|  |
| --- |
| **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 . |
| vyos@vyos:~$ show configuration interfaces {  ethernet eth0 {  address dhcp  hw-id 00:53:00:00:aa:01  }  loopback lo {  }  }  service {  ssh {  port 22  }  }  system { |

**Seeing and navigating the configuration**

|  |
| --- |
| **show configuration** |

|  |
| --- |
| View the current active configuration, also known as the running configuration, from the |

|  |
| --- |
| operational mode. |

|  |  |
| --- | --- |
| } | 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. Previous Next |

|  |
| --- |
| **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** |

|  |  |
| --- | --- |
| **Configuration**  **Common interface configuration**  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 <interface> address <address>** |

|  |
| --- |
| 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. |

|  |
| --- |
| 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 |

|  |  |  |
| --- | --- | --- |
| **Note** |  | segment. |
| **dhcpv6** interface address is received by DHCPv6 from a DHCPv6 server on |
| this segment. |

|  |
| --- |
| 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:

|  |  |  |
| --- | --- | --- |
| **Note** |  | 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)). |

|  |
| --- |
| 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 |

|  |  |  |
| --- | --- | --- |
| **Note** |  | segment. |
| **dhcpv6** interface address is received by DHCPv6 from a DHCPv6 server on |
| this segment. |

|  |
| --- |
| 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 |

|  |  |  |
| --- | --- | --- |
| **Note** |  | segment. |
| **dhcpv6** interface address is received by DHCPv6 from a DHCPv6 server on |
| this segment. |

|  |
| --- |
| 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

|  |  |  |
| --- | --- | --- |
| **Note** |  | **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. |

|  |
| --- |
| 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.

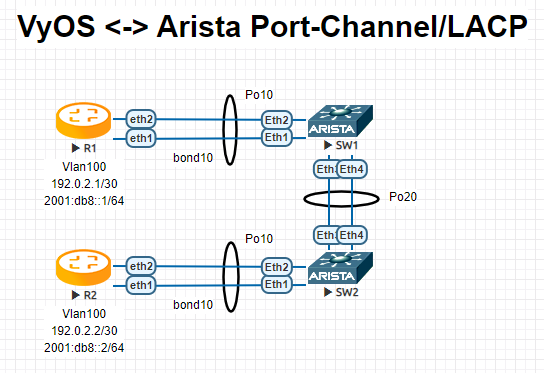
|  |
| --- |
| # Create trunk with 2 member interfaces (interface 1 and 2) and LACP trunk 1-2 Trk1 LACP |

|  |
| --- |
| # 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 |