different directions. Destination ports should be configured for different traffic directions.

set interfaces bondinging <interface> mirror ingress <monitor-interface>   
Configure port mirroring for interface inbound traffic and copy the traffic to monitor-interface Example: Mirror the inbound traffic of bond1 port to eth3   
set interfaces bondinging bond1 mirror ingress eth3  
 set interfaces bondinging <interface> mirror egress <monitor-interface>   
Configure port mirroring for interface outbound traffic and copy the traffic to monitor-interface Example: Mirror the outbound traffic of bond1 port to eth3   
set interfaces bondinging bond1 mirror egress eth3   
Example  
The following configuration on VyOS applies to all following 3rd party vendors. It creates a bond with two links and VLAN 10, 100 on the bonded interfaces with a per VIF IPv4 address.

# Create bonding interface bond0 with 802.3ad LACP set interfaces bonding bond0 hash-policy 'layer2' set interfaces bonding bond0 mode '802.3ad'

# Add the required vlans and IPv4 addresses on them   
set interfaces bonding bond0 vif 10 address 192.168.0.1/24 set interfaces bonding bond0 vif 100 address 10.10.10.1/24

# Add the member interfaces to the bonding interface   
set interfaces bonding bond0 member interface eth1   
set interfaces bonding bond0 member interface eth2   
Note   
If you happen to run this in a virtual environment like by EVE-NG you need to ensure your VyOS NIC is set to use the e1000 driver. Using the default virtio-net-pci or the vmxnet3 driver will not work. ICMP messages will not be properly processed. They are visible on the virtual wire but will not make it fully up the networking stack.

You can check your NIC driver by issuing show interfaces ethernet eth0 physical | grep -i driver Cisco Catalyst  
Assign member interfaces to PortChannel   
interface GigabitEthernet1/0/23  
 description VyOS eth1  
 channel-group 1 mode active   
!

interface GigabitEthernet1/0/24

description VyOS eth2  
 channel-group 1 mode active   
!

A new interface becomes present Port-channel1, all configuration like allowed VLAN interfaces, STP will happen here.

interface Port-channel1  
 description LACP Channel for VyOS  
 switchport trunk encapsulation dot1q  
 switchport trunk allowed vlan 10,100  
 switchport mode trunk  
 spanning-tree portfast trunk   
!

Juniper EX Switch  
For a headstart you can use the below example on how to build a bond with two interfaces from VyOS to a Juniper EX Switch system.

# Create aggregated ethernet device with 802.3ad LACP and port speeds of 10gbit/s set interfaces ae0 aggregated-ether-options link-speed 10g   
set interfaces ae0 aggregated-ether-options lacp active

# Create layer 2 on the aggregated ethernet device with trunking for our vlans set interfaces ae0 unit 0 family ethernet-switching port-mode trunk

# Add the required vlans to the device   
set interfaces ae0 unit 0 family ethernet-switching vlan members 10 set interfaces ae0 unit 0 family ethernet-switching vlan members 100

# Add the two interfaces to the aggregated ethernet device, in this setup both # ports are on the same switch (switch 0, module 1, port 0 and 1)   
set interfaces xe-0/1/0 ether-options 802.3ad ae0   
set interfaces xe-0/1/1 ether-options 802.3ad ae0

# But this can also be done with multiple switches in a stack, a virtual   
# chassis on Juniper (switch 0 and switch 1, module 1, port 0 on both switches)   
set interfaces xe-0/1/0 ether-options 802.3ad ae0   
set interfaces xe-1/1/0 ether-options 802.3ad ae0   
Aruba/HP  
For a headstart you can use the below example on how to build a bond,port-channel with two interfaces from VyOS to a Aruba/HP 2510G switch.

# Create trunk with 2 member interfaces (interface 1 and 2) and LACP trunk 1-2 Trk1 LACP

# Add the required vlans to the trunk   
vlan 10 tagged Trk1   
vlan 100 tagged Trk1

Arista EOS  
When utilizing VyOS in an environment with Arista gear you can use this blue print as an initial setup to get an LACP bond / port-channel operational between those two devices.

Lets assume the following topology:

R1   
interfaces {  
 bonding bond10 {  
 hash-policy layer3+4  
 member {  
 interface eth1  
 interface eth2  
 }  
 mode 802.3ad  
 vif 100 {  
 address 192.0.2.1/30  
 address 2001:db8::1/64  
 }  
 }   
R2   
interfaces {  
 bonding bond10 {  
 hash-policy layer3+4  
 member {  
 interface eth1  
 interface eth2  
 }  
 mode 802.3ad  
 vif 100 {  
 address 192.0.2.2/30  
 address 2001:db8::2/64  
 }  
 }   
SW1   
!

vlan 100  
 name FOO   
!

interface Port-Channel10  
 switchport trunk allowed vlan 100  
 switchport mode trunk  
 spanning-tree portfast   
!

interface Port-Channel20

switchport mode trunk  
 no spanning-tree portfast auto  
 spanning-tree portfast network   
!

interface Ethernet1  
 channel-group 10 mode active   
!

interface Ethernet2  
 channel-group 10 mode active   
!

interface Ethernet3  
 channel-group 20 mode active   
!

interface Ethernet4  
 channel-group 20 mode active   
!

SW2   
!

vlan 100  
 name FOO   
!

interface Port-Channel10  
 switchport trunk allowed vlan 100  
 switchport mode trunk  
 spanning-tree portfast   
!

interface Port-Channel20  
 switchport mode trunk  
 no spanning-tree portfast auto  
 spanning-tree portfast network   
!

interface Ethernet1  
 channel-group 10 mode active   
!

interface Ethernet2  
 channel-group 10 mode active   
!

interface Ethernet3  
 channel-group 20 mode active   
!

interface Ethernet4  
 channel-group 20 mode active   
!

Note

When using EVE-NG to lab this environment ensure you are using e1000 as the desired driver for your VyOS network interfaces. When using the regular virtio network driver no LACP PDUs will be sent by VyOS thus the port-channel will never become active!

Operation  
 show interfaces bonding   
Show brief interface information.

vyos@vyos:~$ show interfaces bonding   
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
bond0 - u/u my-sw1 int 23 and 24   
bond0.10 192.168.0.1/24 u/u office-net   
bond0.100 10.10.10.1/24 u/u management-net  
 show interfaces bonding <interface>   
Show detailed information on given <interface>   
vyos@vyos:~$ show interfaces bonding bond5   
bond5: <NO-CARRIER,BROADCAST,MULTICAST,MASTER,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000  
 link/ether 00:50:56:bf:ef:aa brd ff:ff:ff:ff:ff:ff  
 inet6 fe80::e862:26ff:fe72:2dac/64 scope link tentative  
 valid\_lft forever preferred\_lft forever

RX: bytes packets errors dropped overrun mcast  
 0 0 0 0 0 0  
 TX: bytes packets errors dropped carrier collisions  
 0 0 0 0 0 0  
 show interfaces bonding <interface> detail   
Show detailed information about the underlaying physical links on given bond <interface>.

vyos@vyos:~$ show interfaces bonding bond5 detail Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation Transmit Hash Policy: layer2 (0)   
MII Status: down   
MII Polling Interval (ms): 100   
Up Delay (ms): 0   
Down Delay (ms): 0

802.3ad info   
LACP rate: slow   
Min links: 0   
Aggregator selection policy (ad\_select): stable

Slave Interface: eth1

MII Status: down   
Speed: Unknown   
Duplex: Unknown   
Link Failure Count: 0   
Permanent HW addr: 00:50:56:bf:ef:aa   
Slave queue ID: 0   
Aggregator ID: 1   
Actor Churn State: churned   
Partner Churn State: churned   
Actor Churned Count: 1   
Partner Churned Count: 1

Slave Interface: eth2   
MII Status: down   
Speed: Unknown   
Duplex: Unknown   
Link Failure Count: 0   
Permanent HW addr: 00:50:56:bf:19:26   
Slave queue ID: 0   
Aggregator ID: 2   
Actor Churn State: churned   
Partner Churn State: churned   
Actor Churned Count: 1   
Partner Churned Count: 1

RIP  
RIP is a widely deployed interior gateway protocol. RIP was developed in the 1970s at Xerox Labs as part of the XNS routing protocol. RIP is a distance-vector protocol and is based on the Bellman-Ford algorithms. As a distance-vector protocol, RIP router send updates to its neighbors periodically, thus allowing the convergence to a known topology. In each update, the distance to any given network will be broadcast to its neighboring router.

Supported versions of RIP are:   
 RIPv1 as described in RFC 1058•  
• RIPv2 as described in RFC 2453   
General Configuration  
 set protocols rip network <A.B.C.D/M>   
This command enables RIP and sets the RIP enable interface by NETWORK. The interfaces which have addresses matching with NETWORK are enabled.

set protocols rip interface <interface>   
This command specifies a RIP enabled interface by interface name. Both the sending and receiving of RIP packets will be enabled on the port specified in this command.

set protocols rip neighbor <A.B.C.D>   
This command specifies a RIP neighbor. When a neighbor doesn’t understand multicast, this command is used to specify neighbors. In some cases, not all routers will be able to understand

multicasting, where packets are sent to a network or a group of addresses. In a situation where a neighbor cannot process multicast packets, it is necessary to establish a direct link between routers.

set protocols rip passive-interface interface <interface>   
This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are processed as normal and VyOS does not send either multicast or unicast RIP packets except to RIP neighbors specified with neighbor command.

set protocols rip passive-interface interface default This command specifies all interfaces to passive mode.

Optional Configuration  
 set protocols rip default-distance <distance>   
This command change the distance value of RIP. The distance range is 1 to 255.

Note   
Routes with a distance of 255 are effectively disabled and not installed into the kernel.

set protocols rip network-distance <A.B.C.D/M> distance <distance>   
This command sets default RIP distance to a specified value when the routes source IP address matches the specified prefix.

set protocols rip network-distance <A.B.C.D/M> access-list <name>   
This command can be used with previous command to sets default RIP distance to specified value when the route source IP address matches the specified prefix and the specified access-list.

set protocols rip default-information originate   
This command generate a default route into the RIP.

set protocols rip distribute-list access-list <in|out> <number>   
This command can be used to filter the RIP path using access lists. in and out this is the direction in which the access lists are applied.

set protocols rip distribute-list interface <interface> access-list <in|out> <number> This command allows you apply access lists to a chosen interface to filter the RIP path.

set protocols rip distribute-list prefix-list <in|out> <name>   
This command can be used to filter the RIP path using prefix lists. in and out this is the direction in which the prefix lists are applied.

set protocols rip distribute-list interface <interface> prefix-list <in|out> <name> This command allows you apply prefix lists to a chosen interface to filter the RIP path.

set protocols rip route <A.B.C.D/M>   
This command is specific to FRR and VyOS. The route command makes a static route only inside RIP. This command should be used only by advanced users who are particularly knowledgeable about the RIP protocol. In most cases, we recommend creating a static route in VyOS and redistributing it in RIP using redistribute static.

set protocols rip timers update <seconds>   
This command specifies the update timer. Every update timer seconds, the RIP process is awakened to send an unsolicited response message containing the complete routing table to all neighboring RIP routers. The time range is 5 to 2147483647. The default value is 30 seconds.

set protocols rip timers timeout <seconds>

This command specifies the timeout timer. Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped. The time range is 5 to 2147483647. The default value is 180 seconds.

set protocols rip timers garbage-collection <seconds>   
This command specifies the garbage-collection timer. Upon expiration of the garbage-collection timer, the route is finally removed from the routing table. The time range is 5 to 2147483647. The default value is 120 seconds.

Redistribution Configuration  
 set protocols rip redistribute <route source>   
This command redistributes routing information from the given route source into the RIP tables. There are five modes available for route source: bgp, connected, kernel, ospf, static.

set protocols rip redistribute <route source> metric <metric>   
This command specifies metric for redistributed routes from the given route source. There are five modes available for route source: bgp, connected, kernel, ospf, static. The metric range is 1 to 16.

set protocols rip redistribute <route source> route-map <name>   
This command allows to use route map to filter redistributed routes from the given route source. There are five modes available for route source: bgp, connected, kernel, ospf, static.

set protocols rip default-metric <metric>   
This command modifies the default metric (hop count) value for redistributed routes. The metric range is 1 to 16. The default value is 1. This command does not affect connected route even if it is redistributed by redistribute connected. To modify connected routes metric value, please use redistribute connected metric.

Interfaces Configuration  
 set interfaces <inttype> <intname> ip rip authentication plaintext-password <text>   
This command sets the interface with RIP simple password authentication. This command also sets authentication string. The string must be shorter than 16 characters.

set interfaces <inttype> <intname> ip rip authentication md5 <id> password <text>   
This command sets the interface with RIP MD5 authentication. This command also sets MD5 Key. The key must be shorter than 16 characters.

set interfaces <inttype> <intname> ip rip split-horizon disable   
This command disables split-horizon on the interface. By default, VyOS does not advertise RIP routes out the interface over which they were learned (split horizon).3  
 set interfaces <inttype> <intname> ip rip split-horizon poison-reverse   
This command enables poison-reverse on the interface. If both poison reverse and split horizon are enabled, then VyOS advertises the learned routes as unreachable over the interface on which the route was learned.

Operational Mode Commands  
 show ip rip   
This command displays RIP routes.

Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP Sub-codes:  
 (n) - normal, (s) - static, (d) - default, (r) - redistribute,

(i) - interface

Network Next Hop Metric From Tag Time   
C(i) 10.0.12.0/24 0.0.0.0 1 self 0   
C(i) 10.0.13.0/24 0.0.0.0 1 self 0   
R(n) 10.0.23.0/24 10.0.12.2 2 10.0.12.2 0 02:53  
 show ip rip status   
The command displays current RIP status. It includes RIP timer, filtering, version, RIP enabled interface and RIP peer information.

Routing Protocol is "rip"  
 Sending updates every 30 seconds with +/-50%, next due in 11 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 any version  
 Interface Send Recv Key-chain  
 eth0 2 1 2  
 eth2 2 1 2  
 Routing for Networks:  
 10.0.12.0/24  
 eth0  
 Routing Information Sources:  
 Gateway BadPackets BadRoutes Distance Last Update  
 10.0.12.2 0 0 120 00:00:11  
 Distance: (default is 120)   
Configuration Example  
Simple RIP configuration using 2 nodes and redistributing connected interfaces.

Node 1:   
set interfaces loopback address 10.1.1.1/32   
set protocols rip network 192.168.0.0/24   
set protocols rip redistribute connected   
Node 2:   
set interfaces loopback address 10.2.2.2/32   
set protocols rip network 192.168.0.0/24   
set protocols rip redistribute connected   
RPKI  
There are two types of Network Admins who deal with BGP, those who have created an international incident and/or outage, and those who are lying  
—tweet by EvilMog, 2020-02-21   
RPKI is a framework PKI designed to secure the Internet routing infrastructure. It associates BGP route announcements with the correct originating ASN which BGP routers can then use to check each route against the corresponding ROA for validity. RPKI is described in RFC 6480.

A BGP-speaking router like VyOS can retrieve ROA information from RPKI “Relying Party software” (often just called an “RPKI server” or “RPKI validator”) by using RTR protocol. There are several open source implementations to choose from, such as NLNetLabs’ Routinator (written in Rust), Cloudflare’s GoRTR and OctoRPKI (written in Go), and RIPE NCC’s RPKI Validator (written in Java). The RTR protocol is described in RFC 8210.

Tip   
If you are new to these routing security technologies then there is an excellent guide to RPKI by NLnet Labs which will get you up to speed very quickly. Their documentation explains   
everything from what RPKI is to deploying it in production. It also has some help and   
operational guidance including “What can I do about my route having an Invalid state?”Getting started  
First you will need to deploy an RPKI validator for your routers to use. The RIPE NCC helpfully provide some instructions to get you started with several different options. Once your server is running you can start validating announcements.

Imported prefixes during the validation may have values:   
valid   
The prefix and ASN that originated it match a signed ROA. These are probably trustworthy route announcements.

invalid   
The prefix or prefix length and ASN that originated it doesn’t match any existing ROA. This could be the result of a prefix hijack, or merely a misconfiguration, but should probably be treated as untrustworthy route announcements.

notfound   
No ROA exists which covers that prefix. Unfortunately this is the case for about 80% of the IPv4 prefixes which were announced to the DFZ at the start of 2020   
Note   
If you are responsible for the global addresses assigned to your network, please make sure that your prefixes have ROAs associated with them to avoid being notfound by RPKI. For most ASNs this will involve publishing ROAs via your RIR (RIPE NCC, APNIC, ARIN, LACNIC or AFRINIC), and is something you are encouraged to do whenever you plan to announce addresses into the DFZ. Particularly large networks may wish to run their own RPKI certificate authority and publication server instead of publishing ROAs via their RIR. This is a subject far beyond the scope of VyOS’ documentation. Consider reading about Krill if this is a rabbit hole you need or especially want to dive down.

Features of the Current Implementation  
In a nutshell, the current implementation provides the following features:  
• The BGP router can connect to one or more RPKI cache servers to receive validated prefix to origin AS mappings. Advanced failover can be implemented by server sockets with different preference values.

• If no connection to an RPKI cache server can be established after a pre-defined timeout, the router will process routes without prefix origin validation. It still will try to establish a connection to an RPKI cache server in the background.

• By default, enabling RPKI does not change best path selection. In particular, invalid prefixes will still be considered during best path selection. However, the router can be configured to ignore all invalid prefixes.

• Route maps can be configured to match a specific RPKI validation state. This allows the creation of local policies, which handle BGP routes based on the outcome of the Prefix Origin Validation.

• Updates from the RPKI cache servers are directly applied and path selection is updated accordingly. (Soft reconfiguration must be enabled for this to work).

Configuration  
 set protocols rpki polling-period <1-86400>   
Define the time interval to update the local cache The default value is 300 seconds.

set protocols rpki cache <address> port <port>   
Defined the IPv4, IPv6 or FQDN and port number of the caching RPKI caching instance which is used.

This is a mandatory setting.

set protocols rpki cache <address> preference <preference>   
Multiple RPKI caching instances can be supplied and they need a preference in which their result sets are used.

This is a mandatory setting.

SSH  
Connections to the RPKI caching server can not only be established by HTTP/TLS but you can also rely on a secure SSH session to the server. To enable SSH, first you need to create an SSH client keypair using generate ssh client-key /config/auth/id\_rsa\_rpki. Once your key is created you can setup the connection.

set protocols rpki cache <address> ssh username <user>   
SSH username to establish an SSH connection to the cache server.

set protocols rpki cache <address> ssh known-hosts-file <filepath> Local path that includes the known hosts file.

set protocols rpki cache <address> ssh private-key-file <filepath> Local path that includes the private key file of the router.

set protocols rpki cache <address> ssh public-key-file <filepath> Local path that includes the public key file of the router.

Note   
When using SSH, known-hosts-file, private-key-file and public-key-file are mandatory options.

Example  
We can build route-maps for import based on these states. Here is a simple RPKI configuration, where routinator is the RPKI-validating “cache” server with ip 192.0.2.1:   
set protocols rpki cache 192.0.2.1 port '3323'   
set protocols rpki cache 192.0.2.1 preference '1'   
Here is an example route-map to apply to routes learned at import. In this filter we reject prefixes with the state invalid, and set a higher local-preference if the prefix is RPKI valid rather than merely notfound.

set policy route-map ROUTES-IN rule 10 action 'permit'

set policy route-map ROUTES-IN rule 10 match rpki 'valid'   
set policy route-map ROUTES-IN rule 10 set local-preference '300'   
set policy route-map ROUTES-IN rule 20 action 'permit'   
set policy route-map ROUTES-IN rule 20 match rpki 'notfound'   
set policy route-map ROUTES-IN rule 20 set local-preference '125'   
set policy route-map ROUTES-IN rule 30 action 'deny'   
set policy route-map ROUTES-IN rule 30 match rpki 'invalid'   
Once your routers are configured to reject RPKI-invalid prefixes, you can test whether the configuration is working correctly using the RIPE Labs RPKI Test experimental tool.

RSA-Keys  
RSA can be used for services such as key exchanges and for encryption purposes. To make IPSec work with dynamic address on one/both sides, we will have to use RSA keys for authentication. They are very fast and easy to setup.

First, on both routers run the operational command “generate pki key-pair install <key-pair nam>>”. You may choose different length than 2048 of course.

vyos@left# run generate pki key-pair install ipsec-LEFT   
Enter private key type: [rsa, dsa, ec] (Default: rsa)   
Enter private key bits: (Default: 2048)   
Note: If you plan to use the generated key on this router, do not encrypt the private key.

Do you want to encrypt the private key with a passphrase? [y/N] N   
Configure mode commands to install key pair:   
Do you want to install the public key? [Y/n] Y   
set pki key-pair ipsec-LEFT public key 'MIIBIjANBgkqh...'   
Do you want to install the private key? [Y/n] Y   
set pki key-pair ipsec-LEFT private key 'MIIEvgIBADAN...'   
[edit]   
Configuration commands will display. Note the command with the public key (set pki key-pair ipsec-LEFT public key ‘MIIBIjANBgkqh…’). Then do the same on the opposite router:   
vyos@left# run generate pki key-pair install ipsec-RIGHT   
Note the command with the public key (set pki key-pair ipsec-RIGHT public key   
‘FAAOCAQ8AMII…’).

The noted public keys should be entered on the opposite routers.

On the LEFT:   
set pki key-pair ipsec-RIGHT public key 'FAAOCAQ8AMII...'   
On the RIGHT:   
set pki key-pair ipsec-LEFT public key 'MIIBIjANBgkqh...'   
Now you are ready to setup IPsec. The key points:   
1. Since both routers do not know their effective public addresses, we set the local-address of the peer to “any”.

On the initiator, we set the peer address to its public address, but on the responder we 2.

only set the id.

3. On the initiator, we need to set the remote-id option so that it can identify IKE traffic from the responder correctly.

4. On the responder, we need to set the local id so that initiator can know who’s talking to it for the point #3 to work.   
On the LEFT (static address):   
set vpn ipsec interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128 set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1

set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2   
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128 set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer @RIGHT authentication id LEFT   
set vpn ipsec site-to-site peer @RIGHT authentication mode rsa   
set vpn ipsec site-to-site peer @RIGHT authentication rsa local-key ipsec-LEFT   
set vpn ipsec site-to-site peer @RIGHT authentication rsa remote-key ipsec-RIGHT   
set vpn ipsec site-to-site peer @RIGHT authentication remote-id RIGHT   
set vpn ipsec site-to-site peer @RIGHT default-esp-group MyESPGroup   
set vpn ipsec site-to-site peer @RIGHT ike-group MyIKEGroup   
set vpn ipsec site-to-site peer @RIGHT local-address 192.0.2.10   
set vpn ipsec site-to-site peer @RIGHT connection-type respond   
set vpn ipsec site-to-site peer @RIGHT tunnel 1 local prefix 192.168.99.1/32 # Additional loopback address on the local   
set vpn ipsec site-to-site peer @RIGHT tunnel 1 remote prefix 192.168.99.2/32 # Additional loopback address on the remote   
On the RIGHT (dynamic address):   
set vpn ipsec interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128 set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1

set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2   
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128 set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer 192.0.2.10 authentication id RIGHT   
set vpn ipsec site-to-site peer 192.0.2.10 authentication mode rsa   
set vpn ipsec site-to-site peer 192.0.2.10 authentication rsa local-key ipsec-RIGHT set vpn ipsec site-to-site peer 192.0.2.10 authentication rsa remote-key ipsec-LEFT set vpn ipsec site-to-site peer 192.0.2.10 authentication remote-id LEFT   
set vpn ipsec site-to-site peer 192.0.2.10 connection-type initiate   
set vpn ipsec site-to-site peer 192.0.2.10 default-esp-group MyESPGroup   
set vpn ipsec site-to-site peer 192.0.2.10 ike-group MyIKEGroup   
set vpn ipsec site-to-site peer 192.0.2.10 local-address any

set vpn ipsec site-to-site peer 192.0.2.10 tunnel 1 local prefix 192.168.99.2/32 # Additional loopback address on the local   
set vpn ipsec site-to-site peer 192.0.2.10 tunnel 1 remote prefix 192.168.99.1/32 # Additional loopback address on the remote

Salt  
VyOS supports op-mode and configuration via salt.

Without proxy it requires VyOS minion configuration and supports op-mode data: set service salt-minion id 'r14'   
set service salt-minion master '192.0.2.250'   
Check salt-keys on the salt master   
/ # salt-key --list-all   
Accepted Keys:   
r11   
Denied Keys:   
Unaccepted Keys:   
r14   
Rejected Keys:   
Accept minion key   
/ # salt-key --accept r14   
The following keys are going to be accepted:   
Unaccepted Keys:   
r14   
Proceed? [n/Y] y   
Key for minion r14 accepted.

Check that salt master can communicate with minions   
/ # salt '\*' test.ping   
r14:  
 True   
r11:  
 True   
At this step we can get some op-mode information from VyOS nodes: / # salt '\*' network.interface eth0   
r11:  
 |\_  
 ----------  
 address:  
 192.0.2.11  
 broadcast:  
 192.0.2.255  
 label:  
 eth0  
 netmask:

255.255.255.0   
r14:  
 |\_  
 ----------  
 address:  
 192.0.2.14  
 broadcast:  
 192.0.2.255  
 label:  
 eth0  
 netmask:  
 255.255.255.0

/ # salt r14 network.arp   
r14:  
 ----------  
 aa:bb:cc:dd:f3:db:  
 192.0.2.1  
 aa:bb:cc:dd:2e:80:  
 203.0.113.1   
Netmiko-proxy  
It is possible to configure VyOS via netmiko proxy module. It requires a minion with installed packet python3-netmiko module who has a connection to VyOS nodes. Salt-minion have to communicate with salt master   
Configuration  
Salt master configuration:   
/ # cat /etc/salt/master   
file\_roots:  
 base:  
 - /srv/salt/states

pillar\_roots:  
 base:  
 - /srv/salt/pillars   
Structure of /srv/salt:   
/ # tree /srv/salt/   
/srv/salt/   
|\_\_\_ pillars   
| |\_\_ r11-proxy.sls   
| |\_\_ top.sls   
|\_\_\_ states  
 |\_\_ commands.txt   
top.sls

/ # cat /srv/salt/pillars/top.sls   
base:  
 r11-proxy:  
 - r11-proxy   
r11-proxy.sls Includes parameters for connecting to salt-proxy minion   
/ # cat /srv/salt/pillars/r11-proxy.sls   
proxy:  
 proxytype: netmiko # how to connect to proxy minion, change it  
 device\_type: vyos #  
 host: 192.0.2.250  
 username: user  
 password: secret\_passwd   
commands.txt   
/ # cat /srv/salt/states/commands.txt   
set interfaces ethernet eth0 description 'WAN'   
set interfaces ethernet eth1 description 'LAN'   
Check that proxy minion is alive:   
/ # salt r11-proxy test.ping   
r11-proxy:  
 True   
/ #   
Examples  
Example of op-mode:   
/ # salt r11-proxy netmiko.send\_command 'show interfaces ethernet eth0 brief'   
host=192.0.2.14 device\_type=vyos username=vyos password=vyos   
r11-proxy:  
 Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down  
 Interface IP Address S/L Description  
 --------- ---------- --- -----------  
 eth0 192.0.2.14/24 u/u Upstream   
/ #   
Example of configuration:   
/ # salt r11-proxy netmiko.send\_config config\_commands=['set interfaces ethernet eth0 description Link\_to\_WAN'] commit=True host=192.0.2.14 device\_type=vyos username=vyos password=vyos   
r11-proxy:  
 configure  
 set interfaces ethernet eth0 description Link\_to\_WAN  
 [edit]  
 vyos@r14# commit  
 [edit]  
 vyos@r14#   
/ #   
Example of configuration commands from the file “/srv/salt/states/commands.txt”

/ # salt r11-proxy netmiko.send\_config config\_file=salt://commands.txt commit=True   
host=192.0.2.11 device\_type=vyos username=vyos password=vyos   
r11-proxy:  
 configure  
 set interfaces ethernet eth0 description 'WAN'  
 [edit]  
 vyos@r1# set interfaces ethernet eth1 description 'LAN'  
 [edit]  
 vyos@r1# commit  
 [edit]  
 vyos@r1#   
/ #   
Segment Routing  
Segment Routing (SR) is a network architecture that is similar to source-routing . In this architecture, the ingress router adds a list of segments, known as SIDs, to the packet as it enters the network. These segments represent different portions of the network path that the packet will take.

The SR segments are portions of the network path taken by the packet, and are called SIDs. At each node, the first SID of the list is read, executed as a forwarding function, and may be popped to let the next node read the next SID of the list. The SID list completely determines the path where the packet is forwarded.

Segment Routing can be applied to an existing MPLS-based data plane and defines a control plane network architecture. In MPLS networks, segments are encoded as MPLS labels and are added at the ingress router. These MPLS labels are then exchanged and populated by Interior Gateway Protocols (IGPs) like IS-IS or OSPF which are running on most ISPs.

Note   
Segment routing defines a control plane network architecture and can be applied to an existing MPLS based dataplane. In the MPLS networks, segments are encoded as MPLS labels and are imposed at the ingress router. MPLS labels are exchanged and populated by IGPs like IS-IS.Segment Routing as per RFC8667 for MPLS dataplane. It supports IPv4, IPv6 and ECMP and has been tested against Cisco & Juniper routers.however,this deployment is still EXPERIMENTAL for FRR.

IS-IS SR Configuration  
Segment routing (SR) is used by the IGP protocols to interconnect network devices, below configuration shows how to enable SR on IS-IS:   
Note   
Known limitations:   
No support for level redistribution (L1 to L2 or L2 to L1)   
No support for binding SID   
No support for SRLB   
Only one SRGB and default SPF Algorithm is supported  
 set protocols isis segment-routing global-block high-label-value <label-value>   
Set the Segment Routing Global Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.

set protocols isis segment-routing global-block low-label-value <label-value>   
Set the Segment Routing Global Block i.e. the low label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.

set protocols isis segment-routing local-block high-label-value <label-value>   
Set the Segment Routing Local Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.Segment Routing Local Block, The negative command always unsets both.

set protocols isis segment-routing local-block <low-label-value <label-value>   
Set the Segment Routing Local Block i.e. the low label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.Segment Routing Local Block, The negative command always unsets both.

set protocols isis segment-routing maximum-label-depth <1-16>   
Set the Maximum Stack Depth supported by the router. The value depend of the MPLS dataplane.

set protocols isis segment-routing prefix <address> index value <0-65535>   
A segment ID that contains an IP address prefix calculated by an IGP in the service provider core network. Prefix SIDs are globally unique, this value indentify it  
 set protocols isis segment-routing prefix <address> index <no-php-flag | explicit-null| n-flag-clear>   
this option allows to configure prefix-sid on SR. The ‘no-php-flag’ means NO Penultimate Hop Popping that allows SR node to request to its neighbor to not pop the label. The ‘explicit-null’ flag allows SR node to request to its neighbor to send IP packet with the EXPLICIT-NULL label.

The ‘n-flag-clear’ option can be used to explicitly clear the Node flag that is set by default for Prefix-SIDs associated to loopback addresses. This option is necessary to configure Anycast-SIDs.

show isis segment-routing node   
Show detailed information about all learned Segment Routing Nodes  
 show isis route prefix-sid   
Show detailed information about prefix-sid and label learned   
Note   
more information related IGP - IS-IS   
OSPF SR Configuration  
Segment routing (SR) is used by the IGP protocols to interconnect network devices, below configuration shows how to enable SR on OSPF:  
 set protocols ospf parameters opaque-lsa   
Enable the Opaque-LSA capability (rfc2370), necessary to transport label on IGP  
 set protocols ospf segment-routing global-block high-label-value <label-value>   
Set the Segment Routing Global Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.

set protocols ospf segment-routing global-block low-label-value <label-value>   
Set the Segment Routing Global Block i.e. the low label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.

set protocols ospf segment-routing local-block high-label-value <label-value>

Set the Segment Routing Local Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.Segment Routing Local Block, The negative command always unsets both.

set protocols ospf segment-routing local-block <low-label-value <label-value>   
Set the Segment Routing Local Block i.e. the low label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535.Segment Routing Local Block, The negative command always unsets both.

set protocols ospf segment-routing maximum-label-depth <1-16>   
Set the Maximum Stack Depth supported by the router. The value depend of the MPLS dataplane.

set protocols ospf segment-routing prefix <address> index value <0-65535>   
A segment ID that contains an IP address prefix calculated by an IGP in the service provider core network. Prefix SIDs are globally unique, this value indentify it  
 set protocols ospf segment-routing prefix <address> index <no-php-flag | explicit-null| n-flag-clear>   
this option allows to configure prefix-sid on SR. The ‘no-php-flag’ means NO Penultimate Hop Popping that allows SR node to request to its neighbor to not pop the label. The ‘explicit-null’ flag allows SR node to request to its neighbor to send IP packet with the EXPLICIT-NULL label.

The ‘n-flag-clear’ option can be used to explicitly clear the Node flag that is set by default for Prefix-SIDs associated to loopback addresses. This option is necessary to configure Anycast-SIDs.

Note   
more information related IGP - OSPF   
Configuration Example  
we described the configuration SR ISIS / SR OSPF using 2 connected with them to share label information.

Enable IS-IS with Segment Routing (Experimental)Node 1:   
set interfaces loopback lo address '192.168.255.255/32' set interfaces ethernet eth1 address '192.0.2.1/24'

set protocols isis interface eth1   
set protocols isis interface lo   
set protocols isis net '49.0001.1921.6825.5255.00'   
set protocols isis segment-routing global-block high-label-value '599'   
set protocols isis segment-routing global-block low-label-value '550'   
set protocols isis segment-routing prefix 192.168.255.255/32 index value '1' set protocols isis segment-routing prefix 192.168.255.255/32 index explicit-null set protocols mpls interface 'eth1'   
Node 2:   
set interfaces loopback lo address '192.168.255.254/32'   
set interfaces ethernet eth1 address '192.0.2.2/24'

set protocols isis interface eth1   
set protocols isis interface lo

set protocols isis net '49.0001.1921.6825.5254.00'   
set protocols isis segment-routing global-block high-label-value '599'   
set protocols isis segment-routing global-block low-label-value '550'   
set protocols isis segment-routing prefix 192.168.255.254/32 index value '2'   
set protocols isis segment-routing prefix 192.168.255.254/32 index explicit-null   
set protocols mpls interface 'eth1'   
This gives us MPLS segment routing enabled and labels for far end loopbacks:   
Node-1@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 ----------------------------------------------------------------------  
 552 SR (IS-IS) 192.0.2.2 IPv4 Explicit Null <-- Node-2 loopback learned on Node-1 15000 SR (IS-IS) 192.0.2.2 implicit-null  
 15001 SR (IS-IS) fe80::e87:6cff:fe09:1 implicit-null  
 15002 SR (IS-IS) 192.0.2.2 implicit-null  
 15003 SR (IS-IS) fe80::e87:6cff:fe09:1 implicit-null

Node-2@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 ---------------------------------------------------------------------  
 551 SR (IS-IS) 192.0.2.1 IPv4 Explicit Null <-- Node-1 loopback learned on Node-2 15000 SR (IS-IS) 192.0.2.1 implicit-null  
 15001 SR (IS-IS) fe80::e33:2ff:fe80:1 implicit-null  
 15002 SR (IS-IS) 192.0.2.1 implicit-null  
 15003 SR (IS-IS) fe80::e33:2ff:fe80:1 implicit-null   
Here is the routing tables showing the MPLS segment routing label operations:   
Node-1@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup  
 t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.2, eth1 inactive, weight 1, 00:07:48   
I>\* 192.168.255.254/32 [115/20] via 192.0.2.2, eth1, label IPv4 Explicit Null, weight 1, 00:03:39

Node-2@vyos:~$ show ip route isis   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

I 192.0.2.0/24 [115/20] via 192.0.2.1, eth1 inactive, weight 1, 00:07:46   
I>\* 192.168.255.255/32 [115/20] via 192.0.2.1, eth1, label IPv4 Explicit Null, weight 1, 00:03:43 Enable OSPF with Segment Routing (Experimental):  
Node 1   
set interfaces loopback lo address 10.1.1.1/32   
set interfaces ethernet eth0 address 192.168.0.1/24   
set protocols ospf area 0 network '192.168.0.0/24'   
set protocols ospf area 0 network '10.1.1.1/32'   
set protocols ospf parameters opaque-lsa   
set protocols ospf parameters router-id '10.1.1.1'   
set protocols ospf segment-routing global-block high-label-value '1100'   
set protocols ospf segment-routing global-block low-label-value '1000'   
set protocols ospf segment-routing prefix 10.1.1.1/32 index explicit-null   
set protocols ospf segment-routing prefix 10.1.1.1/32 index value '1'   
Node 2   
set interfaces loopback lo address 10.1.1.2/32   
set interfaces ethernet eth0 address 192.168.0.2/24   
set protocols ospf area 0 network '192.168.0.0/24'   
set protocols ospf area 0 network '10.1.1.2/32'   
set protocols ospf parameters opaque-lsa   
set protocols ospf parameters router-id '10.1.1.2'   
set protocols ospf segment-routing global-block high-label-value '1100'   
set protocols ospf segment-routing global-block low-label-value '1000'   
set protocols ospf segment-routing prefix 10.1.1.2/32 index explicit-null   
set protocols ospf segment-routing prefix 10.1.1.2/32 index value '2'   
This gives us MPLS segment routing enabled and labels for far end loopbacks:   
Node-1@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 -----------------------------------------------------------  
 1002 SR (OSPF) 192.168.0.2 IPv4 Explicit Null <-- Node-2 loopback learned on Node-1 15000 SR (OSPF) 192.168.0.2 implicit-null  
 15001 SR (OSPF) 192.168.0.2 implicit-null

Node-2@vyos:~$ show mpls table  
 Inbound Label Type Nexthop Outbound Label  
 -----------------------------------------------------------  
 1001 SR (OSPF) 192.168.0.1 IPv4 Explicit Null <-- Node-1 loopback learned on Node-2 15000 SR (OSPF) 192.168.0.1 implicit-null  
 15001 SR (OSPF) 192.168.0.1 implicit-null   
Here is the routing tables showing the MPLS segment routing label operations:   
Node-1@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,

f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

O 10.1.1.1/32 [110/0] is directly connected, lo, weight 1, 00:03:43   
O>\* 10.1.1.2/32 [110/1] via 192.168.0.2, eth0, label IPv4 Explicit Null, weight 1, 00:03:32 O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:43

Node-2@vyos:~$ show ip route ospf   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup t - trapped, o - offload failure

O>\* 10.1.1.1/32 [110/1] via 192.168.0.1, eth0, label IPv4 Explicit Null, weight 1, 00:03:36 O 10.1.1.2/32 [110/0] is directly connected, lo, weight 1, 00:03:51   
O 192.168.0.0/24 [110/1] is directly connected, eth0, weight 1, 00:03:51

SSTP Server  
SSTP is a form of VPN tunnel that provides a mechanism to transport PPP traffic through an SSL/TLS channel. SSL/TLS provides transport-level security with key negotiation, encryption and traffic integrity checking. The use of SSL/TLS over TCP port 443 allows SSTP to pass through virtually all firewalls and proxy servers except for authenticated web proxies.

SSTP is available for Linux, BSD, and Windows.

VyOS utilizes accel-ppp to provide SSTP server functionality. We support both local and RADIUS authentication.

As SSTP provides PPP via a SSL/TLS channel the use of either publically signed certificates as well as a private PKI is required.

Note   
All certificates should be stored on VyOS under /config/auth. If certificates are not stored in the /config directory they will not be migrated during a software update.

Certificates  
Self Signed CA  
To generate the CA, the server private key and certificates the following commands can be used.

vyos@vyos:~$ mkdir -p /config/user-data/sstp   
vyos@vyos:~$ openssl req -newkey rsa:4096 -new -nodes -x509 -days 3650 -keyout /config/user-data/sstp/server.key -out /config/user-data/sstp/server.crt

Generating a 4096 bit RSA private key   
.........................++   
...............................................................++

writing new private key to 'server.key'   
[...]   
Country Name (2 letter code) [AU]:   
State or Province Name (full name) [Some-State]:   
Locality Name (eg, city) []:   
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Organizational Unit Name (eg, section) []:   
Common Name (e.g. server FQDN or YOUR name) []:   
Email Address []:

vyos@vyos:~$ openssl req -new -x509 -key /config/user-data/sstp/server.key -out /config/user-data/sstp/ca.crt   
[...]   
Country Name (2 letter code) [AU]:   
State or Province Name (full name) [Some-State]:   
Locality Name (eg, city) []:   
Organization Name (eg, company) [Internet Widgits Pty Ltd]:   
Organizational Unit Name (eg, section) []:   
Common Name (e.g. server FQDN or YOUR name) []:   
Email Address []:   
Configuration  
 set vpn sstp authentication local-users username <user> password <pass>   
Create <user> for local authentication on this system. The users password will be set to <pass>.

set vpn sstp authentication local-users username <user> disable Disable <user> account.

set vpn sstp authentication local-users username <user> static-ip <address> Assign static IP address to <user> account.

set vpn sstp authentication local-users username <user> rate-limit download <bandwidth> Download bandwidth limit in kbit/s for <user>.

set vpn sstp authentication local-users username <user> rate-limit upload <bandwidth> Upload bandwidth limit in kbit/s for <user>.

set vpn sstp authentication protocols <pap | chap | mschap | mschap-v2>   
Require the peer to authenticate itself using one of the following protocols: pap, chap, mschap, mschap-v2.

set vpn sstp authentication mode <local | radius>   
Set authentication backend. The configured authentication backend is used for all queries.• radius: All authentication queries are handled by a configured RADIUS server.

• local: All authentication queries are handled locally.

set vpn sstp gateway-address <gateway>   
Specifies single <gateway> IP address to be used as local address of PPP interfaces.

set vpn sstp port <port>   
Specifies the port <port> that the SSTP port will listen on (default 443).

set vpn sstp client-ip-pool subnet <subnet>   
Use <subnet> as the IP pool for all connecting clients.

set vpn sstp client-ipv6-pool prefix <address> mask <number-of-bits>   
Use this comand to set the IPv6 address pool from which an SSTP client will get an IPv6 prefix of your defined length (mask) to terminate the SSTP endpoint at their side. The mask length can be set from 48 to 128 bit long, the default value is 64.

set vpn sstp client-ipv6-pool delegate <address> delegation-prefix <number-of-bits>   
Use this command to configure DHCPv6 Prefix Delegation (RFC3633) on SSTP. You will have to set your IPv6 pool and the length of the delegation prefix. From the defined IPv6 pool you will be handing out networks of the defined length (delegation-prefix). The length of the delegation prefix can be set from 32 to 64 bit long.

set vpn sstp name-server <address>   
Connected client should use <address> as their DNS server. This command accepts both IPv4 and IPv6 addresses. Up to two nameservers can be configured for IPv4, up to three for IPv6.

Maximum number of IPv4 nameservers   
SSL Certificates  
 set vpn sstp ssl ca-cert-file <file>   
Path to <file> pointing to the certificate authority certificate.

set vpn sstp ssl cert-file <file>   
Path to <file> pointing to the servers certificate (public portion).

PPP Settings  
 set vpn sstp ppp-options lcp-echo-failure <number>   
Defines the maximum <number> of unanswered echo requests. Upon reaching the value <number>, the session will be reset.

set vpn sstp ppp-options lcp-echo-interval <interval>   
If this option is specified and is greater than 0, then the PPP module will send LCP pings of the echo request every <interval> seconds.

set vpn sstp ppp-options lcp-echo-timeout   
Specifies timeout in seconds to wait for any peer activity. If this option specified it turns on adaptive lcp echo functionality and “lcp-echo-failure” is not used.

set vpn sstp ppp-options mppe <require | prefer | deny> Specifies MPPE negotioation preference.

• require - ask client for mppe, if it rejects drop connection  
• prefer - ask client for mppe, if it rejects don’t fail  
• deny - deny mppe   
Default behavior - don’t ask client for mppe, but allow it if client wants. Please note that RADIUS may override this option by MS-MPPE-Encryption-Policy attribute.

RADIUS  
Server  
 set vpn sstp authentication radius server <server> port <port>   
Configure RADIUS <server> and its required port for authentication requests.

set vpn sstp authentication radius server <server> key <secret>   
Configure RADIUS <server> and its required shared <secret> for communicating with the RADIUS server.

set vpn sstp authentication radius server <server> fail-time <time> Mark RADIUS server as offline for this given <time> in seconds.

set vpn sstp authentication radius server <server> disable Temporary disable this RADIUS server.

Options  
 set vpn sstp authentication radius acct-timeout <timeout>   
Timeout to wait reply for Interim-Update packets. (default 3 seconds)  
 set vpn sstp authentication radius dynamic-author server <address>   
Specifies IP address for Dynamic Authorization Extension server (DM/CoA)  
 set vpn sstp authentication radius dynamic-author port <port>   
Port for Dynamic Authorization Extension server (DM/CoA)  
 set vpn sstp authentication radius dynamic-author key <secret>   
Secret for Dynamic Authorization Extension server (DM/CoA)  
 set vpn sstp authentication radius max-try <number>   
Maximum number of tries to send Access-Request/Accounting-Request queries  
 set vpn sstp authentication radius timeout <timeout>   
Timeout to wait response from server (seconds)  
 set vpn sstp authentication radius nas-identifier <identifier>   
Value to send to RADIUS server in NAS-Identifier attribute and to be matched in DM/CoA requests.

set vpn sstp authentication radius nas-ip-address <address>   
Value to send to RADIUS server in NAS-IP-Address attribute and to be matched in DM/CoA requests. Also DM/CoA server will bind to that address.

set vpn sstp authentication radius source-address <address> Source IPv4 address used in all RADIUS server queires.

set vpn sstp authentication radius rate-limit attribute <attribute>   
Specifies which RADIUS server attribute contains the rate limit information. The default attribute is Filter-Id.

set vpn sstp authentication radius rate-limit enable Enables bandwidth shaping via RADIUS.

set vpn sstp authentication radius rate-limit vendor   
Specifies the vendor dictionary, dictionary needs to be in /usr/share/accel-ppp/radius.

Example  
 Use local user foo with password bar•  
• Client IP addresses will be provided from pool 192.0.2.0/25 set vpn sstp authentication local-users username vyos password vyos set vpn sstp authentication mode local   
set vpn sstp gateway-address 192.0.2.254   
set vpn sstp client-ip-pool subnet 192.0.2.0/25   
set vpn sstp name-server 10.0.0.1   
set vpn sstp name-server 10.0.0.2   
set vpn sstp ssl ca-cert-file /config/auth/ca.crt   
set vpn sstp ssl cert-file /config/auth/server.crt   
set vpn sstp ssl key-file /config/auth/server.key   
Testing SSTP

Once you have setup your SSTP server there comes the time to do some basic testing. The Linux client used for testing is called sstpc. sstpc requires a PPP configuration/peer file.

The following PPP configuration tests MSCHAP-v2:   
$ cat /etc/ppp/peers/vyos   
usepeerdns   
#require-mppe   
#require-pap   
require-mschap-v2   
noauth   
lock   
refuse-pap   
refuse-eap   
refuse-chap   
refuse-mschap   
#refuse-mschap-v2   
nobsdcomp   
nodeflate   
debug   
You can now “dial” the peer with the follwoing command: sstpc --log-level 4 --log-stderr --user vyos --password vyos vpn.example.com -- call vyos.

A connection attempt will be shown as:   
$ sstpc --log-level 4 --log-stderr --user vyos --password vyos vpn.example.com -- call vyos

Mar 22 13:29:12 sstpc[12344]: Resolved vpn.example.com to 192.0.2.1 Mar 22 13:29:12 sstpc[12344]: Connected to vpn.example.com   
Mar 22 13:29:12 sstpc[12344]: Sending Connect-Request Message Mar 22 13:29:12 sstpc[12344]: SEND SSTP CRTL PKT(14)   
Mar 22 13:29:12 sstpc[12344]: TYPE(1): CONNECT REQUEST, ATTR(1): Mar 22 13:29:12 sstpc[12344]: ENCAP PROTO(1): 6   
Mar 22 13:29:12 sstpc[12344]: RECV SSTP CRTL PKT(48)   
Mar 22 13:29:12 sstpc[12344]: TYPE(2): CONNECT ACK, ATTR(1): Mar 22 13:29:12 sstpc[12344]: CRYPTO BIND REQ(4): 40   
Mar 22 13:29:12 sstpc[12344]: Started PPP Link Negotiation   
Mar 22 13:29:15 sstpc[12344]: Sending Connected Message   
Mar 22 13:29:15 sstpc[12344]: SEND SSTP CRTL PKT(112)   
Mar 22 13:29:15 sstpc[12344]: TYPE(4): CONNECTED, ATTR(1):   
Mar 22 13:29:15 sstpc[12344]: CRYPTO BIND(3): 104   
Mar 22 13:29:15 sstpc[12344]: Connection Established

$ ip addr show ppp0   
164: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER\_UP> mtu 1452 qdisc fq\_codel state UNKNOWN group default qlen 3  
 link/ppp promiscuity 0  
 inet 100.64.2.2 peer 100.64.1.1/32 scope global ppp0

valid\_lft forever preferred\_lft forever   
Static  
Static routes are manually configured routes, which, in general, cannot be updated dynamically from information VyOS learns about the network topology from other routing protocols. However, if a link fails, the router will remove routes, including static routes, from the RIPB that used this interface to reach the next hop. In general, static routes should only be used for very simple network topologies, or to override the behavior of a dynamic routing protocol for a small number of routes. The collection of all routes the router has learned from its configuration or from its dynamic routing protocols is stored in the RIB. Unicast routes are directly used to determine the forwarding table used for unicast packet forwarding.

Static Routes  
 set protocols static route <subnet> next-hop <address>   
Configure next-hop <address> for an IPv4 static route. Multiple static routes can be created.

set protocols static route <subnet> next-hop <address> disable Disable this IPv4 static route entry.

set protocols static route <subnet> next-hop <address> distance <distance>   
Defines next-hop distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

Range is 1 to 255, default is 1.

Note   
Routes with a distance of 255 are effectively disabled and not installed into the kernel.

set protocols static route6 <subnet> next-hop <address>   
Configure next-hop <address> for an IPv6 static route. Multiple static routes can be created.

set protocols static route6 <subnet> next-hop <address> disable Disable this IPv6 static route entry.

set protocols static route6 <subnet> next-hop <address> distance <distance>   
Defines next-hop distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

Range is 1 to 255, default is 1.

Note   
Routes with a distance of 255 are effectively disabled and not installed into the kernel.

Interface Routes  
 set protocols static route <subnet> interface <interface>   
Allows you to configure the next-hop interface for an interface-based IPv4 static route.

<interface> will be the next-hop interface where traffic is routed for the given <subnet>.

set protocols static route <subnet> interface <interface> disable Disables interface-based IPv4 static route.

set protocols static route <subnet> interface <interface> distance <distance>   
Defines next-hop distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

Range is 1 to 255, default is 1.

set protocols static route6 <subnet> interface <interface>   
Allows you to configure the next-hop interface for an interface-based IPv6 static route.

<interface> will be the next-hop interface where traffic is routed for the given <subnet>.

set protocols static route6 <subnet> interface <interface> disable Disables interface-based IPv6 static route.

set protocols static route6 <subnet> interface <interface> distance <distance>   
Defines next-hop distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

Range is 1 to 255, default is 1.

Blackhole  
 set protocols static route <subnet> blackhole   
Use this command to configure a “black-hole” route on the router. A black-hole route is a route for which the system silently discard packets that are matched. This prevents networks leaking out public interfaces, but it does not prevent them from being used as a more specific route inside your network.

set protocols static route <subnet> blackhole distance <distance>   
Defines blackhole distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

set protocols static route6 <subnet> blackhole   
Use this command to configure a “black-hole” route on the router. A black-hole route is a route for which the system silently discard packets that are matched. This prevents networks leaking out public interfaces, but it does not prevent them from being used as a more specific route inside your network.

set protocols static route6 <subnet> blackhole distance <distance>   
Defines blackhole distance for this route, routes with smaller administrative distance are elected prior to those with a higher distance.

Alternate Routing Tables  
TBD   
Alternate routing tables are used with policy based routing by utilizing VRF.

ARP  
ARP is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address. This mapping is a critical function in the Internet protocol suite. ARP was defined in 1982 by RFC 826 which is Internet Standard STD 37.

In Internet Protocol Version 6 (IPv6) networks, the functionality of ARP is provided by the Neighbor Discovery Protocol (NDP).

To manipulate or display ARP table entries, the following commands are implemented.

Configure  
 set protocols static arp interface <interface> address <host> mac <mac>   
This will configure a static ARP entry always resolving <address> to <mac> for interface <interface>.

Example:   
set protocols static arp interface eth0 address 192.0.2.1 mac 01:23:45:67:89:01 Operation  
 show protocols static arp   
Display all known ARP table entries spanning across all interfaces   
vyos@vyos:~$ show protocols static arp

Address HWtype HWaddress Flags Mask Iface   
10.1.1.1 ether 00:53:00:de:23:2e C eth1   
10.1.1.100 ether 00:53:00:de:23:aa CM eth1  
 show protocols static arp interface eth1   
Display all known ARP table entries on a given interface only (eth1):   
vyos@vyos:~$ show protocols static arp interface eth1   
Address HWtype HWaddress Flags Mask Iface   
10.1.1.1 ether 00:53:00:de:23:2e C eth1   
10.1.1.100 ether 00:53:00:de:23:aa CM eth1   
Troubleshooting  
Sometimes things break or don’t work as expected. This section describes several troubleshooting tools provided by VyOS that can help when something goes wrong.

Connectivity Tests  
Basic Connectivity Tests  
Verifying connectivity can be done with the familiar ping and traceroute commands. The options for each are shown (the options for each command were displayed using the built-in help as described in the Command Line Interface section and are omitted from the output here):  
 ping <destination>   
Send ICMP echo requests to destination host. There are multiple options to ping, inkl. VRF support.

vyos@vyos:~$ ping 10.1.1.1   
Possible completions:  
 <Enter> Execute the current command  
 adaptive Ping options  
 allow-broadcast  
 audible  
 bypass-route  
 count  
 deadline  
 do-not-fragment  
 flood  
 interface  
 interval  
 mark  
 no-loopback  
 numeric  
 pattern  
 quiet  
 record-route  
 size  
 timestamp  
 tos  
 ttl

verbose  
 vrf  
 traceroute <destination>   
Trace path to target.

vyos@vyos:~$ traceroute   
Possible completions:  
 <hostname> Track network path to specified node  
 <x.x.x.x>  
 <h:h:h:h:h:h:h:h>  
 ipv4 Track network path to <hostname|IPv4 address>  
 ipv6 Track network path to <hostname|IPv6 address>   
Advanced Connectivity Tests  
 monitor traceroute <destination>   
However, another helper is available which combines ping and traceroute into a single tool. An example of its output is shown:   
vyos@vyos:~$ mtr 10.62.212.12

My traceroute [v0.85]   
vyos (0.0.0.0)   
Keys: Help Display mode Restart statistics Order of fields quit  
 Packets Pings   
Host Loss% Snt Last Avg Best Wrst StDev   
1. 10.11.110.4 0.0% 34 0.5 0.5 0.4 0.8 0.1   
2. 10.62.255.184 0.0% 34 1.1 1.0 0.9 1.4 0.1   
3. 10.62.255.71 0.0% 34 1.4 1.4 1.3 2.0 0.1   
4. 10.62.212.12 0.0% 34 1.6 1.6 1.6 1.7 0.0   
Note   
The output consumes the screen and will replace your command prompt.

Several options are available for changing the display output. Press h to invoke the built in help system. To quit, just press q and you’ll be returned to the VyOS command prompt.

IPv6 Topology Discovery  
IPv6 uses different techniques to discover its Neighbors/topology.

Router Discovery  
 force ipv6-rd interface <interface> [address <ipv6-address>] Discover routers via eth0.

Example:   
vyos@vyos:~$ force ipv6-rd interface eth0   
Soliciting ff02::2 (ff02::2) on eth0...

Hop limit : 60 ( 0x3c)   
Stateful address conf. : No   
Stateful other conf. : No   
Mobile home agent : No   
Router preference : high

Neighbor discovery proxy : No   
Router lifetime : 1800 (0x00000708) seconds   
Reachable time : unspecified (0x00000000)   
Retransmit time : unspecified (0x00000000)  
 Prefix : 240e:fe:8ca7:ea01::/64  
 On-link : Yes  
 Autonomous address conf.: Yes  
 Valid time : 2592000 (0x00278d00) seconds  
 Pref. time : 14400 (0x00003840) seconds  
 Prefix : fc00:470:f1cd:101::/64  
 On-link : Yes  
 Autonomous address conf.: Yes  
 Valid time : 2592000 (0x00278d00) seconds  
 Pref. time : 14400 (0x00003840) seconds  
 Recursive DNS server : fc00:470:f1cd::ff00  
 DNS server lifetime : 600 (0x00000258) seconds  
 Source link-layer address: 00:98:2B:F8:3F:11  
 from fe80::298:2bff:fef8:3f11   
Neighbor Discovery  
 force ipv6-nd interface <interface> address <ipv6-address>   
Example:   
vyos@vyos:~$ force ipv6-nd interface eth0 address fc00:470:f1cd:101::1

Soliciting fc00:470:f1cd:101::1 (fc00:470:f1cd:101::1) on eth0...

Target link-layer address: 00:98:2B:F8:3F:11 from fc00:470:f1cd:101::1   
Interface names  
If you find the names of your interfaces have changed, this could be because your MAC addresses have changed.

• For example, you have a VyOS VM with 4 Ethernet interfaces named eth0, eth1, eth2 and eth3. Then, you migrate your VyOS VM to a different host and find your interfaces now are eth4, eth5, eth6 and eth7.

One way to fix this issue taking control of the MAC addresses is: Log into VyOS and run this command to display your interface settings.

show interfaces detail   
Take note of MAC addresses.

Now, in order to update a MAC address in the configuration, run this command specifying the interface name and MAC address you want.

set interfaces eth0 hw-id 00:0c:29:da:a4:fe   
If it is a VM, go into the settings of the host and set the MAC address to the settings found in the config.boot file. You can also set the MAC to static if the host allows so.

• Another example could be when cloning VyOS VMs in GNS3 and you get into the same issue: interface names have changed.

And a more generic way to fix it is just deleting every MAC address at the configuration file of the cloned machine. They will be correctly regenerated automatically.

Monitoring  
VyOS features several monitoring tools.

vyos@vyos:~$ monitor   
Possible completions:  
 bandwidth Monitor interface bandwidth in real time  
 bandwidth-test  
 Initiate or wait for bandwidth test  
 cluster Monitor clustering service  
 command Monitor an operational mode command (refreshes every 2 seconds) conntrack-sync  
 Monitor conntrack-sync  
 content-inspection  
 Monitor Content-Inspection  
 dhcp Monitor Dynamic Host Control Protocol (DHCP)  
 dns Monitor a Domain Name Service (DNS) daemon  
 firewall Monitor Firewall  
 https Monitor the Secure Hypertext Transfer Protocol (HTTPS) service  
 lldp Monitor Link Layer Discovery Protocol (LLDP) daemon  
 log Monitor last lines of messages file  
 nat Monitor network address translation (NAT)  
 ndp Monitor the NDP information received by the router through the device openvpn Monitor OpenVPN  
 protocol Monitor routing protocols  
 snmp Monitor Simple Network Management Protocol (SNMP) daemon  
 stop-all Stop all current background monitoring processes  
 traceroute Monitor the path to a destination in realtime  
 traffic Monitor traffic dumps  
 vpn Monitor VPN  
 vrrp Monitor Virtual Router Redundancy Protocol (VRRP)  
 webproxy Monitor Webproxy service   
Traffic Dumps  
To monitor interface traffic, issue the monitor traffic interface <name> command, replacing <name> with your chosen interface.

vyos@vyos:~$ monitor traffic interface eth0   
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode   
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes   
15:54:28.581601 IP 192.168.0.1 > vyos: ICMP echo request, id 1870, seq 3848, length 64 15:54:28.581660 IP vyos > 192.168.0.1: ICMP echo reply, id 1870, seq 3848, length 64 15:54:29.583399 IP 192.168.0.1 > vyos: ICMP echo request, id 1870, seq 3849, length 64 15:54:29.583454 IP vyos > 192.168.0.1: ICMP echo reply, id 1870, seq 3849, length 64 ^C   
4 packets captured   
4 packets received by filter   
0 packets dropped by kernel

vyos@vyos:~$   
To quit monitoring, press Ctrl-c and you’ll be returned to the VyOS command prompt. Traffic can be filtered and saved.

vyos@vyos:~$ monitor traffic interface eth0   
Possible completions:  
 <Enter> Execute the current command  
 filter Monitor traffic matching filter conditions  
 save Save traffic dump from an interface to a file   
Interface Bandwidth Usage  
to take a quick view on the used bandwidth of an interface use the monitor bandwidth command   
vyos@vyos:~$ monitor bandwidth interface eth0   
show the following:  
 B (RX Bytes/second)   
198.00 .|....|.....................................................

165.00 .|....|.....................................................

132.00 ||..|.|.....................................................

99.00 ||..|.|.....................................................

66.00 |||||||.....................................................

33.00 |||||||.....................................................

1 5 10 15 20 25 30 35 40 45 50 55 60

KiB (TX Bytes/second)  
 3.67 ......|.....................................................

3.06 ......|.....................................................

2.45 ......|.....................................................

1.84 ......|.....................................................

1.22 ......|.....................................................

0.61 :::::||.....................................................

1 5 10 15 20 25 30 35 40 45 50 55 60   
Interface Performance  
To take a look on the network bandwidth between two nodes, the monitor bandwidth-test command is used to run iperf.

vyos@vyos:~$ monitor bandwidth-test   
Possible completions:  
 accept Wait for bandwidth test connections (port TCP/5001)  
 initiate Initiate a bandwidth test   
 The accept command opens a listening iperf server on TCP Port 5001•  
• The initiate command connects to that server to perform the test.

vyos@vyos:~$ monitor bandwidth-test initiate   
Possible completions:  
 <hostname> Initiate a bandwidth test to specified host (port TCP/5001) <x.x.x.x>  
 <h:h:h:h:h:h:h:h>

Monitor command  
The monitor command command allows you to repeatedly run a command to view a continuously refreshed output. The command is run and output every 2 seconds, allowing you to monitor the output continuously without having to re-run the command. This can be useful to follow routing adjacency formation.

vyos@router:~$ monitor command "show interfaces"   
Will clear the screen and show you the output of show interfaces every 2 seconds.

Every 2.0s: /opt/vyatta/bin/vyatta-op-cmd-wrapper Sun Mar 26 02:49:46 2019

Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down   
Interface IP Address S/L Description  
--------- ---------- --- -----------  
eth0 192.168.1.1/24 u/u   
eth0.5 198.51.100.4/24 u/u WAN   
lo 127.0.0.1/8 u/u  
 ::1/128   
vti0 172.25.254.2/30 u/u   
vti1 172.25.254.9/30 u/u   
Terminal/Console  
Sometimes you need to clear counters or statistics to troubleshoot better. To do this use the clear command in Operational mode.

to clear the console output   
vyos@vyos:~$ clear console   
to clear interface counters   
# clear all interfaces   
vyos@vyos:~$ clear interface ethernet counters   
# clear specific interface   
vyos@vyos:~$ clear interface ehternet eth0 counters   
The command follow the same logic as the set command in configuration mode.

# clear all counters of a interface type   
vyos@vyos:~$ clear interface <interface\_type> counters   
# clear counter of a interface in interface\_type   
vyos@vyos:~$ clear interface <interface\_type> <interace\_name> counters to clear counters on firewall rulesets or single rules   
vyos@vyos:~$ clear firewall name <ipv4 ruleset name> counters   
vyos@vyos:~$ clear firewall name <ipv4 ruleset name> rule <rule#> counters

vyos@vyos:~$ clear firewall ipv6-name <ipv6 ruleset name> counters   
vyos@vyos:~$ clear firewall ipv6-name <ipv6 ruleset name> rule <rule#> counters   
System Information  
Boot Steps  
VyOS 1.2 uses Debian Jessie as the base Linux operating system. Jessie was the first version of Debian that uses systemd as the default init system.

These are the boot steps for VyOS 1.2

1. The BIOS loads Grub (or isolinux for the Live CD)   
2. Grub then starts the Linux boot and loads the Linux Kernel /boot/vmlinuz   
3. Kernel Launches Systemd /lib/systemd/systemd   
4. Systemd loads the VyOS service file /lib/systemd/system/vyos-router.service   
5. The service file launches the VyOS router init script /usr/libexec/vyos/init/vyos-router - this is part of the vyatta-cfg Debian package   
1. Starts FRR - successor to GNU Zebra and Quagga   
2. Initialises the boot configuration file - copies over config.boot.default if there is no configuration   
3. Runs the configuration migration, if the configuration is for an older version of VyOS 4. Runs The pre-config script, if there is one /config/scripts/vyos-preconfig-bootup.script 5. If the config file was upgraded, runs any post upgrade scripts /config/scripts/post-upgrade.d   
 Starts rl-system and firewall 6.

7. Mounts the /boot partition   
8. The boot configuration file is then applied by /opt/vyatta/sbin/ vyatta-boot-config-loader/opt/vyatta/etc/config/config.boot   
1. The config loader script writes log entries to /var/log/vyatta-config-loader.log 9.   
 Runs telinit q to tell the init system to reload /etc/inittab 10. Finally it runs the post-config script /config/scripts/vyos-postconfig-bootup.script

VRF  
VRF devices combined with ip rules provides the ability to create virtual routing and forwarding domains (aka VRFs, VRF-lite to be specific) in the Linux network stack. One use case is the multi-tenancy problem where each tenant has their own unique routing tables and in the very least need different default gateways.

Configuration  
A VRF device is created with an associated route table. Network interfaces are then enslaved to a VRF device.

set vrf name <name>   
Create new VRF instance with <name>. The name is used when placing individual interfaces into the VRF.

set vrf name <name> table <id>   
Configured routing table <id> is used by VRF <name>.

Note   
A routing table ID can not be modified once it is assigned. It can only be changed by deleting and re-adding the VRF instance.

set vrf bind-to-all   
By default the scope of the port bindings for unbound sockets is limited to the default VRF. That is, it will not be matched by packets arriving on interfaces enslaved to a VRF and processes may bind to the same port if they bind to a VRF.

TCP & UDP services running in the default VRF context (ie., not bound to any VRF device) can work across all VRF domains by enabling this option.

Zebra/Kernel route filtering  
Zebra supports prefix-lists and Route Mapss to match routes received from other FRR components. The permit/deny facilities provided by these commands can be used to filter which routes zebra will install in the kernel.

set vrf <name> ip protocol <protocol> route-map <route-map> Apply a route-map filter to routes for the specified protocol.

The following protocols can be used: any, babel, bgp, connected, eigrp, isis, kernel, ospf, rip, static, table   
Note   
If you choose any as the option that will cause all protocols that are sending routes to zebra.

set vrf <name> ipv6 protocol <protocol> route-map <route-map> Apply a route-map filter to routes for the specified protocol.

The following protocols can be used: any, babel, bgp, connected, isis, kernel, ospfv3, ripng, static, table   
Note   
If you choose any as the option that will cause all protocols that are sending routes to zebra.

Interfaces  
When VRFs are used it is not only mandatory to create a VRF but also the VRF itself needs to be assigned to an interface.

set interfaces <dummy | ethernet | bonding | bridge | pppoe> <interface> vrf <name> Assign interface identified by <interface> to VRF named <name>.

Routing  
Note   
VyOS 1.4 (sagitta) introduced dynamic routing support for VRFs.

Currently dynamic routing is supported for the following protocols:  
• BGP  
• IS-IS  
• OSPF   
 OSPFv3 (IPv6)  
•• Static   
The CLI configuration is same as mentioned in above articles. The only difference is, that each routing protocol used, must be prefixed with the vrf name <name> command.

Example  
The following commands would be required to set options for a given dynamic routing protocol inside a given vrf:  
• BGP: set vrf name <name> protocols bgp ...

• IS-IS: set vrf name <name> protocols isis ...

• OSPF: set vrf name <name> protocols ospf ...

• OSPFv3 (IPv6): set vrf name <name> protocols ospfv3 ...

• Static: set vrf name <name> protocols static ...

Operation  
It is not sufficient to only configure a VRF but VRFs must be maintained, too. For VRF maintenance the following operational commands are in place.

show vrf

Lists VRFs that have been created   
vyos@vyos:~$ show vrf   
VRF name state mac address flags interfaces  
-------- ----- ----------- ----- ----------  
blue up 00:53:12:d8:74:24 noarp,master,up,lower\_up dum200,eth0.302   
red up 00:53:de:02:df:aa noarp,master,up,lower\_up   
dum100,eth0.300,bond0.100,peth0   
Note   
Command should probably be extended to list also the real interfaces assigned to this one VRF to get a better overview.

show vrf <name>   
vyos@vyos:~$ show vrf name blue   
VRF name state mac address flags interfaces  
-------- ----- ----------- ----- ----------  
blue up 00:53:12:d8:74:24 noarp,master,up,lower\_up dum200,eth0.302 show ip route vrf <name>   
Display IPv4 routing table for VRF identified by <name>.

vyos@vyos:~$ show ip route vrf blue   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,  
 F - PBR, f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued route, r - rejected route

VRF blue:   
K 0.0.0.0/0 [255/8192] unreachable (ICMP unreachable), 00:00:50 S>\* 172.16.0.0/16 [1/0] via 192.0.2.1, dum1, 00:00:02   
C>\* 192.0.2.0/24 is directly connected, dum1, 00:00:06  
 show ipv6 route vrf <name>   
Display IPv6 routing table for VRF identified by <name>.

vyos@vyos:~$ show ipv6 route vrf red   
Codes: K - kernel route, C - connected, S - static, R - RIPng,  
 O - OSPFv3, I - IS-IS, B - BGP, N - NHRP, T - Table,  
 v - VNC, V - VNC-Direct, A - Babel, D - SHARP, F - PBR,  
 f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued route, r - rejected route

VRF red:   
K ::/0 [255/8192] unreachable (ICMP unreachable), 00:43:20   
C>\* 2001:db8::/64 is directly connected, dum1, 00:02:19   
C>\* fe80::/64 is directly connected, dum1, 00:43:19   
K>\* ff00::/8 [0/256] is directly connected, dum1, 00:43:19  
 ping <host> vrf <name>   
The ping command is used to test whether a network host is reachable or not.

Ping uses ICMP protocol’s mandatory ECHO\_REQUEST datagram to elicit an ICMP   
ECHO\_RESPONSE from a host or gateway. ECHO\_REQUEST datagrams (pings) will have an IP and ICMP header, followed by “struct timeval” and an arbitrary number of pad bytes used to fill out the packet.

When doing fault isolation with ping, you should first run it on the local host, to verify that the local network interface is up and running. Then, continue with hosts and gateways further down the road towards your destination. Round-trip time and packet loss statistics are computed. Duplicate packets are not included in the packet loss calculation, although the round-trip time of these packets is used in calculating the minimum/ average/maximum round-trip time numbers.

Note   
Ping command can be interrupted at any given time using <Ctrl>+c. A brief statistic is shown afterwards.

vyos@vyos:~$ ping 192.0.2.1 vrf red   
PING 192.0.2.1 (192.0.2.1) 56(84) bytes of data.

64 bytes from 192.0.2.1: icmp\_seq=1 ttl=64 time=0.070 ms   
64 bytes from 192.0.2.1: icmp\_seq=2 ttl=64 time=0.078 ms   
^C  
--- 192.0.2.1 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 4ms   
rtt min/avg/max/mdev = 0.070/0.074/0.078/0.004 ms  
 traceroute vrf <name> [ipv4 | ipv6] <host>   
Displays the route packets taken to a network host utilizing VRF instance identified by <name>.

When using the IPv4 or IPv6 option, displays the route packets taken to the given hosts IP address family. This option is useful when the host is specified as a hostname rather than an IP address.

force vrf <name>   
Join a given VRF. This will open a new subshell within the specified VRF.

The prompt is adjusted to reflect this change in both config and op-mode.

vyos@vyos:~$ force vrf blue   
vyos@vyos(vrf:blue):~$   
Example  
VRF route leaking  
The following example topology was built using EVE-NG.

VRF route leaking  
• PC1 is in the default VRF and acting as e.g. a “fileserver”  
• PC2 is in VRF blue which is the development department  
• PC3 and PC4 are connected to a bridge device on router R1 which is in VRF red. Say this is the HR department.

• R1 is managed through an out-of-band network that resides in VRF mgmt Configuration  
set interfaces bridge br10 address '10.30.0.254/24'   
set interfaces bridge br10 member interface eth3

set interfaces bridge br10 member interface eth4 set interfaces bridge br10 vrf 'red'

set interfaces ethernet eth0 address 'dhcp'   
set interfaces ethernet eth0 vrf 'mgmt'   
set interfaces ethernet eth1 address '10.0.0.254/24' set interfaces ethernet eth2 address '10.20.0.254/24' set interfaces ethernet eth2 vrf 'blue'

set protocols static route 10.20.0.0/24 interface eth2 vrf 'blue' set protocols static route 10.30.0.0/24 interface br10 vrf 'red'

set service ssh disable-host-validation   
set service ssh vrf 'mgmt'

set system name-server 'eth0'

set vrf name blue protocols static route 10.0.0.0/24 interface eth1 vrf 'default' set vrf name blue table '3000'   
set vrf name mgmt table '1000'   
set vrf name red protocols static route 10.0.0.0/24 interface eth1 vrf 'default' set vrf name red table '2000'   
VRF and NAT  
Configuration  
set interfaces ethernet eth0 address '172.16.50.12/24'   
set interfaces ethernet eth0 vrf 'red'

set interfaces ethernet eth1 address '192.168.130.100/24' set interfaces ethernet eth1 vrf 'blue'

set nat destination rule 110 description 'NAT ssh- INSIDE'   
set nat destination rule 110 destination port '2022'   
set nat destination rule 110 inbound-interface 'eth0'   
set nat destination rule 110 protocol 'tcp'   
set nat destination rule 110 translation address '192.168.130.40'

set nat source rule 100 outbound-interface 'eth0'   
set nat source rule 100 protocol 'all'   
set nat source rule 100 source address '192.168.130.0/24' set nat source rule 100 translation address 'masquerade'

set service ssh vrf 'red'

set vrf bind-to-all

set vrf name blue protocols static route 0.0.0.0/0 next-hop 172.16.50.1 vrf 'red' set vrf name blue protocols static route 172.16.50.0/24 interface eth0 vrf 'red' set vrf name blue table '1010'

set vrf name red protocols static route 0.0.0.0/0 next-hop 172.16.50.1   
set vrf name red protocols static route 192.168.130.0/24 interface eth1 vrf 'blue'   
set vrf name red table '2020'   
Operation  
After committing the configuration we can verify all leaked routes are installed, and try to ICMP ping PC1 from PC3.

PCS> ping 10.0.0.1

84 bytes from 10.0.0.1 icmp\_seq=1 ttl=63 time=1.943 ms 84 bytes from 10.0.0.1 icmp\_seq=2 ttl=63 time=1.618 ms 84 bytes from 10.0.0.1 icmp\_seq=3 ttl=63 time=1.745 ms VPCS> show ip

NAME : VPCS[1]   
IP/MASK : 10.30.0.1/24   
GATEWAY : 10.30.0.254   
DNS :   
MAC : 00:50:79:66:68:0f   
VRF default routing table  
vyos@R1:~$ show ip route   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,  
 F - PBR, f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup

C>\* 10.0.0.0/24 is directly connected, eth1, 00:07:44   
S>\* 10.20.0.0/24 [1/0] is directly connected, eth2 (vrf blue), weight 1, 00:07:38 S>\* 10.30.0.0/24 [1/0] is directly connected, br10 (vrf red), weight 1, 00:07:38 VRF red routing table  
vyos@R1:~$ show ip route vrf red   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,  
 F - PBR, f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup

VRF red:   
K>\* 0.0.0.0/0 [255/8192] unreachable (ICMP unreachable), 00:07:57   
S>\* 10.0.0.0/24 [1/0] is directly connected, eth1 (vrf default), weight 1, 00:07:40

C>\* 10.30.0.0/24 is directly connected, br10, 00:07:54   
VRF blue routing table  
vyos@R1:~$ show ip route vrf blue   
Codes: K - kernel route, C - connected, S - static, R - RIP,  
 O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,  
 T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,  
 F - PBR, f - OpenFabric,  
 > - selected route, \* - FIB route, q - queued, r - rejected, b - backup

VRF blue:   
K>\* 0.0.0.0/0 [255/8192] unreachable (ICMP unreachable), 00:08:00   
S>\* 10.0.0.0/24 [1/0] is directly connected, eth1 (vrf default), weight 1, 00:07:44   
C>\* 10.20.0.0/24 is directly connected, eth2, 00:07:53   
L3VPN VRFs  
L3VPN VRFs bgpd supports for IPv4 RFC 4364 and IPv6 RFC 4659. L3VPN routes, and their associated VRF MPLS labels, can be distributed to VPN SAFI neighbors in the default, i.e., non VRF, BGP instance. VRF MPLS labels are reached using core MPLS labels which are distributed using LDP or BGP labeled unicast. bgpd also supports inter-VRF route leaking.

VRF Route Leaking  
BGP routes may be leaked (i.e. copied) between a unicast VRF RIB and the VPN SAFI RIB of the default VRF for use in MPLS-based L3VPNs. Unicast routes may also be leaked between any VRFs (including the unicast RIB of the default BGP instance). A shortcut syntax is also available for specifying leaking from one VRF to another VRF using the default instance’s VPN RIB as the intemediary . A common application of the VRF-VRF feature is to connect a customer’s private routing domain to a provider’s VPN service. Leaking is configured from the point of view of an individual VRF: import refers to routes leaked from VPN to a unicast VRF, whereas export refers to routes leaked from a unicast VRF to VPN.

Note   
Routes exported from a unicast VRF to the VPN RIB must be augmented by two parameters: an RD / RTLIST   
Configuration for these exported routes must, at a minimum, specify these two parameters.

Configuration  
Configuration of route leaking between a unicast VRF RIB and the VPN SAFI RIB of the default VRF is accomplished via commands in the context of a VRF address-family.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> rd vpn export <asn:nn|address:nn>   
Specifies the route distinguisher to be added to a route exported from the current unicast VRF to VPN.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> route-target vpn <import|export|both> [RTLIST]   
Specifies the route-target list to be attached to a route (export) or the route-target list to match against (import) when exporting/importing between the current unicast VRF and VPN.The RTLIST is a space-separated list of route-targets, which are BGP extended community values as described in Extended Communities Attribute.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> label vpn export <0-1048575|auto>   
Enables an MPLS label to be attached to a route exported from the current unicast VRF to VPN.

If the value specified is auto, the label value is automatically assigned from a pool maintained.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> label vpn allocation-mode per-nexthop   
Select how labels are allocated in the given VRF. By default, the per-vrf mode is selected, and one label is used for all prefixes from the VRF. The per-nexthop will use a unique label for all prefixes that are reachable via the same nexthop.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> route-map vpn <import|export> [route-map <name>]   
Specifies an optional route-map to be applied to routes imported or exported between the current unicast VRF and VPN.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> <import| export> vpn   
Enables import or export of routes between the current unicast VRF and VPN.

set vrf name <name> protocols bgp address-family <ipv4-unicast|ipv6-unicast> import vrf <name>   
Shortcut syntax for specifying automatic leaking from vrf VRFNAME to the current VRF using the VPN RIB as intermediary. The RD and RT are auto derived and should not be specified explicitly for either the source or destination VRF’s.

set vrf name <name> protocols bgp interface <interface> mpls forwarding   
It is possible to permit BGP install VPN prefixes without transport labels. This configuration will install VPN prefixes originated from an e-bgp session, and with the next-hop directly connected.

Operation  
It is not sufficient to only configure a L3VPN VRFs but L3VPN VRFs must be maintained, too.For L3VPN VRF maintenance the following operational commands are in place.

show bgp <ipv4|ipv6> vpn   
Print active IPV4 or IPV6 routes advertised via the VPN SAFI.

BGP table version is 2, local router ID is 10.0.1.1, vrf id 0   
Default local pref 100, local AS 65001   
Status codes: s suppressed, d damped, h history, \* valid, > best, = multipath, i internal, r RIB-failure, S Stale, R Removed   
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self   
Origin codes: i - IGP, e - EGP, ? - incomplete

Network Next Hop Metric LocPrf Weight Path Route Distinguisher: 10.50.50.1:1011   
\*>i10.50.50.0/24 10.0.0.7 0 100 0 i  
 UN=10.0.0.7 EC{65035:1011} label=80 type=bgp, subtype=0 Route Distinguisher: 10.60.60.1:1011   
\*>i10.60.60.0/24 10.0.0.10 0 100 0 i  
 UN=10.0.0.10 EC{65035:1011} label=80 type=bgp, subtype=0 show bgp <ipv4|ipv6> vpn summary

Print a summary of neighbor connections for the specified AFI/SAFI combination. BGP router identifier 10.0.1.1, local AS number 65001 vrf-id 0   
BGP table version 0   
RIB entries 9, using 1728 bytes of memory   
Peers 4, using 85 KiB of memory   
Peer groups 1, using 64 bytes of memory

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd PfxSnt 10.0.0.7 4 65001 2860 2870 0 0 0 1d23h34m 2 10

VyOS API  
For configuration and enabling the API see HTTP-API   
Authentication  
All endpoints only listen on HTTP POST requests and the API KEY must set as key in the formdata.

Below see one example for curl and one for python. The rest of the documentation is reduced to curl.

curl --location --request POST 'https://vyos/retrieve' \  
--form data='{"op": "showConfig", "path": []}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'   
import requests   
url = "https://vyos/retrieve"   
payload={'data': '{"op": "showConfig", "path": []}',  
 'key': 'MY-HTTPS-API-PLAINTEXT-KEY'  
 }   
headers = {}   
response = requests.request("POST", url, headers=headers, data=payload) print(response.text)   
API Endpoints  
/retrieve  
With the retrieve endpoint you get parts or the whole configuration.

To get the whole configuration, pass an empty list to the path field curl --location --request POST 'https://vyos/retrieve' \  
--form data='{"op": "showConfig", "path": []}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response (shorted)   
{  
 "success": true,  
 "data": {  
 "interfaces": {  
 "ethernet": {  
 "eth0": {

"address": "dhcp",  
 "duplex": "auto",  
 "hw-id": "50:00:00:01:00:00",  
 "speed": "auto"  
 },  
 "eth1": {  
 "duplex": "auto",  
 "hw-id": "50:00:00:01:00:01",  
 "speed": "auto"  
 ...

},  
 "error": null   
}   
To only get a part of the configuration, for example system syslog.

curl -k --location --request POST 'https://vyos/retrieve' \  
--form data='{"op": "showConfig", "path": ["system", "syslog"]}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": {  
 "global": {  
 "facility": {  
 "all": {  
 "level": "info"  
 },  
 "protocols": {  
 "level": "debug"  
 }  
 }  
 }  
 },  
 "error": null   
}   
if you just want the Value of a multi-valued node, use the returnValues operation. For example, get the addresses of a dum0 interface.

curl -k --location --request POST 'https://vyos/retrieve' \  
--form data='{"op": "returnValues", "path": ["interfaces","dummy","dum0","address"]}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

respone:   
{

"success": true,  
 "data": [  
 "10.10.10.10/24",  
 "10.10.10.11/24",  
 "10.10.10.12/24"  
 ],  
 "error": null   
}   
/reset  
The reset endpoint run a reset command.

curl --location --request POST 'https://vyos/reset' \  
--form data='{"op": "reset", "path": ["ip", "bgp", "192.0.2.11"]}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

respone:   
{  
 "success": true,  
 "data": "",  
 "error": null   
}   
/image  
To add or delete an image, use the /image endpoint.

add an image   
curl -k --location --request POST 'https://vyos/image' \  
--form data='{"op": "add", "url": "https://downloads.vyos.io/rolling/current/amd64/vyos-rolling-latest.iso"}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

respone (shorted):   
{  
 "success": true,  
 "data": "Trying to fetch ISO file from https://downloads.vyos.io/rolling-latest.iso\n ...

Setting up grub configuration...\nDone.\n",  
 "error": null   
}   
delete an image, for example 1.3-rolling-202006070117   
curl -k --location --request POST 'https://vyos/image' \  
--form data='{"op": "delete", "name": "1.3-rolling-202006070117"}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,

"data": "Deleting the \"1.3-rolling-202006070117\" image...\nDone\n", "error": null   
}   
/show  
The /show endpoint is to show everything in the operational mode. For example, show which images are installed.

curl -k --location --request POST 'https://vyos/show' \--form data='{"op": "show", "path": ["system", "image"]}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": "The system currently has the following image(s) installed:\n\n 1: 1.4-rolling-202102280559 (default boot)\n  
 2: 1.4-rolling-202102230218\n  
 3: 1.3-beta-202102210443\n\n",  
 "error": null   
}   
/generate  
The generate endpoint run a generate command.

curl -k --location --request POST 'https://vyos/generate' \  
--form data='{"op": "generate", "path": ["wireguard", "default-keypair"]}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": "",  
 "error": null   
}   
/configure  
You can pass a set, delete or comment command to the /configure endpoint.

set a single command   
curl -k --location --request POST 'https://vyos/configure' \  
--form data='{"op": "set", "path": ["interfaces", "dummy", "dum1", "address", "10.11.0.1/32"]}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": null,  
 "error": null

}   
delete a single command   
curl -k --location --request POST 'https://vyos/configure' \  
--form data='{"op": "delete", "path": ["interfaces", "dummy", "dum1", "address", "10.11.0.1/32"]}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": null,  
 "error": null   
}   
The API pushes every request to a session and commit it. But some of VyOS components like DHCP and PPPoE Servers, IPSec, VXLAN, and other tunnels require full configuration for commit.

The endpoint will process multiple commands when you pass them as a list to the data field.

curl -k --location --request POST 'https://vyos/configure' \  
--form data='[{"op": "set","path":["interfaces","vxlan","vxlan1","remote","203.0.113.99"]}, {"op": "set","path":["interfaces","vxlan","vxlan1","vni","1"]}]' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": null,  
 "error": null   
}   
/config-file  
The endpoint /config-file is to save or load a configuration.

Save a running configuration to the startup configuration. When you don’t specify the file when saving, it saves to /config/config.boot.

curl -k --location --request POST 'https://vyos/config-file' \--form data='{"op": "save"}' \  
--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": "Saving configuration to '/config/config.boot'...\nDone\n", "error": null   
}   
Save a running configuration to a file.

curl -k --location --request POST 'https://vyos/config-file' \--form data='{"op": "save", "file": "/config/test.config"}' \

--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": "Saving configuration to '/config/test.config'...\nDone\n", "error": null   
}   
To Load a configuration file.

curl -k --location --request POST 'https://vyos/config-file' \--form data='{"op": "load", "file": "/config/test.config"}' \--form key='MY-HTTPS-API-PLAINTEXT-KEY'

response:   
{  
 "success": true,  
 "data": null,  
 "error": null   
}